Advances in Greedy Algorithms

Advances in Greedy Algorithms

Edited by
Witold Bednorz

I-Tech

Published by In-Teh

In-Teh is Croatian branch of I-Tech Education and Publishing KG, Vienna, Austria.

Abstracting and non-profit use of the material is permitted with credit to the source. Statements and opinions expressed in the chapters are these of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published articles. Publisher assumes no responsibility liability for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained inside. After this work has been published by the In-Teh, authors have the right to republish it, in whole or part, in any publication of which they are an author or editor, and the make other personal use of the work.

© 2008 In-teh
www.in-teh.org
Additional copies can be obtained from:
publication@ars-journal.com

First published November 2008
Printed in Croatia

A catalogue record for this book is available from the University Library Rijeka under no. 120115050
Advances in Greedy Algorithms, Edited by Witold Bednorz
p. cm.
ISBN 978-953-7619-27-5
1. Advances in Greedy Algorithms, Witold Bednorz

Preface

The greedy algorithm is one of the simplest approaches to solve the optizmization problem in which we want to determine the global optimum of a given function by a sequence of steps where at each stage we can make a choice among a class of possible decisions. In the greedy method the choice of the optimal decision is made on the information at hand without worrying about the effect these decisions may have in the future. Greedy algorithms are easy to invent, easy to implement and most of the time quite efficient. However there are many problems that cannot be solved correctly by the greedy approach. The common example of the greedy concept is the problem of 'Making Change' in which we want to make a change of a given amount using the minimum number of US coins. We can use five different values: dollars (100 cents), quarters (25 cents), dimes (10 cents), nickels (5 cents) and pennies (1 cent). The greedy algorithm is to take the largest possible amount of coins of a given value starting from the highest one (100 cents). It is easy to see that the greedy strategy is optimal in this setting, indeed for proving this it suffices to use the induction principle which works well because in each step either the procedure has ended or there is at least one coin we can use of the actual value. It means that the problem has a certain optimal substructure, which makes the greedy algorithm effective. However a slight modification of 'Making Change', e.g. where one value is missing, may turn the greedy strategy to be the worst choice. Therefore there are obvious limits for using the greedy method: whenever there is no optimal substructure of the problem we cannot hope that the greedy algorithm will work. On the other hand there is a lot of problems where the greedy strategy works unexpectedly well and the purpose of this book is to communicate various results in this area. The key point is the simplicity of the approach which makes the greedy algorithm a natural first choice to analyze the given problem. In this book there are discussed several algorithmic questions in: biology, combinatorics, networking, scheduling or even pure mathematics, where the greedy algorithm can be used to produce the optimal or nearly optimal answer.

The book was written in 2008 by the numerous authors who contributed the publication by presenting their researches in a form of a self-contained chapters. The idea was to coordinate the international project where specialists all over the world can share their knowledge on the greedy algorithms theory. Each chapter comprises a separate study on some optimization problem giving both an introductory look into the theory the problem comes from and some new developments invented by author(s). Usually some elementary knowledge is assumed, yet all the required facts are quoted mostly in examples, remarks or theorems. The publication may be useful for all graduates and undergraduates interested in the algorithmic theory with the focus on the greedy approach and applications of this

method to various concrete examples. Most of scientists involved in the project are young at the full strength of their career, hence the presented content is fresh and acquaints with the new directions where the theory of greedy algorithms evolves to.

On the behalf of authors I would like to acknowledge all who made the publication possible, in particular to Vedran Kordic who coordinated this huge project. Many thanks also for those who helped in the manuscripts preparation making useful suggestions and finding errors.

November 2008

Editor

Witold Bednorz
Warsaw,
Poland,

Contents

1

A Greedy Algorithm with Forward-Looking Strategy

Mao Chen
Engineering Research Center for Educational Information Technology,
Huazhong Normal University,
China

1. Introduction

The greedy method is a well-known technique for solving various problems so as to optimize (minimize or maximize) specific objective functions. As pointed by Dechter *et al* [1], greedy method is a controlled search strategy that selects the next state to achieve the largest possible improvement in the value of some measure which may or may not be the objective function. In recent years, many modern algorithms or heuristics have been introduced in the literature, and many types of improved greedy algorithms have been proposed. In fact, the core of many Meta-heuristic such as simulated annealing and genetic algorithms are based on greedy strategy.

"The one with maximum benefit from multiple choices is selected" is the basic idea of greedy method. A greedy method arrives at a solution by making a sequence of choices, each of which simply looks the best at the moment. We refer to the resulting algorithm by this principle the basic greedy (BG) algorithm, the details of which can be described as follow:

```
Procedure BG (partial solution S, sub-problem P)

Begin
    generate all candidate choices as list L for current sub-problem P;
    while (L is not empty OR other finish condition is not met)
        compute the fitness value of each choice in L;
        modify S and P by taking the choice with highest fitness value;
        update L according to S and P;
    end while;
    return the quality of the resulting complete solution;
End.
```

For an optimization problem, what remains is called a sub-problem after making one or several steps of greedy choice. For problem or sub-problem P, let S be the partial solution, and L be the list of candidate choices at the current moment.

To order or prioritize the choices, some evaluation criteria are used to express the fitness value. According to the BG algorithm, the candidate choice with the highest fitness value is selected, and the partial solution is updated accordingly. This procedure repeated step by step until a resulting complete solution is obtained.

The representation of the BG algorithm can be illustrated by a search tree as shown in Fig.1. Each node in the search tree corresponds to a partial solution, and a line between two nodes represents the decision to add a candidate choice to the existing partial solution. Consequently, leaf nodes at the end of tree correspond to complete solutions.
In Fig.1, the black circle at level 1 denotes an initial partial solution. At level 2, there are four candidate choices for current partial solution, which denotes by four nodes. In order to select the best node, promise of each node should be determined. After some evaluation function has been employed, the second node with highest benefit (the circle in gray at level 2) is selected. Then, the partial solution and sub-problem are updated accordingly.

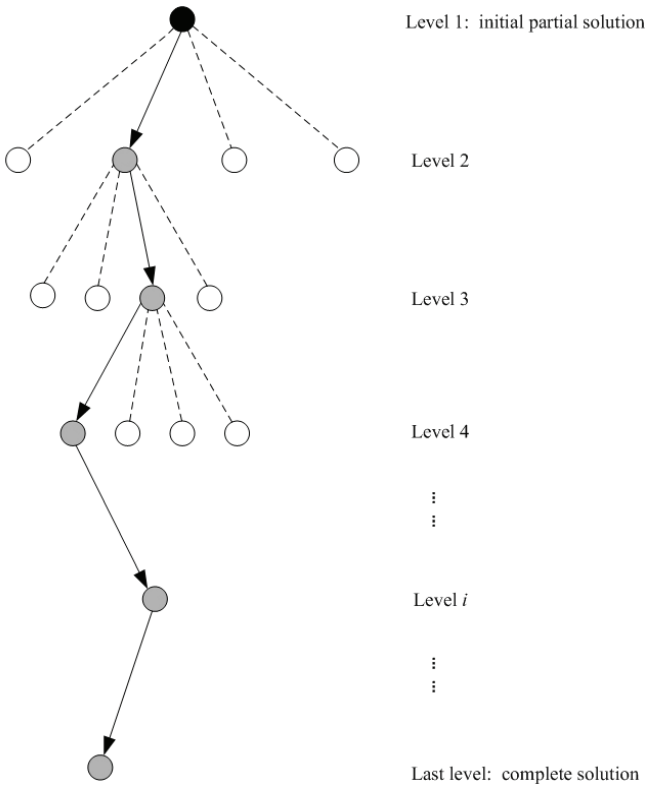


Fig. 1. Representation of basic greedy algorithm

Two important features of greedy method make it so popular are simple implementation and efficiency. Simple as it is, BG algorithm is highly efficient and sometimes it can produce an optimal solution for some optimization problem. For example, for problems such as activity-selection problem, fractional knapsack problem and minimum spanning trees problem, BG algorithm can obtain optimal solution by making a series of greedy choice. For these problems that the BG algorithm can obtain optimal solution, there is something in common: the optimal solution to the problem contains within it optimal solutions to sub-problems.
However, for other optimization problems that do not exhibit such property, the BG algorithm will not lead to optimal solution. Especially for the combinatorial optimization problems or NP-hard problem, the solution by BG algorithm is far away from satisfactory.

In BG algorithm, we make whatever choice seems best at the moment and then turn to solve the sub-problem arising after the choice is made. That is to say, the benefit is only locally evaluated. Consequently, even though we select the best at each step, we still missed the optimal solution. Just liking playing chess, a player who is focused entirely on immediate advantage is easy to be defeated, the player who can think several step ahead will win with more opportunity.

In this chapter, a novel greedy algorithm is introduced in detail, which is of some degree of forward-looking. In this algorithm, all the choices at the moment are evaluated more globally before the best one is selected. The greedy idea and enumeration strategy are both reflected in this algorithm, and we can adjust the enumeration degree so we can balance the efficiency and speed of algorithm.

2. Greedy Algorithm with forward-looking search strategy

To evaluate the benefit of a candidate choice more globally, an improved greedy algorithm with forward-looking search strategy (FG algorithm) was proposed by Huang *et al* [2], which was first proposed for tackling packing problem. It is a kind of growth algorithm and it is efficient for problem that can be divided into a series of sub-problems.

In FG algorithm, the promise of a candidate choice is evaluated not only by the current circumstance, but more globally by considering the quality of the complete solution that can be obtained from the partial solution represented by the node. The idea of FG algorithm can be illustrated by Fig.2:

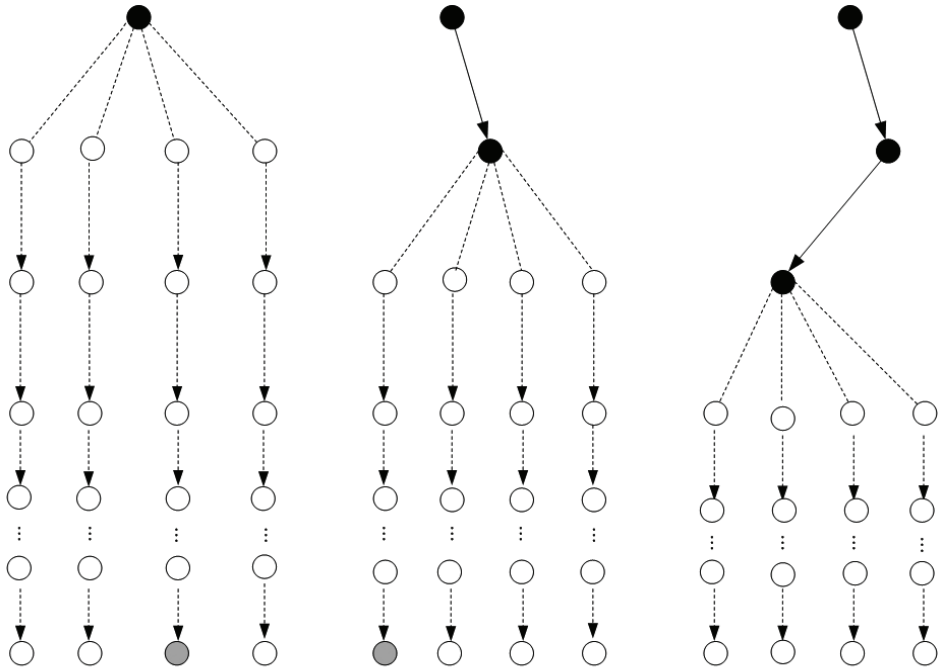


Fig. 2. Representation of greedy algorithm with forward-looking strategy

As shown in Fig.2 (a), there are four nodes at level 2 for the initial partial solution. We do not evaluate the promise of each node at once at the moment. Conversely, we tentatively update the initial partial solution by take the choices at level 2 respectively. For each node at level 2 (i.e., each partial solution at level 2), its benefit is evaluated by the quality of the complete solution resulted from it according to BG algorithm. From the complete solution with maximum quality, we backtrack it to the partial solution and definitely take this step. In other words, the node that corresponds to the complete solution with maximum quality (the gray circle in Fig.2 (a)) is selected as the partial solution. Then the search progresses to level 3. Level by level, this process is repeated until a complete solution is obtained.
After testing the global benefit of each node at current level, the one with great prospect will be selected. This idea can be referred as forward-looking, or backtracking. More formally, the procedure above can be described as follows:

Procedure FG (problem *P*)

```
Begin
    generate the initial partial solution S, and update P to a sub-problem;
    generate all current candidate choice as a list L;
    while (L is not empty AND finish condition is not met)
        max ⇐ 0
        for each choice c in L
            compute the global benefit: GloableBenefit (c, S, P);
            update max with the benefit;
        end for;
        modify S by selecting the choice that the global benefit equal to max;
        update P and L;
    end while;
End.
```

As shown in the above algorithm, in order to more globally evaluate the benefit of a choice and to overcome the limit of BG algorithm, we compute the benefit of a choice using BG itself in the procedure *GlobalBenefit* to obtain the so-called FG algorithm.
Similarly to BG algorithm, we start from the initial partial solution and repeat the above procedure until a complete solution is reached. Note that if there are several complete solutions with the same maximum benefit, we will select the first one to break the tie.
The global benefit of each candidate choice is described as:

Procedure *GlobalBenefit* (choice *c*, partial solution *S*, sub-problem *P*)

```
Begin
    let S′and P′ be copies of S and P;
    modify S′and P′ by taking the choice c;
    return BG(S, P);
End.
```

Given a copy *S′* of the partial solution and a copy *P′*of sub-problem, then we update *S′*by taking the choice c. For the resulted partial solution and sub-problem, we use BG algorithm to obtain the quality of the complete solution.
It should be noted that Procedure FG only gives one initial partial solution. For some problems, there may be several choices for the initial partial solution. Similarly, the

Procedure *globalBenefit*() is implemented for the initial partial solutions respectively, and the one with maximum benefit should be selected.

3. Improved version of FG algorithm

3.1 Filtering mechanism

For some problems, the number of nodes is rather large at each level of search. Therefore, a filtering mechanism is proposed to reduce the computational burden. During filtering some nodes will not be given chance to be evaluated globally and be discarded permanently based on their local evaluation value. Only the remaining nodes are subject to global evaluation.

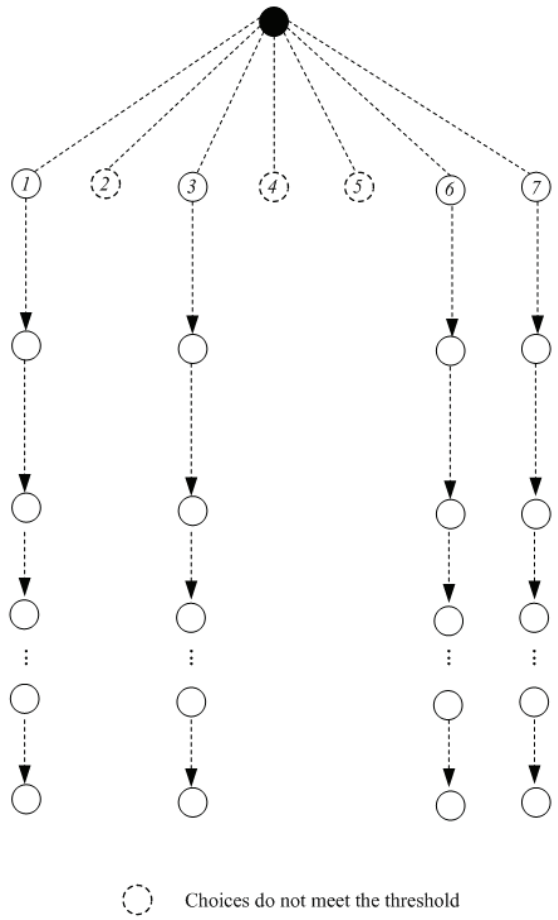


Fig. 3. Representation of filtering mechanism

As shown in Fig.3, there are 7 nodes at level 2. Firstly, the benefit of each node is locally evaluated. Then, only the promising nodes whose local benefit is larger than a given threshold parameter τ will be globally evaluated. The FG algorithm can be modified as FGFM algorithm:

```
Procedure FGFM (problem P)
Begin
    generate the initial partial solution S, update P to a sub-problem;
    generate all current candidate choice as a list L;
    while (L is not empty AND finish condition is not met)
        max ⇐ 0
        for each choice c in L
            if (local benefit > parameter τ )
                compute the global benefit: GloableBenefit (c, S, P);
                update max with global benefit;
            end if;
        end for;
        modify S by selecting the choice that the global benefit equal to max;
        update P and L;
    end while;
End.
```

Obviously, the threshold parameter τ is used to control the trade-off between the quality of the result and the computational time. If τ is set to be large enough, algorithm FGFM turns to be a BG algorithm; If τ is set to be small enough, algorithm FGFM turns to be a FG algorithm.

3.2 Multiple level enumerations

In the FG algorithm, the benefit of a node is globally evaluated by the quality of corresponding complete solution, which is resulted from the node level by level according to the BG algorithm. In order to further improve the quality of the solution, the forward-looking strategy can be applied to several levels.

This multi-level enumeration can be illustrated by Fig.4. For the initial partial solution, there are three candidate choices at level 2. From each node at level 2, there are several branches at level 3. Then we use procedure *GlobalBenefit* () to evaluate the global benefit of each nodes at level 3. That is to say, the three nodes at level 2 have several global benefits. We will choose the highest one as its global benefit. Afterwards, the one with the maximum global benefit from the three nodes at level 2 are selected as the partial solution.

If the number of enumeration levels is equal to (last level number - current level number-1) for each node, the search tree will become a complete enumeration tree, the corresponding solution of which will surely be optimal solution. However, the computational time complexity is unacceptable. Usually, the number of enumeration levels ranges from 1 to 4.

Obviously, the filtering mechanism and multi-level enumeration strategy are the means to control the trade-off between solution quality and runtime effort.

4. Applications

FG algorithm has been successfully applied to job shop scheduling problem [3], circle packing problem [2, 4] and rectangular packing problem [5]. In this section, the two-dimensional (2D) rectangle packing problem and its corresponding bounded enumeration algorithm is presented.

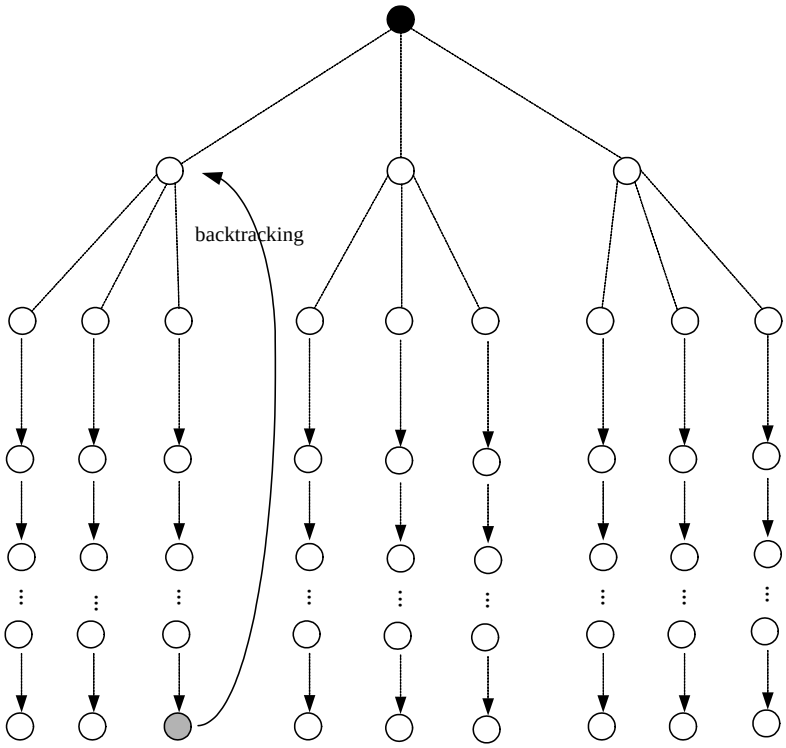


Fig. 4. The multi-level enumeration strategy

4.1 Problem definition

The 2D rectangular packing problem has been widely studied in recent decades, as it has numerous applications in the cutting and packing industry, e.g. wood, glass and cloth industries, newspapers paging, VLSI floor planning and so on, with different applications incorporating different constraints and objectives.

We consider the following rectangular packing problem: given a rectangular empty container with fixed width and infinite height and a set of rectangles with various sizes, the rectangle packing problem is to pack each rectangle into the container such that no two rectangles overlap and the used height of the container is minimized. From this optimization problem, an associated decision problem can be formally stated as follows:

Given a rectangular board with given width W and given height H, and n rectangles with length li and width wi, $1 \le i \le n$, take the origin of the two-dimensional Cartesian coordinate system at the bottom-left corner of the container (see Fig.5). The aim of this problem is to determine if there exist a solution composed of n sets of quadruples $\{x_{11}, y_{11}, x_{12}, y_{12}\}$,…, $\{x_{n1}, y_{n1}, x_{n2}, y_{n2}\}$, where (x_{i1}, y_{i1}) denotes the bottom-left corner coordinates of rectangle i, and (x_{i2}, y_{i2}) denotes the top-right corner coordinates of rectangle i. For all $1 \le i \le n$, the coordinates of rectangle i satisfy the following conditions:

1. $x_{i2}-x_{i1} = l_i \wedge y_{i2}-y_{i1} = w_i$ or $x_{i2}-x_{i1} = w_i \wedge y_{i2}-y_{i1} = l_i$;

2. For all $1 \le i, j \le n$, $j \ne i$, rectangle i and j cannot overlap, i.e., one of the following condition should be met: $x_{i1} \ge x_{j2}$ or $x_j1 \ge x_i2$ or $y_{i1} \ge y_{j2}$ or $y_j1 \ge y_i2$;
3. $0 \le x_{i1}, x_i2 \le W$ and $0 \le y_{i1}, y_{i2} \le H$.

In our packing process, each rectangle is free to rotate and its orientation θ can be 0 (for "not rotated") or 1 (for "rotated by $\pi/2$"). It is noted that the orthogonal rectangular packing problems denote that the packing process has to ensure the edges of each rectangle are parallel to the x- and y-axis, respectively.

Obviously, if we can find an efficient algorithm to solve this decision problem, we can then solve the original optimization problem by using some search strategies. For example, we first apply dichotomous search to get rapidly a "good enough" upper bound for the height, then from this upper bound we gradually reduce it until the algorithm no longer finds a successful solution. The final upper bound is then taken as the minimal height of the container obtained by the algorithm. In the following discussion, we will only concentrate on the decision problem of fixed container.

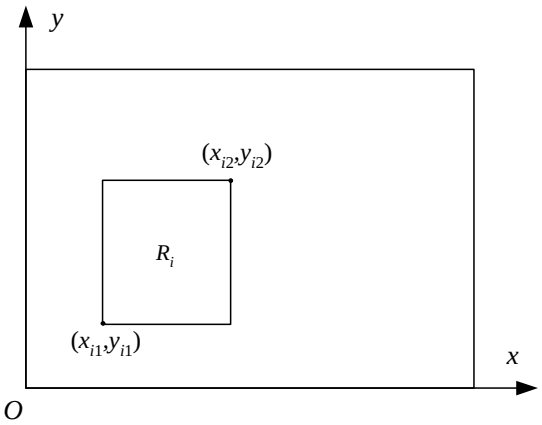


Fig. 5. Cartesian coordinate system

4.2 Preliminary

Definition *Configuration*. A configuration C is a pattern (layout) where m ($0 \le m < n$) rectangles have been already packed inside the container without overlap, and $n-m$ rectangles remain to be packed into the container.

A configuration is said to be successful if $m=n$, i.e., all the rectangles have been placed inside the container without overlapping. A configuration is said to be failure if $m<n$ and none of the rectangles outside the container can be packed into the container without overlapping. A configuration is said final if it is either a successful configuration or a failure configuration.

Definition *Candidate corner-occupying action*. Given a configuration with m rectangles packed, there may be many empty corners formed by the previously packed rectangles and the four sides of the container. Let rectangle i be the current rectangle to be packed, a candidate corner-occupying action (CCOA) is the placement of rectangle i at an empty corner in the container so that rectangle i touches the two items forming the corner and does

not overlap other previously packed rectangles (an item may be a rectangle or one of the four sides of the container). Note that the two items are not necessarily touching each other. Obviously, the rectangle to be packed has two possible orientation choices at each empty corner, that is, the rectangle can be placed with its longer side laid horizontally or vertically. A CCOA can be represented by a quadri-tuple (i, x, y, θ), where (x, y) is the coordinate of the bottom-left corner of the suggested location of rectangle i and θ is the corresponding orientation.

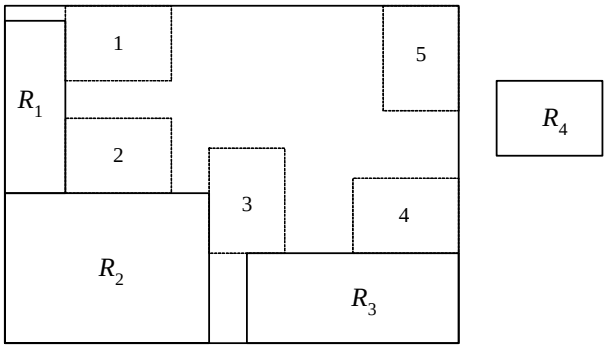


Fig. 6. Candidate corner-occupying action for rectangle R_4

Under current configuration, there may be several candidate packing positions for the current rectangle to be packed. At the configuration in Fig.6, three rectangles R_1, R_2 and R_3 are already placed in the container. There are totally 5 empty corners to pack rectangle R_4, and R_4 can be packed at any one of them with two possible orientations. As a result, there are 10 CCOAs for R_4.

In order to prioritize the candidate packing choices, we need a concept that expresses the fitness value of a CCOA. Here, we introduce the quantified measure λ, called degree to evaluate the fitness value of a CCOA. Before presenting the definition of degree, we first introduce the definition of minimal distance between rectangles as follows.

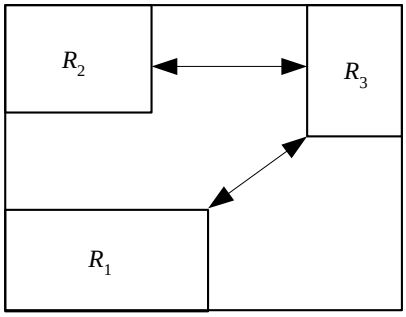


Fig. 7. Illustration of distance

Definition *Minimal distance between rectangles*. Let i and j be two rectangles already placed in the container, and (x_i, y_i), (x_j, y_j) are the coordinates of arbitrary point on rectangle i and j, respectively. The minimal distance d_{ij} between i and j is:

$$d_{ij} = \min\{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\}$$

In Fig.7, R_3 is packed on the position occupying the corner formed by the upper side and the right side of the container. As shown in Fig.7, the minimal distance between R_3 and R_1, and the minimal distance between R_3 and R_2 are illustrated, respectively.

Definition *Degree of CCOA*. Let M be the set of rectangles already placed in the container. Rectangle i is the current rectangle to be packed, (i, x, y, θ) is one of the CCOAs for rectangle i. If corner-occupying action (i, x, y, θ) places rectangle i at a corner formed by two items (rectangle or side of the container) u and v, the degree λ of the corner-occupying action (i, x, y, θ) is defined as:

$$\lambda = 1 - d_{\min}/(\frac{w_i + l_i}{2})$$

where w_i and l_i are the width and the length of rectangle i, and $d_{\min}$ is the minimal distance from rectangle i to other rectangles in M and sides of the container (excluding u and v), that is,

$$d_{\min} = \min\{d_{ij} \mid j \in M \bigcup \{s_1, s_2, s_3, s_4\}, j \neq u, v\}$$

where s_1, s_2, s_3 and s_4 are the four sides of the container.

It is clear that if a corner-occupying action place rectangle i at a position very close to the previously packed rectangles, the corresponding degree will be very high. Note that, if rectangle i can be packed by a CCOA at a corner in the container and touches more than two items, then $d_{\min}=0$ and $\lambda=1$; otherwise $\lambda<1$. The degree of a corner-occupying action describes how the placed rectangle is close to the already existing pattern. Thus, we use it as the benefit of a packing step.

Intuitively, since one should place a rectangle as close as possible to the already existing pattern, it seems quite natural that the CCOA with the highest degree will be selected first to pack the rectangle into the container. We call this principle the highest degree first (HDF) rule. It is just the simple application of BG algorithm.

4.3 The basic algorithm: A_0

Based on the HDF rule and BG algorithm, A_0 is described as follows:

Procedure A_0 (C, L)

Begin

```
  while (L is not empty)
     for each CCOA in L
           calculate the degree;
      end for;
      select the CCOA (i, x, y, θ) with the highest degree;
      modify C by placing rectangle i at (x, y) with orientation θ;
      modify L according to the new configuration C;
  end while;
  return C;
```

End.

At each iteration, a set of CCOAs for each of the unpacked rectangles is generated under current configuration C. Then the CCOAs for all the unpacked rectangles outside the container are gathered as a list L. A_0 calculates the degree of each CCOA in L and selects the CCOA (i, x, y, θ) with the highest degree λ, and place rectangle i at (x, y) with orientation θ. After placing rectangle i, the list L is modified as follows:

1. Remove all the CCOAs involving rectangle i;
2. Remove all infeasible CCOAs. A CCOA becomes infeasible because the involved rectangle would overlap rectangle i if it was placed;
3. Re-calculate the degree λ of the remaining CCOAs;
4. If a rectangle outside the container can be placed inside the container without overlap so that it touches rectangle i and a rectangle inside the container or the side of the container, create a new CCOA and put it into L, and compute the degree λ of the new CCOA.

If none of the rectangles outside the container can be packed into the container without overlap (L is empty) at certain iteration, A_0 stops with failure (returns a failure configuration). If all rectangles are packed in the container without overlap, A_0 stops with success (returns a successful configuration).

It should be pointed out that if there are several CCOAs with the same highest degree, we will select one that packs the corresponding rectangle closest to the bottom left corner of the container.

A_0 is a fast algorithm. However, given a configuration, A_0 only considers the relation between the rectangles already inside the container and the rectangle to be packed. It doesn't examine the relation between the rectangles outside the container. In order to more globally evaluate the benefit of a CCOA and to overcome the limit of A_0, we compute the benefit of a CCOA using A_0 itself in the procedure $BenefitA_1$ to obtain our main packing algorithm called A_1.

4.4 The greedy algorithm with forward-looking strategy: A_1

Based on current configuration C, CCOAs for all unpacked rectangles are gathered as a list L. For each CCOA (i, x, y, θ) in L, the procedure $BenefitA_1$ is designed to evaluate its benefit more globally.

Procedure $BenefitA_1$ (i, x, y, θ, C, L)

Begin
  let C' and L' be copies of C and L;
  modify C' by placing rectangle i at (x, y) with orientation θ, and modify L';
  $C' = A_0(C', L')$;
  if (C' is a successful configuration)
    Return C';
  else
    Return density (C');
  end if-else
End.

Given a copy C' of the current configuration C and a CCOA (i, x, y, θ) in L, $BenefitA_1$ begins by packing rectangle i in the container at (x, y) with orientation θ and call A_0 to reach a final configuration. If A_0 stops with success then $BenefitA_1$ returns a successful configuration,

otherwise $BenefitA_1$ returns the density (the ratio of the total area of the rectangles inside the container to the area of the container) of a failure configuration as the benefit of the CCOA (i, x, y, θ). In this manner, $BenefitA_1$ evaluates all existing CCOAs in L.
Now, using the procedure $BenefitA_1$, the benefit of a CCOA is measured by the density of a failure configuration. The main algorithm A_1 is presented as follow:

Procedure A_1 ( )

Begin
generate the initial configuration C;
generate the initial CCOA list L;
while (L is not empty)
maximum benefit $\leftarrow 0$
for each CCOA (i, x, y, θ) in L
$d = BenefitA_1 (i, x, y, \theta, C, L)$;
if (d is a successful configuration)
stop with success;
else
update the maximum benefit with d;
end if-else;
end for;
select the CCOA $(i^*, x^*, y^*, \theta^*)$ with the maximum benefit;
modify C by placing rectangle i^* at (x^*, y^*) with orientation θ^*;
modify L according to the new configuration C;
end while;
stop with failure
End.

Similarly, A_1 selects the CCOA with the maximum benefit and packs the corresponding rectangle into the container by this CCOA at each iteration. If there are several CCOAs with the maximum benefit, we select one that packs the corresponding rectangle closest to the bottom left corner of the container.

4.5 Computational complexity

We analysis the complexity of A_1 in the worst case, that is, when it cannot find successful configuration, and discuss the real computational cost to find a successful configuration.
A_0 is clearly polynomial. Since every pair of rectangles or sides in the container can give a possible CCOA for a rectangle outside the container, the length of L is bounded by $O(m^2(n-m))$, if m rectangles are already placed in the container. For each CCOA in L, $d_{\min}$ is calculated using the $d_{\min}$ in the last iteration in $O(1)$ time. The creation of new CCOAs and the calculus of their degree is also bounded by $O(m^2(n-m))$ since there are at most $O(m(n-m))$ new CCOAs (a rectangle might form a corner position with each rectangle in the container and each side of the container). So the time complexity of A_0 is bounded by $O(n^4)$.
A_1 uses a powerful search strategy in which the consequence of each CCOA is evaluated by applying $BenefitA_1$ in full, which allows us to examine the relation between all rectangles (inside and outside the container). Note that the benefit of a CCOA is measured by the

density of a final configuration, which means that we should apply $BenefitA_1$ though to the end each time. At every iteration of A_1, $BenefitA_1$ uses a $O(n^4)$ procedure to evaluate all $O(m^2(n-m))$ CCOAs, therefore, the complexity of A_1 is bounded by $O(n^8)$.

It should be pointed out that the above upper bounds of the time complexity of A_0 and A_1 are just rough estimations, because most corner positions are infeasible to place any rectangle outside the container, and the real number of CCOAs in a configuration is thus much smaller than the theoretical upper bound $O(m^2(n-m))$.

The real computational cost of A_0 and A_1 to find a successful configuration is much smaller than the above upper bound. When a successful configuration is found, $BenefitA_1$ does not continue to try other CCOAs, nor A_1 to exhaust the search space. In fact, every call to A_0 in $BenefitA_1$ may lead to a successful configuration and then stops the execution at once. Then, the real computational cost of A_1 essentially depends on the real number of CCOAs in a configuration and the distribution of successful configurations. If the container height is not close to the optimal one, there exists many successful configurations, and A_1 can quickly find such one. However, if the container height is very close to the optimal one, few successful configurations exist in the search space, and then A_1 may need to spend more time to find a successful configuration in this case.

4.6 Computational results

The set of tests is done using the Hopper and Turton instances [6]. There are 21 different sized test instances ranging from 16 to 197 items, and the optimal packing solutions of these test instances are all known (see Table 1). We implemented A_1 in C on a 2.4 GHz PC with 512 MB memory. As shown in Table 1, A_1 generates optimal solutions for 8 of the 21 instances; for the remaining 13 instances, the optimum is missed in each case by a single length unit.

To evaluate the performance of the algorithm, we compare A_1 with two best meta-heuristic (SA+BLF) in [6], HR [7], LFFT [8] and SPGAL [9]. The quality of a solution is measured by the percentage gap, i.e., the relative distance of the solution lU to the optimum length $lOpt$. The gap is computed as $(lU - lOpt)/lOpt$. The indicated gaps for the seven classes are averaged over the respective three instances. As shown in Table 2, the gaps of A_1 ranges form 0.0% to 1.64% with the average gap 0.72, whereas the average gap of the two meta-heuristics and HR are 4.6%, 4.0% and 3.97%, respectively. Obviously, A_1 considerably outperforms these algorithms in terms of packing density. Compared with two other methods, the average gap of A_1 is lower than that of LFFT, however, the average gap of A_1 is slightly higher than that of SPGAL.

As shown in Table 2, with the increasing of the number of rectangles, the running time of the two meta-heuristics and LFFT increases rather fast. HR is a fast algorithm, whose time complexity is only $O(n3)$ [7]. Unfortunately, the running time of each instance for SPGAL is not reported in the literature. The mean time of all test instances for SPGAL is 139 seconds, which is acceptable in practical applications. It can be seen that A_1 is also a fast algorithm. Even for the problem instances of larger size, A_1 can yield solutions of high density within short running time.

It has shown from Table 2 that the running time of A_1 does not consistently accord with its theoretical time complexity. For example, the average time of C3 is 1.71 seconds, while the average time of C4 and C5 are both within 0.5 seconds. As pointed out in the time complexity analysis, once A_0 finds a successful solution, the calculation of A_1 will terminate. Actually, the average time complexity is much smaller than the theoretical upper bound.

Test instance Class / subclass		No. of pieces	Object dimensions	Optimal height	Minimum Height by A_1	% of unpacked area	CPU time (s)
	C11	16	20×20	20	20	0.00	0.37
C1	C12	17	20×20	20	20	0.00	0.50
	C13	16	20×20	20	20	0.00	0.23
	C21	25	15×40	15	15	0.00	0.59
C2	C22	25	15×40	15	15	0.00	0.44
	C23	25	15×40	15	15	0.00	0.79
	C31	28	30×60	30	30	0.00	3.67
C3	C32	29	30×60	30	30	0.00	1.44
	C33	28	30×60	30	31	3.23	0.03
	C41	49	60×60	60	61	1.64	0.22
C4	C42	49	60×60	60	61	1.64	0.13
	C43	49	60×60	60	61	1.64	0.11
	C51	73	90×60	90	91	1.09	0.34
C5	C52	73	90×60	90	91	1.09	0.33
	C53	73	90×60	90	91	1.09	0.52
	C61	97	120×80	120	121	0.83	8.73
C6	C62	97	120×80	120	121	0.83	0.73
	C63	97	120×80	120	121	0.83	2.49
	C71	196	240×160	240	241	0.41	51.73
C7	C72	197	240×160	240	241	0.41	37.53
	C73	196	240×160	240	241	0.41	45.81

Table 1. Computational results of our algorithm for the test instances from Hopper and Turton instances

Class	SA+BLF[1]		HR[2]		LFFT[3]		SPGAL[4]		A_1[5]	
	Gap	Time	Gap	Time	Gap	Time	Gap	Time (s)	Gap	Time
C1	4.0	42	8.33	0	0.0	1	1.7	−	0.00	0.37
C2	6.0	144	4.45	0	0.0	1	0.0	−	0.00	0.61
C3	5.0	240	6.67	0.03	1.0	2	2.2	−	1.07	1.71
C4	3.0	1980	2.22	0.14	2.0	15	0.0	−	1.64	0.15
C5	3.0	6900	1.85	0.69	1.0	31	0.0	−	1.09	0.40
C6	3.0	22920	2.5	2.21	1.0	92	0.3	−	0.83	3.98
C7	4.0	250800	1.8	36.07	1.0	2150	0.3	−	0.41	45.02
Average gap (%)	4.0		3.97		0.86		0.64		0.72	

Table 2. The gaps (%) and the running time (seconds) for meta-heuristics, HR, LFFT, SPGAL and A_1

[1] PC with a Pentium Pro 200MHz processor and 65MB memory [11].
[2] Dell GX260 with a 2.4 GHz CPU [15].
[3] PC with a Pentium 4 1.8GHz processor and 256 MB memory [14].
[4] The machine is 2GHz Pentium [16].
[5] 2.4 GHz PC with 512 MB memory.

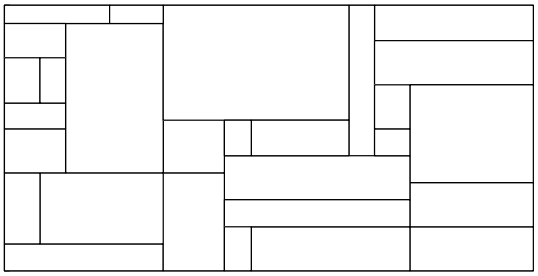


Fig. 8. Packing result of C31

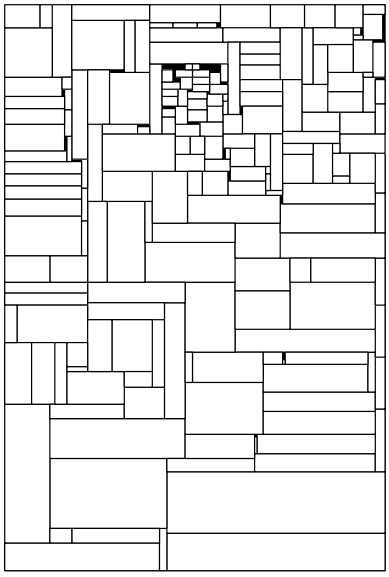


Fig. 9. Packing result of C73

In addition, we give the packing results on test instances C31 and C73 for A_1 in Fig.8~Fig.9. Here, the packing result of C31 is of optimal height, and the height C73 are only one length unit higher than the optimal height

5. Conclusion

The algorithm introduced in this chapter is a growth algorithm. Growth algorithm is a feasible approach for combinatorial optimization problems, which can be solved step by step. After one step is taken, the original problem becomes a sub-problem. In this way, the problem can be solved recursively. For the growth algorithm, the difficulty lies in that for a sub-problem, there are several candidate choices for current step. Then, how to select the most promising one is the core of growth algorithm.

By basic greedy algorithm, we use some concept to compute the fitness value of candidate choice, then, we select one with highest value. The value or fitness is described by quantified measure. The evaluation criterion can be local or global. In this chapter, a novel greedy

algorithm with forward-looking strategy is introduced, the core of which can more globally evaluate a partial solution.

For different problems, this algorithm can be modified accordingly. This chapter gave two new versions. One is of filtering mechanism, i.e., only part of the candidate choices with higher local benefit will be globally evaluated. A threshold parameter is set to allow the trade-off between solution quality and runtime effort to be controlled. The higher the threshold parameter, the faster the search will be finished., and the lower threshold parameter, the more high-quality solution may be expected. The other version of the greedy algorithm is multi-level enumerations, that is, a choice is more globally evaluated.

This greedy algorithm has been successfully used to solve rectangle packing problem, circle packing problem and job-shop problem. Similarly, it can also be applied to other optimization problems.

6. Reference

[1] A. Dechter, R. Dechter. On the greedy solution of ordering problems. ORSA Journal on Computing, 1989, 1: 181-189

[2] W.Q. Huang, Y Li, S Gerard, *et al*. A "learning from human" heuristic for solving unequal circle packing problem. Proceedings of the First International Workshop on Heuristics, Beijing, China, 2002, 39-45.

[3] Z. Huang, W.Q. Huang. A heuristic algorithm for job shop scheduling. Computer Engineering & Appliances (in Chinese), 2004, 26: 25-27

[4] W.Q. Huang, Y. Li, H. Akeb, *et al*. Greedy algorithms for packing unequal circles into a rectangular container. Journal of the Operational Research Society, 2005, 56: 539-548

[5] M. Chen, W.Q. Huang. A two-level search algorithm for 2D rectangular packing problem. Computers & Industrial Engineering, 2007, 53: 123-136

[6] E. Hopper, B.Turton, An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. European J. Oper. Res, 128 (2001): 34-57

[7] D.F. Zhang, Y. Kang, A.S. Deng. A new heuristic recursive algorithm for the strip rectangular packing problem. Computers & Operational Research. 33 (2006): 2209-2217

[8] Y.L. Wu, C.K. Chan. On improved least flexibility first heuristics superior for packing and stock cutting problems. Proceedings for Stochastic Algorithms: Foundations and Applications, SAGA 2005, Moscow, 2005, 70-81

[9] A. Bortfeldt. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. European Journal of Operational Research. 172 (2006): 814-837

2

A Greedy Scheme for Designing Delay Monitoring Systems of IP Networks

Yigal Bejerano[1] and Rajeev Rastogi[2]
[1]*Bell Laboratories, Alcatel-Lucent,*
[2]*Yahoo-Inc,*
[1]*USA*
[2]*India*

1. Introduction

The demand for sophisticated tools for monitoring network utilization and performance has been growing rapidly as Internet Service Providers (ISPs) offer their customers more services that require quality of service (QoS) guarantees and as ISP networks become increasingly complex. Tools for monitoring link delays and faults in an IP network are critical for numerous important network management tasks, including providing QoS guarantees to end applications (e.g., voice over IP), traffic engineering, ensuring service level agreement (SLA) compliance, fault and congestion detection and performance debugging. Consequently, there has been a recent flurry of both research and industrial activity in the area of developing novel tools and infrastructures for measuring network parameters.

Existing network monitoring tools can be divided into two categories. *Node-oriented* tools collect monitoring information from network devices (routers, switches and hosts) using SNMP/RMON probes [1] or the Cisco NetFlow tool [2]. These are useful for collecting statistical and billing information, and for measuring the performance of individual network devices (e.g., link bandwidth usage). However, in addition to the need for monitoring agents to be installed at every device, these tools cannot monitor network parameters that involve several components, like link or end-to-end path latency. The second category contains *path-oriented* tools for connectivity and latency measurement like `ping`, `traceroute` [3], `skitter` [4] and tools for bandwidth measurement such as `pathchar` [5], `Bing` [6], `Cprobe` [7], `Nettimer` [8] and `pathrate` [9]. As an example, `skitter` sends a sequence of probe messages to a set of destinations and measures the latency of a link as the difference in the round-trip times of the two probes to the endpoints of the link. A benefit of path-oriented tools is that they do not require special monitoring agents to be run at each node. However, a node with such a path-oriented monitoring tool, termed a *monitoring station*, is able to measure latencies and monitor faults for only a limited set of links in the node's routing tree, e.g., its *shortest path tree* (SPT). Thus, monitoring stations need to be deployed at a few strategic points in the ISP or Enterprise IP network so as to maximize network coverage, while minimizing hardware and software infrastructure cost, as well as maintenance cost for the stations. Consequently, any monitoring system needs to satisfy two basic requirements.

1. *Coverage* - The system should accurately monitor all the links and paths in the network.
2. *Efficiency* - The systems should minimize the overhead imposed by monitoring on the underlying production network.

The chapter proposes an efficient *two-phased* approach for fully and efficiently monitoring the latencies of links and paths using path-oriented tools. Our scheme ensures complete coverage of measurements by selecting monitoring stations such that each network link is in the routing trees of some monitoring station. It also reduces the monitoring overhead which consists of two costs: the infrastructure and maintenance cost associated with the monitoring stations, as well as the additional network traffic due to probe packets. Minimizing the latter is especially important when information is collected frequently in order to continuously monitor the state and evolution of the network. In the *first phase*, the scheme addresses the *station selection problem*. This phase seeks for the locations of a minimal set of monitoring stations that are capable to perform all the required monitoring tasks, such as monitoring the delay of all the network links. Subsequently, in the *second phase*, the scheme deals with the *probe assignment problem*, which computes a minimal set of probe messages transmitted by each station for satisfying the monitoring requirements.

Although, the chapter focuses primarily on delay monitoring, the presented approach is more generally applicable and can also be used for other management tasks. We consider two variants of monitoring systems. A *link monitoring* (LM) system that guarantees that very link is monitored by a monitoring station. Such system is useful for delay monitoring, bottleneck links detection and fault isolation, as demonstrated in [10]. A *path monitoring* (PM) system that ensures the coverage of every routing path between any pair of nodes by a single station, which provides accurate delay monitoring.

For link monitoring we show that the problem of computing the minimum set of stations whose routing trees (e.g, its shortest path trees), cover all network links is NP-hard. Consequently, we map the station selection problem to the set cover problem [11], and we use a polynomial-time greedy algorithm that yields a solution within a logarithmic factor of the optimal one. For the probe assignment problem, we show that computing the optimal probe set for monitoring the latency of all the network links is also NP-hard. To this problem, we devise a polynomial-time greedy algorithm that computes a set of probes whose cost is within an factor of 2 of the optimal solution. Then, we extend our scheme to path monitoring. Initially, we show that even when the number of monitoring stations is small (in our example only two monitoring stations) every pair of adjacent links along a given routing path may be monitored by two different monitoring stations. This raises the need for a path monitoring system in which every path is monitored by a single station. For station selection we devise a set-cover-based greedy heuristic that computes solutions with logarithmic approximation ratio. Then, we propose a greed algorithm for probe assignment and leave the problem of constructing an efficient algorithm with low approximation ratio for future work.

The chapter is organized as follows. It starts with a brief survey of related work in Section 2. Section 3 presents the network model and a description of the network monitoring framework is given in Section 4. Section 5 describes our link monitoring system and Section 6 extends our scheme to path monitoring. Section 7 provides simulation results that demonstrate the efficiency of our scheme for link monitoring and Section 8 concludes the chapter.

2. Related work

The need for low-overhead network monitoring techniques has gained significant attention in the recent years and below we provide the most relevant studies to this chapter. The network proximity service project, SONAR [12], suggests to add a new client/server service that enables hosts to obtain fast estimations of their distance from different locations in the Internet. However, the problem of acquiring the latency information is not addressed. The IDmaps [13] project produces "*latency maps*" of the internet using special measurement servers called *tracers* that continuously probe each other to determine their distance. These times are subsequently used to approximate the latency of arbitrary network paths. Different methods for distributing tracers in the internet are described in [14], one of which is to place them such that the distance of each network node to the closest tracer is minimized. A drawback of the IDMaps approach is that latency measurements may not be accurate. Essentially, due to the small number of paths actually monitored, it is possible for errors to be introduced when round-trip times between tracers are used to approximate arbitrary path latencies. In [15], Breitbart *et al.* propose a monitoring scheme where a single *network operations center* (NOC) performs all the required measurements. In order to monitor links not in its routing tree, the NOC uses the IP source routing option to explicitly route probe packets along these links. The technique of using source routing for determining the probe routes has been used by other proposals as well for both fault detection [16] and delay monitoring [17]. Unfortunately, due to security problems, many routers frequently disable the IP source routing option. Further, routers usually process IP options separately in their CPU, which in addition to adversely impacting their performance, also causes packets to suffer unknown delays. Consequently, approaches that rely on explicitly routed probe packets for delay and fault monitoring may not be feasible in today's ISP and Enterprise environments. Another delay monitoring approach was presented by Shavit *et al.* in [18]. They propose to solve a linear system of equations to compute delays for smaller path segments from a given a set of end-to-end delay measurements for paths in the network.

The problem of station placement for delay monitoring has been addressed by several studies. In [19], Adler *et al.* focus on the problem of determining the minimum cost set of multicast trees that cover links of interest in a network, which is similar to the station selection problem tackled in this chapter. The two-phase scheme of station placement and probe assignment have been proposed in [10]. In this work, Bejerano and Rastogi show a combined approach for minimizing the cost of both the monitoring stations as well as the probe messages. Moreover, they extend their scheme for delay monitoring and fault isolation in the presence of multiple failures. In [20] Breitbart *et al.* consider two variants of the station placement problem assuming that the routing tree of the nodes are their shortest path trees (SPTs). In the first variant, termed A-Problem, the routing trees of a node may be any one of its SPT, while in the second variant, called E-Problem, the routing tree of a node can be selected among all the possible SPTs for minimizing the monitoring overhead. For both variant they have shown that the problems are NP-hard and they provided approximation algorithms. In [21] Nguyen and Thiran developed a technique for locating multiple failures in IP networks using active measurement. They also proposed a two-phased approach, but unlike the work in [10], they optimize first the probe selection and only then they compute the location of a minimal set of monitoring stations that can generate these probes. Moreover, by using techniques from a max-plus algebra theory, they show that the optimal set of probes can be determined in polynomial time. In [22], Suh *et al.*

propose a scheme for cost-effective placement of monitoring stations for passive monitoring of IP flows and controlling their sampling rate. Recently, Cantieni *et al.* [23], reformulate the monitoring placement problem. They assume that every node may be a monitoring station at any given time and then they ask the question which monitors should be activated and what should be their sampling to achieve a given measurement task? To this problem they provide optimal solution.

3. Network model

We model the Service Provider or Enterprise IP network by an undirected graph $G(V,E)$, where the graph nodes, V, denote the network routers and the edges, E, represent the communication links connecting them. The number of nodes and edges is denoted by $|V|$ and $|E|$, respectively. Further, we use $P_{s,t}$ to denote the path traversed by an IP packet from a source node s to a destination node t. In our model, we assume that packets are forwarded using standard IP forwarding, that is, each node relies exclusively on the destination address in the packet to determine the next hop. Thus, for every node $x \in P_{s,t}$, $P_{x,t}$ is included in $P_{s,t}$. In addition, we also assume that $P_{s,t}$ is the routing path in the opposite direction from node t to node s. This, in turn, implies that for every node $x \in P_{s,t}$, $P_{s,x}$ is a prefix of $P_{s,t}$. As a consequence, it follows that for every node $s \in V$, the subgraph obtained by merging all the paths $P_{s,t}$, for every $t \in V$, must have a tree topology. We refer to this tree for node s as the *routing tree* (RT) of node s and denote it by T_s. Note that tree T_s defines the routing paths from node s to all the other nodes in V and vice versa.

Observe that for a Service Provider network consisting of a single OSPF area, the RT T_s of node s is its shortest path tree (SPT). However, for networks consisting of multiple OSPF areas or autonomous systems (that exchange routing information using BGP), packets between nodes may not necessarily follow shortest paths. In practice, the topology of RTs can be calculated by querying the routing tables of nodes. In our solution, the routing tree of node s may be its SPT but this is not an essential requirement.

We associate a positive cost $c_{u,v}$ with sending a message between any pair of nodes $u, v \in V$. For every intermediate node $w \in P_{u,v}$ both $c_{u,w}$ and $c_{v,w}$ are at most $c_{u,v}$ and $c_{u,w} + c_{v,w} \geq c_{u,v}$. Typical examples of this cost model are the fixed cost, where all messages have the same cost, and hop count, where the message cost is the number of hops in its route.

4. Network monitoring framework

In this section, we describe our methodology for complete IP network monitoring using path-oriented tools. Our primary focus is the measurement of round-trip latency of network links and paths. However, our methodology is also applicable for a wide range of monitoring tasks, like fault and bottleneck link detection, as presented in [10]. For monitoring the round-trip delay of a link $e \in E$, a node $s \in V$ such that e belongs to s's RT (that is, $e \in T_s$), must be selected as a monitoring station. Node s sends two probe messages[1] to the end-points of e, which travel almost identical routes except for the link e. On receiving a probe message, the receiver replies immediately by sending a probe reply message to the

[1] The probe messages are implemented by using "ICMP ECHO REQUEST/REPLY" messages similar to `ping`.

monitoring station. Thus, the monitoring station s can estimate the round-trip delay of the link by measuring the difference in the round-trip times of the two probe messages.

From the above description, it follows that a monitoring station can only measure the delays of links in its RT. Consequently, a monitoring system designated for measuring the delays of all network links has to find a set of *monitoring stations* $S \subseteq V$ and a *probe assignment* $A \subset S \times V$. A probe assignment is basically a set of pairs $\{(s, u) \mid s \in S, u \in V\}$ such that each pair (s, u) represents a probe message that is sent from the monitoring station s to node u. The set S and the probe assignment A are required to satisfy two constraints:

1. The *covering monitoring station set* constraint guarantees that all links are covered by the RTs of the nodes in S, *i.e.*, $\bigcup_{s \in S} T_s = E$.
2. The *covering probe assignment* constraint ensures that for every edge $e = (u, v) \in E$, there is a node $s \in S$ such that $e \in T_s$ and $\mathcal{A}$ contains the pairs[2] (s, u) and (s, v). In other words, every link is monitored by at least one monitoring station.

A pair $(S,\mathcal{A})$ that satisfies the above constraints is referred to as a *feasible solution*. In instances where the monitoring stations are selected from a subset $Y \subset V$, we assume that $\bigcup_{s \in Y} T_s = E$ which guarantees the existence of a feasible solution.

The overhead of a monitoring system is composed of two components, the overhead of installing and maintaining the monitoring stations and the communication cost of sending probe messages. In practice, it is preferable to have as few stations as possible since this reduces operational costs, and so we adopt a two-phased approach to optimizing monitoring overheads. In the first phase, we select an optimal set of monitoring stations, while in the second, we compute the optimal probes for the selected stations. Let w_v be the cost of selecting node $v \in V$ as a monitoring station. The *optimal station selection* S is the one that satisfies the covering monitoring station set requirement and minimizes the total cost of all the monitoring stations given be the sum $\sum_{s \in S} w_s$. After selecting the monitoring stations S, the *optimal probe assignment* $\mathcal{A}$ is one that satisfies the covering probe assignment constraint and minimizes the total probing cost defined by the sum $\sum_{(s,v) \in \mathcal{A}} c_{s,v}$. Note that choosing $c_{sv} = 1$ essentially results in an assignment $\mathcal{A}$ with the minimum number of probes, while choosing $c_{s,v}$ to be the minimum number of hops between s and v yields a set of probes that traverse the fewest possible network links.

A final component of our monitoring infrastructure is the *network operations center* (NOC) which is responsible for coordinating the actions of the set of monitoring stations S. The NOC queries the network nodes to determine their RTs, and subsequently uses these to compute a near-optimal set of monitoring stations and a probe assignment for them. In the following two sections, we develop approximation algorithms for the station selection and probe assignment problems. Section 5 considers the problem of monitoring links, while path monitoring is addressed in Section 6. Note that our proposed framework deals only with the aspect of efficient collection of monitoring information. It does not deal with the aspects of analyzing and distributing this information, which are application-dependent.

---

[2] If one of the end points of e is in S, let say $u \in S$, then $\mathcal{A}$ is only required to include the probe (u, v).

5. Link monitoring

We show in this section that for link monitoring both the station selection and probe assignment problems are NP-hard. Then, we present polynomialtime approximation algorithms for solving them. For station selection, we develop a $\ln(|V|)$-approximation algorithm where the lower bound is $1/2 \cdot \ln(|V|)$ and for probe assignment, we present a 2 approximation algorithm.

5.1 An efficient station selection algorithm

The problem addressed here is covering all the graph edges with a small number of RTs, and we consider both the un-weighted and the weighted versions of this problem.

Definition 1 (The Link Monitoring Problem - LM):

Given a graph $G(V,E)$ and a RT, T_v, for every node $v \in V$, find the smallest set $S \subseteq V$ such that $\bigcup_{v \in S} T_v = E$. □

Definition 2 (The Weighted LM Problem - WLM) :

Given a graph $G(V,E)$ with a non-negative weight w_v and a RT T_v for every node $v \in V$, find the set $S \subseteq V$ such that $\bigcup_{v \in S} T_v = E$ and the sum $\Sigma_{v \in S} w_v$ is minimum. □

We show a similarity between the link monitoring problem and the *set cover* (SC) problem, which is a well-known NP-hard problem [24]. An efficient algorithm for solving one of them can be also used to efficiently solve the other. Let us recall the SC problem. Consider an instance $I(Z,\mathcal{Q})$ of the SC problem, where $Z = \{z_1, z_2, \ldots, z_m\}$ is a universe of m elements and $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_n\}$ is a collection of n subsets of Z, (assume that $\bigcup_{Q \in \mathcal{Q}} Q = Z$). The SC problem seeks to find the smallest collection of subsets $\mathcal{S} \subseteq \mathcal{Q}$ such that their union contains all the elements in Z, *i.e.*, $\bigcup_{Q \in \mathcal{S}} Q = Z$. At the weighted version of the CS problem, each one of the subsets $Q \in \mathcal{Q}$ has a cost w_Q and the optimal solution is the *lowest-cost* collection of subsets $\mathcal{S} \subseteq \mathcal{Q}$, such that their union contains all the elements in Z. For SC problem the greedy heuristic [11] is a commonly used approximation algorithm and it achieves a tight approximation ratio of $\ln(k)$, where k is the size of the biggest set $Q \in \mathcal{Q}$. Note that in the worst case $k = m$.

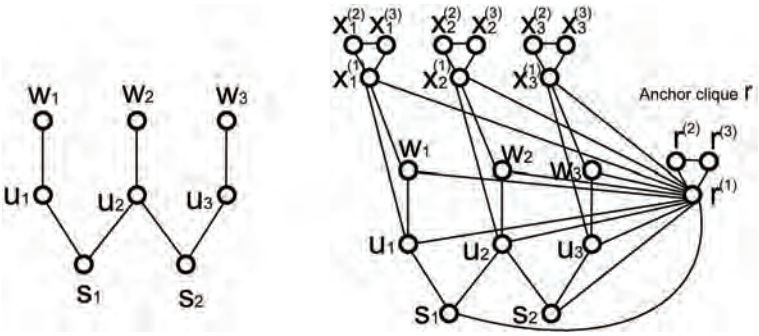


Fig. 1. The graph $G_{\mathcal{R}(I)}(V,E)$ for the given instance of the SC problem.

5.1.1 Hardness of the LM and WLM problems

Theorem 1 *The LM and WLM problems are NP-hard, even when the routing tree (RT) of each node is restricted to be its shortest path tree (SPT).*

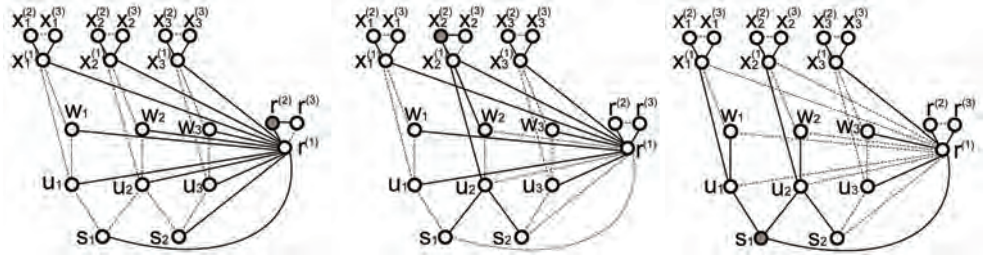


Fig. 2. The RTs of nodes $r^{(2)}$, $x_2^{(2)}$ and s_1.

Proof: We show that the LM problem is NP-hard by presenting a polynomial reduction from the *set cover* problem to the LM problem. From this follows that also the WLM problem is NP-hard. Consider an instance $I(Z,\mathcal{Q})$ of the SC problem. Our reduction $\mathcal{R}(I)$ constructs the graph $G_{\mathcal{R}(I)}(V,E)$ where the RT of each node $v \in V$ is also its shortest path tree. For determining these RTs, each edge is associated with a weight[3], and the graph contains the following nodes and edges. For each element $z_i \in Z$, it contains two connected nodes u_i and w_i. For each set $Q_j \in \mathcal{Q}$, we add a node, labeled by s_j, and the edges (s_j, u_i) for each element $z_i \in Q_j$. In addition, we use an auxiliary structure, termed an *anchor clique* x, which is a clique with three nodes, labeled by $x^{(1)}$, $x^{(2)}$ and $x^{(3)}$, and only node $x^{(1)}$ has additional incident edges. For each element $z_i \in Z$, the graph $G_{\mathcal{R}(I)}$ contains one anchor clique x_i whose attachment point, $x_i^{(1)}$, is connected to the nodes u_i and w_i. The weights of all the edges described above is 1. Finally, the graph $G_{\mathcal{R}(I)}$ contains an additional anchor clique r that is connected to the remaining nodes and anchor cliques of the graph, and the weights of these edges is $1 + \epsilon$. An example of such a graph is depicted in Figure 1 for an instance of the SC problem with 3 elements $\{z_1, z_2, z_3\}$ and two sets $Q_1 = \{z_1, z_2\}$ and $Q_2 = \{z_2, z_3\}$.

We claim that there is a solution of size k to the given SC problem if and only if there is a solution of size $k+m+1$ to the LM instance defined by the graph $G_{\mathcal{R}(I)}(V,E)$. We begin by showing that if there is a solution to the SC problem of size k then there exists a set S of at most $k+m+1$ stations that covers all the edges in $G_{\mathcal{R}(I)}$. Let the solution of the SC problem consist of the sets $Q_{i_1}, \ldots, Q_{i_k}$. The set S of monitoring stations contains the nodes $r^{(2)}$, $x_i^{(2)}$ (for each element $z_i \in Z$) and $s_{i_1}, \ldots, s_{i_k}$. We show that the set S contains $k + m + 1$ nodes that cover all the graph edges. The tree $T_{r^{(2)}}$ covers edges $(r^{(1)}, r^{(2)})$, $(r^{(2)}, r^{(3)})$, all edges $(u_i, r^{(1)})$, $(w_i, r^{(1)})$, $(x_i^{(1)}, r^{(1)})$, $(x_i^{(1)}, x_i^{(2)})$, $(x_i^{(1)}, x_i^{(3)})$, for each element z_i, and the edges $(s_j, r^{(1)})$ for every set $Q_j \in \mathcal{Q}$. An example of such a $T_{r^{(2)}}$ is depicted in Figure 2-(a). Similarly, for every $z_i \in Z$, the RT $T_{x_i^{(2)}}$ covers edges $(x_i^{(2)}, x_i^{(3)})$, $(x_i^{(2)}, x_i^{(1)})$, $(x_i^{(1)}, u_i)$ and $(x_i^{(1)}, w_i)$. $T_{x_i^{(2)}}$ also covers all edges (s_j, u_i) for every set Q_j that contains element z_i, and edges $(r^{(1)}, r^{(2)})$ and $(r^{(1)}, r^{(3)})$. An example of the RT $T_{x_2^{(2)}}$ is depicted in Figure 2-(b). Thus, the only remaining uncovered edges are (u_i, w_i), for each element z_i. Since Q_{i_j}, $j = 1, \ldots, k$, is a solution to the SC problem, these edges are covered by the RTs $T_{s_{i_j}}$, as depicted in Figure 2- (c). Thus, S is a set of at most $k + m + 1$ stations that covers all the edges in the graph $G_{\mathcal{R}(I)}$.

[3] These weights do not represent communication costs.

Next, we show that if there is a set of at most k+m+1 stations that covers all the graph edges then there is a solution for the SC problem of size at most k. Note that there needs to be a monitoring station in each anchor clique and suppose w.l.o.g that the selected stations are $r^{(2)}$ and $x_i^{(2)}$ for each element z_i. None of these m + 1 stations covers edges (u_i,w_i) for elements $z_i \in Z$. The other k monitoring stations are placed in the nodes u_i,w_i and s_j. In order to cover edge (u_i,w_i), there needs to have a station at one of the nodes u_i, w_i or s_j for some set Q_j containing element z_i. Also, observe that the RTs of u_i and w_i cover only edge (u_i,w_i) for element z_i and no other element edges. Similarly, the RT of s_j covers only edges (u_i,w_i) for elements z_i contained in set Q_j . Let $\mathcal{S}$ be a collection of sets defined as follows. For every monitoring station at any node s_j add the set $Q_j \in \mathcal{Q}$ to $\mathcal{S}$, and for every monitoring station at any node u_i or w_i we add to $\mathcal{S}$ an arbitrary set $Q_j \in \mathcal{Q}$ such that $z_i \in Q_j$. Since the set of monitoring stations cover all the element edges, the collection $\mathcal{S}$ covers all the elements of Z, and is a solution to the SC problem of size at most k. □

The above reduction $\mathcal{R}(I)$ can be extended to derive a lower bound for the best approximation ratio achievable by any algorithm. This reduction and the proof of Theorem 2 are given in [25].

Theorem 2 *The lower bound of any approximation algorithm for the LM problem is* $\frac{1}{2} \cdot \ln(|V|)$.

5.1.2 A greedy algorithm for the LM and WLM problems

We turn to present an efficient algorithm for solving the LM and the WLM problems. The algorithm maps the given instance of LM or WLM problem to an instance of the SC problem and uses a greedy heuristic for solving the SC instance, which achieves a near tight upper bound for the LM and WLM problems.

$\mathcal{S} \leftarrow \emptyset$, $C \leftarrow Z$
While $C \neq \emptyset$ do
    For every $Q_v \in \mathcal{Q} - \mathcal{S}$ do
        $n_v \leftarrow |Q_v \bigcap C|$
    $Q \leftarrow arg\ \min_{Q_v, Q_v \notin \mathcal{S}} \frac{w_v}{n_v}$
    $\mathcal{S} \leftarrow \mathcal{S} \bigcup \{Q\}$
    $C \leftarrow C - Q$
End While
Return $\mathcal{S}$

Fig. 3. A formal description of the Greedy Heuristic for Set-Cover.

For a given WLM problem involving the graph $G(V,E)$ we define an instance of the SC problem as follows. The set of edges, E, defines the universe of elements, Z. The collection of sets $\mathcal{Q}$ includes the subsets $Q_v = \{e \mid e \in T_v\}$ for every node $v \in V$, where the weight of each subset Q_v is equal to w_v, the weight of the corresponding node v. The greedy heuristic is an iterative algorithm that selects in each iteration the most cost-effective set. Let $C \subseteq Z$ be the set of uncovered elements. In addition, let $n_v = \{Q_v \cap C\}$ be the number of uncovered elements in the set Q_v, for every $v \in V$, at the beginning of each iteration. The algorithm works as follows. It initializes $C \leftarrow Z$. Then, in each iteration, it selects the set Q_v with the

minimum $\frac{w_v}{n_v}$ ratio and removes all its elements from the set C. This step is done until C becomes empty. A formal description of the algorithm is presented in Figure 3.

Theorem 3 *The greedy algorithm computes a* $\ln(|V|)$*-approximation for the LM and WLM problems.*

Proof: According to [11], the greedy algorithm is a H(d)-approximation algorithm for the SC problem, where d is the size of the biggest subset and $\mathcal{H}(d) = \sum_{i=1}^{d} \frac{1}{i}$ is the harmonic sequence. For the LM and WLM problems, every subset includes all the edges of the corresponding RT and its size is exactly $|V| - 1$. Hence, the approximation ratio of the greedy algorithm is $H(|V| - 1) \leq \ln(|V|)$.

Note that the worst-case time complexity of the greedy algorithm can be shown to be $O(|V|^3)$.

5.2 An efficient probe assignment algorithm

Once we have selected a set S of monitoring stations, we need to compute a probe assignment $\mathcal{A}$ for measuring the latency of the network links. Recall from Section 4 that a *feasible probe assignment* is a set of pairs $\{(s, u) | s \in S, u \in V\}$. Each pair (s, u) represents a probe message that is sent from station s to node u and for every edge $e = (u, v) \in E$, there is a station $s \in S$ such that $e \in T_s$ and $\mathcal{A}$ contains the pairs (s, u) and (s, v). The cost of a probe assignment $\mathcal{A}$ is $COST_{\mathcal{A}} = \sum_{(s,u)\in \mathcal{A}} c_{s,u}$ and the *optimal probe assignment* is the one with the minimum cost.

5.2.1 Hardness of the probe assignment problem

In the following, we show that computing the optimal probe assignment is NP-hard even if we choose all $c_{s,u} = 1$ that minimizes the number of probes in $\mathcal{A}$. A similar proof can be used to show that the problem is NP-hard for the case when $c_{s,u}$ equals the minimum number of hops between s and u (this results in a set of probes traversing the fewest possible network links).

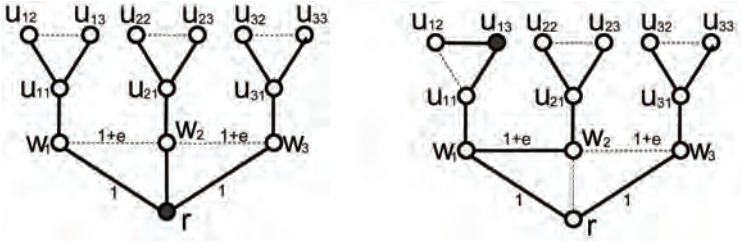


Fig. 4. The RTs of nodes r and u_{13}.

Theorem 4 *Given a set of stations S, the problem of computing the optimal probe assignment is NP-hard.*

Proof: We show a reduction from the vertex cover (VC) problem [24], which is as follows: Given k and a graph $\hat{G} = (\hat{V}, \hat{E})$, does there exist a subset $V' \subseteq \hat{V}$ containing at most k vertices such that each edge in $\hat{E}$ is incident on some node in V'. For a graph $\hat{G}$, we define an instance of the probe assignment problem, and show that there is a vertex cover of size at most k for $\hat{G}$ if and only if there is a feasible probe assignment $\mathcal{A}$ with cost no more than

$COST_{\mathcal{A}} = 5 \cdot |\hat{V}| + |\hat{E}| + k$. We assume that all $c_{s,u} = 1$ (thus, $COST_{\mathcal{A}}$ is the number of probes in $\mathcal{A}$).

For a graph $\hat{G}$, we construct the network graph $G(V,E)$ and a set of stations S for the probe assignment problem as follows. In addition to a root node r, the graph G contains, for each node $\hat{v}_i$ in $\hat{G}$, four nodes denoted by w_i, u_{i1}, u_{i2} and u_{i3}. These nodes are connected with the following edges (w_i, r), (w_i,u_{i1}), (u_{i1}, u_{i2}), (u_{i1}, u_{i3}) and (u_{i2}, u_{i3}). Also, for edge $(\hat{v}_i,\hat{v}_j)$ in $\hat{G}$, we add the edge (w_i,w_j) to G. For instance, the graph G for $\hat{G}$ containing nodes $\hat{v}_1$, $\hat{v}_2$ and $\hat{v}_3$, and edges $(\hat{v}_1,\hat{v}_2)$ and $(\hat{v}_2,\hat{v}_3)$ is shown in Figure 4. The weight of each edge (w_i,w_j) in G is set to $1 + \epsilon$, while the remaining edges have a weight of 1. Finally, we assume that there are monitoring stations at node r and nodes u_{i3} for each vertex $\hat{v}_i \in \hat{G}$. Figure 4 illustrates the RTs of nodes r and u_{13}. Note that edge (w_i,w_j) is only contained in the RTs of u_{i3} and u_{j3}, and (u_{i1}, u_{i2}) is not contained in the RT of u_{i3}.

We first show that if there exists a vertex cover V' of size at most k for $\hat{G}$, then there exists a feasible assignment $\mathcal{A}$ containing no more than $5 \cdot |\hat{V}| + |\hat{E}| + k$ probes. For measuring the latency of the five edges corresponding to $\hat{v}_i \in \hat{V}$, $\mathcal{A}$ contains five probe messages: (r,w_i), (r, u_{i1}), (r, u_{i2}), (u_{i3}, u_{i1}) and (u_{i3}, u_{i2}). So (w_i,w_j) (corresponding to edges $(\hat{v}_i,\hat{v}_j)$ in $\hat{E}$) are the only edges in G whose latency still remains to be measured. Since V' is a vertex cover of $\hat{G}$, it must contain one of $\hat{v}_i$ or $\hat{v}_j$. Suppose $\hat{v}_i \in V'$. Then, $\mathcal{A}$ contains the following two probes (u_{i3},w_i) and (u_{i3},w_j) for each edge (w_i,w_j). Since the probe message (u_{i3},w_i) is common to the measurement of all edges (w_i,w_j) corresponding to edges covered by $\hat{v}_i \in V'$ in $\hat{G}$, and size of V' is at most k, $\mathcal{A}$ contains at most $5 \cdot |\hat{V}| + |\hat{E}| + k$ probes.

We next show that if there exists a feasible probe assignment $\mathcal{A}$ containing at most $5 \cdot |\hat{V}| + |\hat{E}| + k$ probes, then there exists a vertex cover of size at most k for $\hat{G}$. Let V' consist of all nodes $\hat{v}_i$ such that $\mathcal{A}$ contains the probe (u_{i3},w_i). Since each edge (w_i,w_j) is in the RT of only u_{i3} or u_{j3}, $\mathcal{A}$ must contain one of (u_{i3},w_i) or (u_{j3},w_j), and thus V' must be a vertex cover of $\hat{G}$. Further, we can show that V' contains at most k nodes. Suppose that this is not the case and V' contains more than k nodes. Then, $\mathcal{A}$ must contain greater than k probes (u_{i3},w_i) for $\hat{v}_i \in \hat{V}$. Further, in order to measure the latencies of all edges in E, $\mathcal{A}$ must contain $5 \cdot |\hat{V}| + |\hat{E}|$ additional probes. Of these, $|\hat{E}|$ are needed for edges (w_i,w_j), $3 \cdot |\hat{V}|$ for edges (u_{i3}, u_{i1}), (u_{i3}, u_{i2}) and (r,w_i), and $2 \cdot |\hat{V}|$ for edges (u_{i1}, u_{i2}). $\mathcal{A}$ contains 2 probe messages for each edge (u_{i1}, u_{i2}) because the edge does not belong to the RT of u_{i3} and thus 2 probe messages (v, u_{i2}) and (v, u_{i1}), $v \neq u_{i3}$ are needed to measure the latency of edge (u_{i1}, u_{i2}). This, however, leads to a contradiction since $\mathcal{A}$ would contain more than $5 \cdot |\hat{V}| + |\hat{E}| + k$ probes. Thus V' must be a vertex cover of size no greater than k. □

5.2.2 Probe assignment algorithms

We first describe a *simple probe assignment* algorithm that computes an assignment $\mathcal{A}$ whose cost is within a factor of 2 of the optimal. Consider a set of monitoring stations S and for every edge $e \in E$, let $S_e = \{s \mid s \in S \wedge e \in T_s\}$ be the set of stations that can monitor e. For each $e = (u, v) \in E$, select the station $s_e \in S_e$ for which the cost $c_{s_e,u} + c_{s_e,v}$ is minimum. Then add the pairs (s_e, u) and (s_e, v) to $\mathcal{A}$. As a result, the returned assignment is,

$$\mathcal{A} = \{(s_e, u), (s_e, v) | e = (u, v) \in E \wedge s_e \in S_e \wedge s_e = arg \min_{s \in S_e}(c_{s,u} + c_{s,v})\}$$

Theorem 5 *The approximation ratio of the simple probe assignment algorithm is* 2.

Proof: For monitoring the delay of any edge $e \in E$, at least one station $s \in S$ must send two probe messages, one to each endpoint of e. As a result, in any feasible probe assignment at least one probe message can be associated with each edge e. Let it be the message that is sent to the farthest endpoint of e from the monitoring station. Let $\mathcal{A}^*$ be the optimal probe assignment and let s_e^* be the station that monitors edge e in $\mathcal{A}^*$. So, in $\mathcal{A}^*$, the cost of monitoring edge $e = (u, v)$ is at least $\max\{c_{s_e^*,u}, c_{s_e^*,v}\}$. Let s_e be the selected station for monitoring edge e in the assignment $\mathcal{A}$ returned by the simple probe assignment algorithm. s_e minimizes the cost $c_{s,u} + c_{s,v}$, for every $s \in S_e$. Thus, $c_{s_e,u} + c_{s_e,v} \le c_{s_e^*,u} + c_{s_e^*,v} \le 2 \cdot \max\{c_{s_e^*,u}, c_{s_e^*,v}\}$. Thus, $COST_{\mathcal{A}} \le 2 \cdot COST_{\mathcal{A}^*}$. □

Note that the time complexity of the simple probe assignment algorithm can be shown to be $O(|S| \cdot |V|^2)$.

Example 1 This example shows that the simple probe assignment algorithm has a tight approximation ratio of 2. Suppose that the cost of sending any message is 1 and consider the graph depicted in Figure 5. Let the monitoring stations be $S = \{s_1, s_2\}$ and consider the following message assignment, $\mathcal{A}$, that may be calculated by the simple algorithm. The edges (s_1, s_2), (s_1, v_1) and (v_i, v_{i+1}) of every odd i are assigned to station s_1. The edges (s_2, v_1) and (v_i, v_{i+1}) of every even i are assigned to station s_2. In this message assignment both s_1 and s_2 send probe messages to every node v_i and in additional s_1 send probe message to s_2. Hence, $COST_{\mathcal{A}} = 1 + 2n$. At the optimal assignment, $\mathcal{A}^*$, all the edges (v_i, v_{i+1}) are assigned to a single station either s_1 or s_2. Here, s_1 sends messages to s_2 and v_1, station s_2 also sends message to v_1, and one message is sent to every node v_i, $i > 1$ either from s_1 or s_2. Hence, $COST_{\mathcal{A}^*} = 2 + n$, and the limit $\lim_{n\to\infty} \frac{COST_{\mathcal{A}}}{COST_{\mathcal{A}^*}} = \lim_{n\to\infty} \frac{1+2\cdot n}{2+n} = 2$.

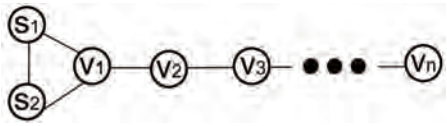


Fig. 5. An example of a probe assignment that cost twice than the optimal.

We turn now to describe a *greedy probe assignment* algorithm that also guarantees a cost within a factor of 2 of the optimal, but yields better results than the simple algorithm in the average case. It is based on the observation that a pair of probe messages is needed for monitoring a link, however, a single message may appear in several such pairs. It attempts to maximize the usage of each message for monitoring the delay of several adjacent links. This is an iterative algorithm that keeps for each station-edge pair (s, e), $e \in T_s$, the current cost, $w_{s,e}$, of monitoring edge e by station s. At each iteration the algorithm select the pair (s', e') with the minimal cost and add the required messages to the message assignment $\mathcal{A}$. If several pairs have the same cost the one with minimal number of hopes between the station and the edge is selected. Probe messages in $\mathcal{A}$ are considered as already been paid and the algorithm update the cost of monitoring the adjacent edges of e' by station s'. This operation is done until all the edges are monitored. A formal description of the algorithm is given in Figure 6, where L is the set of unassigned edges and the initial value of $w_{s,e} \leftarrow c_{s,u} + c_{s,v}$, for every $e = (u, v) \in E$ and $s \in S_e$.

$\mathcal{A} \leftarrow \emptyset$
$L \leftarrow E$
For every $s \in S$ and for every $e = (u, v) \in T_s$ do
    $w_{s,e} \leftarrow c_{s,u} + c_{s,v}$
While $L \neq \emptyset$ do
    Let (s', e') be the pair with minimal $w_{s,e}$ value
    $\mathcal{A} \leftarrow \mathcal{A} \bigcup \{(s', u'), (s', v') | e' = (u', v')\}$
    $L \leftarrow L - \{e'\}$
    For every $\tilde{e} = (x, y) \in T_{s'} \ \wedge y \in \{u', v'\}$ do
        $w_{s',\tilde{e}} \leftarrow c_{s',x}$
    End For
End While
Return $\mathcal{A}$

Fig. 6. The Greedy Probe Assignment Algorithm.

Recall that the algorithm assigns links to the monitoring stations from near to far. First it assigns to each station its adjacent links. Then it continues by assigning links, which are adjacent to the already assigned links. In this way it attempts to avoid the situation where two adjacent links, that should be assigned to the same station, eventually are assigned to two different monitoring stations. The greedy algorithm yields the optimal probe assignment for the graph in Example 1.

Theorem 6 *The approximation ratio of the greedy probe assignment algorithm is* 2.

Proof: Each link $e = (u, v) \in E$ is monitored by the station that minimize the cost $w_{s,e}$. This cost is at most $\min_{s \in S_e}(c_{s,u} + c_{s,v})$. As we have shown in Theorem 5 this guarantees a solution with in a factor of 2 from the optimal. □

6. Path monitoring algorithms

In this section, we address the problem of designing an accurate *path monitoring* system that guarantees that every routing path is monitored by a single monitoring station. First, we present the need for path monitoring and then we provide greedy algorithms for station selection and probe assignment.

6.1 The need for path monitoring

A delay-monitoring system should be able to provide accurate estimates of the end-to-end delay of the routing paths between arbitrary nodes in the network. In the monitoring framework described in the previous section, each link is associated with a single monitoring station that monitors its delay. Thus, the end-to-end delay of any path can be estimated by accumulating the delays of all the links along the path. A drawback of this approach is that the accuracy of a path delay estimation decreases as the number of links that compose the path increases. A better estimate can be achieved by partitioning each path into a few contiguous segments. Each segment is then required to be in the RT of a single monitoring station, which estimates its delay by sending two probe messages to the segment's end-points. Of course, the best estimate of delay is obtained when every path consists of a single segment. Unfortunately, the link monitoring scheme presented in Section

5.1 cannot guarantee an upper bound on the number of segments in a path. In fact, this number may be as high as the number of links in the path, even when the number of monitoring stations is small, as illustrated by the following example.

Example 2 Consider a graph that consists of a grid of size $k \times k$ and two additional nodes, a and b, as depicted in Figure 7-(a). The weight of each grid edge is 1 except for edges along the main diagonal from node c to d whose weights are 1-ϵ. Also, the weights of edges incident on nodes a and b vary between 1* and k* as shown in Figure 7-(a), where $n^* = n \cdot (1 - \epsilon)$. Monitoring stations are located at nodes a and b, and their RTs are along their SPTs, as shown in Figures 7-(b) and 7-(c), respectively. In this graph, the shortest path from node c to d is composed of the edges along the main diagonal of the grid, as shown in Figure 7-(d). Note that any pair of adjacent edges along this path are monitored by two different stations. Thus, each edge in this path represents a separate segment and the number of segments that cover this path is $2 \cdot (k\text{-}1)$, even though the number of stations is only two. □

In this section, we address the problem of designing an accurate *path monitoring* system that guarantees that every routing path is monitored by a single station. Thus, for every path $P_{u,v}$ there is a monitoring station $s \in S$ such that $P_{u,v} \in T_s$. In such case, the end-to-end delay of the path can be estimated by sending at most three probe messages, as described later in Sub-Section 6.3.

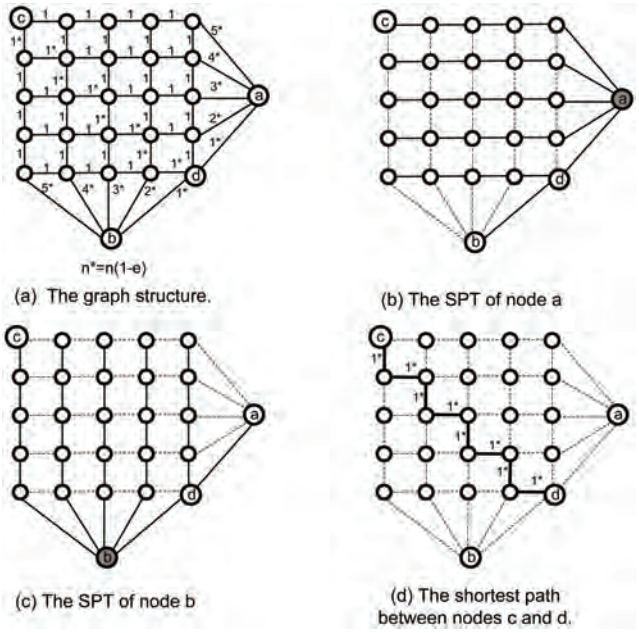


Fig. 7. An example where each edge along a given path is included in a separate segment.

6.2 An efficient station selection algorithm

The station selection problem for path monitoring is defined as follows.

Definition 3 (The Weighted Path Monitoring Problem - WPM): Given a graph $G(V,E)$, with a weight w_v and a RT T_v for every node $v \in V$, and a routing path $P_{u,v}$ between any pair of

nodes $u, v \in V$, find the set $S \subseteq V$ that minimizes the sum $\Sigma_{v \in S} w_v$ such that for every pair $u, v \in V$ there is a station $s \in S$ such that $P_{u,v} \subseteq T_s$. □

In the un-weighted version of the WPM problem, termed the *path monitoring* (PM) problem, the weight of every node is 1.

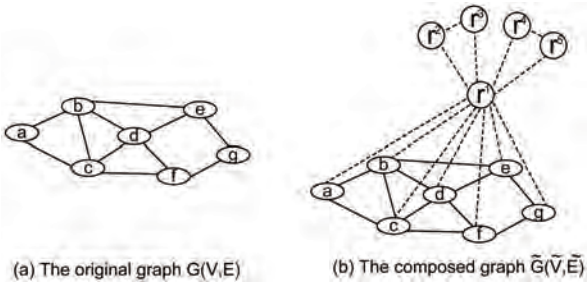


Fig. 8. An example of a graph $G(V,E)$ and the corresponding graph $\tilde{G}$ ($\tilde{V}$, $\tilde{E}$).

Theorem 7 The PM and WPM problems are both NP-Hard.

Proof: We show that the PM and WPM problems are NP-hard by presenting a polynomial reduction from the *vertex cover* (VC) problem[4] to the PM problem. Since the VC problem is a well-known NP-complete problem this proves that the PM and the WPM problems are also NP-hard.

Consider the following reduction from the VC problem to the PM problem. For a given graph $G(V,E)$ we construct a graph $\tilde{G}$ ($\tilde{V}$, $\tilde{E}$) that contains the following nodes and edges. $\tilde{V} = V \bigcup \{r_1, r_2, r_3, r_4, r_5\}$ and the edges $\tilde{E} = E \bigcup \{(v, r_1) \mid v \in V\} \bigcup \{(r_1, r_2), (r_1, r_3), (r_1, r_4), (r_1, r_5), (r_2, r_3), (r_4, r_5)\}$. The weight of every edge $e \in E$ is 3 and the weight of any edge $e \notin E$ is 2. In the following $R = \{r_1, r_2, r_3, r_4, r_5\}$. An example of such graph is given in Figure 8.

Now we will show that the given VC instance, graph $G(V,E)$, has a solution, S of size k if and only if the PM instance, graph $\tilde{G}$ ($\tilde{V}$, $\tilde{E}$) has a solution, $\tilde{S}$ of size $k + 2$. In this proof we assume without lose of generality that the routing tree (RT) of every node is its shortest path tree (SPT). First, let considered the auxiliary structure defined by the nodes in R. The edge (r^2, r^3) is covered only by the SPTs T_{r^2} and T_{r^3}. Therefore, one of these nodes must be included in $\tilde{S}$. Similarly, one of the nodes r^4 or r^5 must be included in $\tilde{S}$ for covering the edge (r^4, r^5). Suppose without lose of generality that the selected nodes are r^2 and r^4.

Let us turn to describe the different SPTs of the nodes in $\tilde{G}$ ($\tilde{V}$, $\tilde{E}$). The SPTs T_{r^2} and T_{r^4} are very similar. The SPT T_{r^2} contains the edge (r^2, r^3) and all the incident edges of node r^1 except edge (r^1, r^3). The SPT T_{r^4} contains the edge (r^4, r^5) and all the incident edges of node r^1 except edge (r^1, r^5). These two SPTs together guarantee that any shortest path that one of its end-node is in the set R is covered. They also cover the shortest path between every pair of nodes $u, v \in V$ such that $(u, v) \notin E$. The only shortest paths that are not covered by the two SPTs T_{r^2} and T_{r^4} are the one-edge paths defined by E. Let N_v be the set of adjacent nodes to node v in the graph $\tilde{G}$ ($\tilde{V}$, $\tilde{E}$). The SPT T_v of every node $v \in V$ contains of the set of edges, $T_v = \{(v, u) \mid u \in N_v\} \bigcup \{(r^1, u) \mid u \notin \tilde{V} - N_v\}$.

[4] Definition of the vertex cover problem is given in the proof of Theorem 4.

Consider a solution S of size k to the VC problem defined by graph $G(V,E)$. Then $\tilde{S} = S \bigcup \{r^2, r^4\}$ is a solution of size k+2 to the corresponding PM instance $\tilde{G}(\tilde{V}, \tilde{E})$. At least one end-point of every edge $e \in E$ is a node in S. Therefore, $\bigcup_{s \in S} T_s$ covers all the paths with only one edge between any pairs of nodes in V. The rest of the paths are covered by the SPTs T_{r^2} and T_{r^4}. Hence, $\tilde{S}$ is a solution of size k + 2 to the PM problem.

Let $\tilde{S}$ be a solution of size k + 2 to the PM problem defined by the graph $\tilde{G}(\tilde{V}, \tilde{E})$. Then $S = \tilde{S} \cap V$ is a solution of size at most k to the VC instance $G(V,E)$. The set $\tilde{S}$ must include at least two nodes from the set R. Thus, $|S| \leq k$. The SPTs of the nodes in R do not contain any edge in E. Therefore the edges of E are covered only by SPTs of nodes in S. Since for every node $v \in V$ holds that $T_v \cap E = \{(v, u) \mid u \in N_v - \{r^1\}$, The set S is a solution to the instance $G(V,E)$ of the given VC problem. □

6.2.1 Lower bounds for the PM and WPM problems

We now turn our attention to computing the lower bounds on the best approximations for the PM and WPM problems. In the sequel, we limit our discussion to cases where monitoring stations are selected from a given subset of nodes $Y \subseteq V$. In our lower bound proof, we use a polynomial reduction, $\hat{\mathcal{R}}(I)$, from any instance $I(Z, \mathcal{Q})$ of the SC problem to a corresponding PM instance. The graph $\hat{G}_{\hat{\mathcal{R}}(I)}(V,E)$ computed by the reduction $\hat{\mathcal{R}}(I)$ contains the following nodes. The nodes u_i and s_j for every element $z_i \in Z$ and set $Q_j \in \mathcal{Q}$, respectively, and three additional nodes u_0, t and r. The node u_0 corresponds to a dummy element z_0 that is included in every set $Q_j \in \mathcal{Q}$, and each one of the nodes t and r is the hub of a star that is connected to the rest of the nodes. The weight of all the graph edges is 1. An example of such a graph $\hat{G}_{\hat{\mathcal{R}}(I)}(V,E)$ is depicted in Figure 9-(a), for the SC instance with four elements $\{z_1, z_2, z_3, z_4\}$ and three sets $Q_1 = \{z_1, z_2\}$, $Q_2 = \{z_2, z_4\}$, $Q_3 = \{z_3, z_4\}$.

We next describe the routing paths between every pair of nodes, which are along their shortest paths. The nodes t and r have a direct edge to every other node. The shortest path between every pair of nodes s_j and s_k is through node t, and between every pair u_i and u_l, it is through node r. Between every pair of nodes s_j and u_i for a set $Q_j \in \mathcal{Q}$ and an element $z_i \in Z$, the shortest path traverses through node t if $z_i \in Q_j$, otherwise it passes through node r. The RTs of the various nodes for of the given example above are depicted in Figure 9. Moreover, for the proof, let assume that the set of possible monitoring stations is $Y = \{s_j \mid \forall Q_j \in \mathcal{Q}\} \bigcup \{r\}$.

Lemma 1 *Consider an instance $I(Z, \mathcal{Q})$ of the SC problem and the corresponding graph $\hat{G}_{\hat{\mathcal{R}}(I)}(V,E)$ and set Y. Then there is a solution of size k to the SC problem if and only if there is a solution of size k + 1 to the corresponding PM problem defined by $\hat{G}_{\hat{\mathcal{R}}(I)}(V,E)$ and Y.*

Proof: Let $\mathcal{S}$ be a solution of size k to the given SC instance. Let set $\hat{S} = \{s_j \mid Q_j \in \mathcal{S}\} \bigcup \{r\}$. We claim that $\hat{S}$ is a feasible solution of size k + 1 to the given PM problem. First, observe that $\hat{S} \subseteq Y$. Further, the RT T_r covers all the shortest paths that pass through node r. Also, the RT of any node $s_j \in \hat{S}$ for a set $Q_j \in \mathcal{Q}$ covers all the shortest paths between arbitrary pairs of nodes s_j and s_k. Thus, we only need to show that all the shortest paths between pairs of nodes s_k and u_i that pass through node t are also covered. This is satisfied since for every u_i, there is a $Q_j \in \mathcal{S}$ such that $z_i \in Q_j$. Thus, $s_j \in \hat{S}$ and T_{s_j} contains all such paths between u_i and s_k through t.

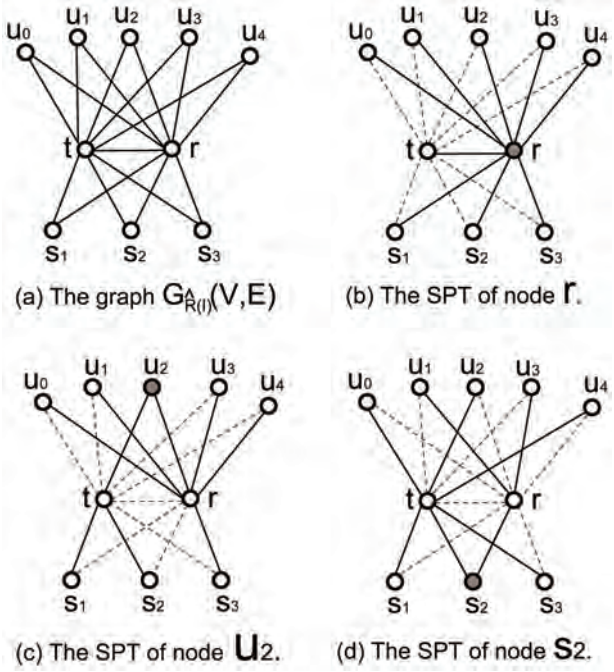


Fig. 9. The graph $G_{\hat{\mathcal{R}}}$ (V,E) and the RTs of the nodes.

Now consider a solution $\hat{S}$ of size $k + 1$ to the PM problem and let $\mathcal{S} = \{Q_j \mid s_j \in \hat{S}\}$. We claim that $\mathcal{S}$ is a solution of size k to the given SC problem. Note that $r \in \hat{S}$ for covering all the shortest paths between node u_0 and any node u_i. This is because element z_0 is contained in all sets of $\mathcal{Q}$ and thus edge (u_0, r) is not contained in any RT T_{s_j}. Hence, $\mathcal{S}$ is of size k. The set $\hat{S} - \{r\}$ covers all the shortest paths that pass through node t. Every element $z_i \in Z$ appears in at least one set $Q_k \in \mathcal{Q}$. Thus, there is a shortest path through t between every node u_i and at least one node s_k. This path is covered by at least one node $s_j \in \hat{S}$ for whom the shortest path to u_i also passes through t. As a result, for every element $z_i \in Z$ there is a set $Q_j \in \mathcal{S}$ such that $z_i \in Q_j$. □

Theorem 8 *The lower bound of any approximation algorithm for the PM and WPM problems is* $\ln(|V|)$.

Proof: Let J be a bad SC instance with m elements and $n \simeq c_1 \cdot (\ln m)^{c_2}$ subsets, as constructed in [26] for proving the $\ln(m)$ lower bound for the SC problem. Let $\hat{G}_{\hat{\mathcal{R}}(J)}$ be the graph calculated by $\hat{\mathcal{R}}$. The lower bound for the PM problem, $\delta_{PM}(|V|)$, satisfies,

$$\delta_{PM}(|V|) \geq \frac{COST_{PM}(\hat{G}_{\hat{\mathcal{R}}(J)})}{OPT_{PM}(\hat{G}_{\hat{\mathcal{R}}(J)})} \simeq \frac{\ln(m) \cdot OPT_{SC}(J) + 1}{OPT_{SC}(J) + 1} \simeq \ln(m)$$

The number of nodes in the graph $\hat{G}_{\hat{\mathcal{R}}(J)}$ is $|V| = 3+m+n \simeq m+c_1 \cdot (\ln m)^{c_2}$. Consequently, for a large m we assume that $|V| \simeq m$ and thus, $\delta_{PM}(|V|) \geq \ln(|V|)$. □

6.2.2 A greedy algorithm for station selection

Similar to the WLM problem, an efficient solution to a WPM instance is obtained by mapping it to an instance of the CS problem and using the greedy heuristic given in Figure 3 to solve this problem. Consider a graph $G(V,E)$, a weight w_v and an RT T_v for every node $v \in V$ and let $P_{u,v}$ be the routing path between any nodes $u, v \in V$. The corresponding SC instance in as follows. Every shortest path $P_{u,v}$ is represented by an element, denoted by $[u, v]$. Thus, the universe of elements, Z, contains $\binom{|V|}{2} = \frac{|V|\cdot(|V|-1)}{2}$ elements. For every node $v \in V$ we define a set Q_v with weight w_v that contains all the routing paths covered by the RT of node v, *i.e.*, $Q_v = \{[x, y] \mid x, y \in V, x \notin y, P_{x,y} \subseteq T_v\}$. Now consider a feasible solution $S = \{Q_{v_1}, Q_{v_2}, \cdots, Q_{v_k}\}$ to the defined SC problem. Then, $S = \{v \mid Q_v \in S\}$ defines a feasible solution to the WPM problem and for every path $P_{u,v}$, $u, v \in V$ there is a monitoring station $s \in S$ such that $P_{u,v} \subseteq T_s$. As a result, an efficient solution to the CS problem defines also an efficient solution of the WPM problem.

Theorem 9 *The greedy algorithm computes a* $2 \cdot \ln(|V|)$*-approximation for the PM and WPM problems.*

Proof: Similar to the proof of Theorem 3. □

6.3 An efficient probe assignment algorithm

Suppose that the greedy algorithm selects a set of monitoring stations S. A monitoring station $s \in S$ can monitor any path $P_{u,v} \subseteq T_s$ by sending at most three probe messages to nodes u, v and w, where $w \in P_{u,v}$ is the common ancestor of nodes u, v in the tree T_s. Let $delay(y, x)$ be the estimated delay of the path between nodes y and x. Since, s can estimate its delay to the nodes u, v and w, the delay of the path $P_{u,v}$ can be computed as follows:

$$delay(u,v) = delay(s,u) + delay(s,v) - 2 \cdot delay(s,w)$$

Theorem 10 *Given a set of stations S, the problem of computing the optimal probe assignment is NP-hard.*

Proof: We show a similar reduction to the one given in the proof of Theorem 4 from the vertex cover (VC) problem. For a given graph $\hat{G}$ $((\hat{V}), \hat{E})$, we define an instance of the probe assignment problem and show that there exists a vertex cover of size at most k for $\hat{G}$ if and only if there exists a feasible probe assignment $\mathcal{A}$ with cost no more than $COST_{\mathcal{A}} = 2 \cdot |\hat{V}| + 2 \cdot |\hat{E}| + k + 1$. We assume that all $c_{s,u} = 1$ (thus, $COST_{\mathcal{A}}$ is the number of probes in $\mathcal{A}$).

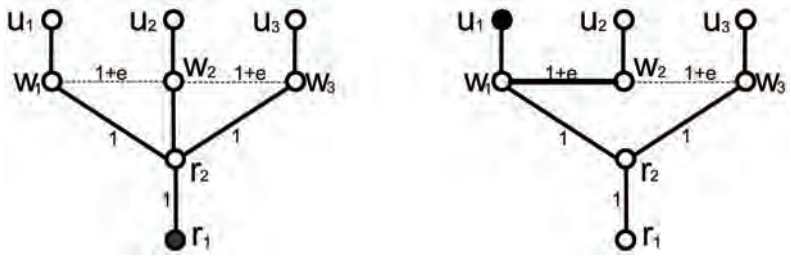


Fig. 10. The RTs of nodes r and u_{13}.

For a graph $\hat{G}$, we construct the network graph $G(V,E)$ and set of stations S for the probe assignment problem as follows. The graph G contains two root nodes, denoted by r_1, r_2, and for each node $\hat{v}_i$ in $\hat{G}$ it contains two additional nodes denoted by w_i and u_i. The set E of edges of the graph G consists of the following edges. The edge (r_1, r_2) and for each node $\hat{v}_i$ in G, the edges (r_2,w_i) and (w_i, u_i). Also, for every edge $(\hat{v}_i,\hat{v}_j)$ in $\hat{G}$, we add the edge (w_i,w_j) to G. The weight of each edge (w_i,w_j) in G is set to $1 + \epsilon$, while the remaining edges have a weight of 1. Finally, we assume that there are monitoring stations at node r_1 and nodes u_i for each vertex $\hat{v}_i \in \hat{G}$. For example, consider the graph $\hat{G}(\hat{V}, \hat{E})$ that contains nodes $\hat{V} = \{\hat{v}_1, \hat{v}_2, \hat{v}_3\}$ and edges $(\hat{E}) = \{\hat{v}_1, \hat{v}_2), (\hat{v}_2,\hat{v}_3)\}$. Figure 10 shows the corresponding graph G as well as the routing trees of the nodes r_1 and u_1. Note that edge (w_i,w_j) is only contained in the RTs of u_i and u_j.

We first show that if there exists a vertex cover V' of size at most k for $\hat{G}$, then there exists a feasible assignment $\mathcal{A}$ containing no more than $2 \cdot \left| \hat{V} \right| + 2 \cdot \left| \hat{E} \right| + k + 1$ probes. Recall that by sending a single probe from r_1 to every other node We can calculate the path delay of every path that traverses through node r_2 and every sub-paths of these paths. This requires $2 \cdot \left| \hat{V} \right| + 1$ probes assign to r_1. Thus the only paths that are not monitored by r_2 are the ones that contain an edge (w_i,w_j), corresponding to edges $(\hat{v}_i,\hat{v}_j)$ in $\hat{E}$. These include the paths $P_{u_i,u_j} = \{u_i,w_i,w_j, u_j\}$, $P_{w_i,u_j} = \{w_i,w_j, u_j\}$, $P_{u_i,w_j} = \{u_i,w_i,w_j\}$ and $P_{w_i,w_j} = \{w_i,w_j\}$. Consider such path $P_{u_i,u_j} = \{u_i,w_i,w_j, u_j\}$. This path can be monitored only by u_i or u_j. Let assume without the lose of generality that it is monitored by u_i. This is done by sending a single probe from u_i to u_j. Similarly the path $P_{u_i,w_j} = \{u_i,w_i,w_j\}$ can be monitored by sending a single probe from u_i to w_j. From this follows that $2 \cdot \left| \hat{E} \right|$ are required for each edge in $\hat{E}$. Yet, we still need to monitor the paths $P_{w_i,u_j} = \{w_i,w_j, u_j\}$ and $P_{w_i,w_j} = \{w_i,w_j\}$. This can be done by sending a single message from u_i to w_i. Recall that this probe message can be used for path monitoring of every path P_{w_i,u_j} and P_{w_i,w_j}such that $(\hat{v}_i,\hat{v}_j)$ in $\hat{E}$. Since V' is a vertex cover of $\hat{G}$, it must contain one of $\hat{v}_i$ or $\hat{v}_j$. Let assume the node $\hat{v}_i$. So by selecting node u_i as the monitoring station of the path P_{u_i,u_j} and the corresponding sub-paths that contains the edge (w_i,w_j), only $2 \cdot \left| \hat{E} \right| + k$ additional probes are required.

We next show that if there exists a feasible probe assignment $\mathcal{A}$ containing at most $2 \cdot \left| \hat{V} \right| + 2 \cdot \left| \hat{E} \right| + k + 1$ probes, then there exists a vertex cover of size at most k for $\hat{G}$. As mentioned above, at least $2 \cdot \left| \hat{V} \right| + 1$ probes are required to monitors all the paths that traverse through node r_2 and any sub path of them. Now, let V' consists of all nodes $\hat{v}_i$ such that $\mathcal{A}$ contains the probe (u_i,w_i). Since each edge (w_i,w_j) is in the RT of only u_i or u_j, $\mathcal{A}$ must contain one of (u_i,w_i) or (u_j ,w_j), and thus V' must be a vertex cover of $\hat{G}$. Further, we can show that V' contains at most k nodes. Suppose that this is not the case and V' contains more than k nodes. Then, $\mathcal{A}$ must contain more than k probes (u_i,w_i) for $\hat{v}_i \in \hat{V}$. However, as mentioned above at least $2 \cdot \left| \hat{E} \right|$ probes are required to measure any path $P_{u_i,u_j}$and one of the paths P_{w_i,u_j} or P_{u_i,w_j}. This contradict the assumption that there are $2 \cdot \left| \hat{V} \right| + 2 \cdot \left| \hat{E} \right| + k + 1$ probes. □

Finding a low-cost optimal probe assignment for monitoring path delays is more challenging than computing the probe assignment for link monitoring (see Section 5.2).

Unlike the link monitoring case, we cannot claim that the optimal solution for path monitoring must contain at least one probe for every routing path[5], which makes the problem for paths more dificult to approximate. We believe that a greedy algorithm similar to the one described in Section 5.2 will achieves good results. Yet, finding an algorithm with a low approximation ratio is still an open problem.

7. Experimental results

In this section, we present experimental results for the number of monitoring stations and probe messages required to measure the latency of every link in the network. The experiments are based on network topologies generated using the Waxman Model [27], which is a popular model used by the networking research community (e.g., [15]). Different network topologies are generated by varying three parameters: (1) n, the number of nodes in the network graph; (2) α, a parameter that controls the density of short edges in the network; and (3) β, a parameter that controls the average node degree.

For computing the locations of monitoring stations that cover all the links in the network graph, we use the greedy algorithm described in Section 5.1.2. These monitoring stations are then used to compute probe messages for measuring the latency of every network link using the greedy algorithm from Sub-Section 5.2.2. We consider two measures for the cost of a probe: $c_{s,v} = 1$ and $c_{s,v}$ is the number of links in the shortest path between s and v. The former optimizes the number of probe messages generated, while the latter optimizes total number of links traversed by all the probe messages.

Num Edges	Num Monitors	Num Probes $c_{sv} = 1$	Num Links $c_{sv} =$ hop count
2189	10	2277	7936
5540	24	5651	14104
8947	41	9150	19649
11178	55	11363	22860
16566	93	16699	30755

Table 1. Number of Monitoring Stations/Probes, $n = 1000$, $\alpha = 0.2$, $\beta \in \{0.02, 0.05, 0.08, 0.1, 0.15\}$.

Table 1 depicts the results of our experiments for networks containing 1000 nodes. We vary the number of edges in the graph by increasing the β parameter and leaving α fixed. From the tables, it follows that as the number of edges in the graph increases, we need more monitoring stations to cover all the links in the network. However, even for large networks, a few monitoring stations suffice for monitoring all the links in the network. For instance, for n = 1000 and 2200 edges, only 10 monitoring stations cover all the network links.

[5] We only know that it must contain one probe for every edge.

In terms of probe messages, the number of probe messages generated by the greedy algorithm closely tracks the number of edges, which is good since this implies a high degree of sharing among the probe messages. Note that this is almost optimal since the number of probe messages needed to measure all the network links is at least the number of edges in the graph. Finally, observe that the average number of links traversed by each probe message is fairly low, ranging between 2 and 4 in most cases.

8. Summary

This chapter introduces a greedy approach for delay monitoring of IP networks. It proposed two-phased monitoring scheme that ensures complete coverage of the network from both links and paths point of views, and it minimizes the monitoring overhead on the underlying production network. In the first phase it computes the locations of monitoring stations such that all network links or paths are covered by the minimal number of stations. Subsequently, in the second phase, it computes the minimal set of probe messages to be sent by each station such that the latency of every routing path can be measured. Unfortunately, both the station selection and the probe assignment problems are NP-hard. However, by using greedy approximation algorithms the scheme finds solutions close to the best possible approximations to both the station selection and the probe assignment problems. Further, the experimental results demonstrate the effectiveness of the presented algorithms for accurately monitoring large networks with very few monitoring stations and probe messages close to the number of network links.

9. References

[1] William Stallings. “SNMP, SNMPv2, SNMPv3, and RMON 1 and 2”. *Addison-Wesley Longman* Inc., 1999. (Third Edition).

[2] “NetFlow Services and Applications”. Cisco Systems White Paper, 1999.

[3] S. R. Stevens, “TCP/IP illustrated”, *Addison-Wesley Publishing Company*, 1994.

[4] Cooperative Association for Internet Data Analysis (CAIDA). http://www.caida.org/.

[5] V. Jacobsen. “Pathchar - A Tool to Infer Characteristics of Internet Paths”, April 1997. ftp://ftp.ee.lbl.gov/pathchar.

[6] P. Beyssac, “Bing - Bandwidth Ping”, URL: http://www.freenix.fr/freenix/logiciels/bing.html.

[7] R. L. Carter and M. E. Crovella. “Server Selection Using Dynamic Path Characterization in Wide-Area Networks”, In *Proceedings of IEEE INFOCOM'99*, Kobe, Japan, April 1997.

[8] K. Lai and M. Baker. “Measuring Bandwidth”. In *Proceedings of IEEE INFOCOM’99*, New York City, New York, March 1999.

[9] C. Dovrolis, P. Ramanathan and D. Moore. “What Do Packet Dispersion Techniques Measure?”. In *Proceedings of IEEE INFOCOM’2001*, Anchorage, Alaska, April 2001.

[10] Y. Bejerano abd R. Rastogi, "Robust monitoring of link delays and faults in IP networks". In *Proceedings of the IEEE INFOCOM'2003*, San Francisco, CA, USA, April 2003.

[11] V. Chavatel, "A Greedy Heuristic for the Set-Covering Problem", *Math. of Operation Research*, Vol. 4, No. 3, pp 233-235, 1979.

[12] K. Moore, "SONAR - A Network Proximity Service, Version 1". Internet-Draft, http://www.netlib.org/utk/projects/sonar/ August 1998.

[13] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. F. Gryniewicz, and Y. Jin, "An Architecture for a Global Internet Host Distance Estimation Service", In *Proceedings of IEEE INFOCOM'99*, New York City, New York, March 1999.

[14] S. Jamin, C. Jin, Y. Jin, Y. Raz, Y. Shavitt, and L. Zhang, "On the Placement of Internet Instrumentation", In *Proceedings of IEEE INFOCOM'2000*, Tel Aviv, Israel, March 2000.

[15] Y. Breitbart, C-Y. Chan, M. Garofalakis, R. Rastogi and A. Silberschatz, "Efficiently Monitoring Bandwidth and Latency in IP Networks", In *Proceedings of the IEEE INFOCOM'2000*, Tel-Aviv, Israel, March 2000,

[16] M. Brodie, I. Rish and S. Ma, "Intelligent probing: A cost-effective approach to fault diagnosis in computer networks", In *IBM Systems Journal*, Vol. 31, No. 3, pp 372-385, 2002.

[17] F. Li, M. Thottan, End-to-End Service Quality Measurement Using Source-Routed Probes In *Proceedings of the IEEE INFOCOM'2006*, Barcelona, Spain, April 2006.

[18] Y. Shavitt, X. Sun, A. Wool and B. Yener, "Computing the Unmeasured: An Algebraic Approach to Internet Mapping", In *Proceedings of IEEE INFOCOM'2001*, Alaska, April 2001.

[19] M. Adler, T. Bu, R. Sitaraman and D. Towsley, "Tree Layout for Internal Network Characterizations in Multicast Networks", In *Proceedings of NGC'01*, London, UK, November 2001.

[20] Y. Breitbar, F. Dragan and H. Gobjuka, " Effective network monitoring", In *Proceedings of ICCCN'04*, Rosemont, IL, USA, October 2004.

[21] H. X. Nguyen and P. Thiran, "Active Measurement for Multiple Link Failures Diagnosis in IP Networks, In *Proceedings of PAM'04*, Antibes, France, 2004.

[22] K. Suh, Y. Guo, J. F. Kurose, D. F. Towsley, Locating network monitors: Complexity, heuristics, and coverage, In *Proceedings of Infocom'05*, Miami, FL, USA, March, 2004.

[23] G. R. Cantieni, G. Iannaccone, C. Barakat, C. Diot and P. Thiran, "Reformulating the monitor placement problem: optimal network-wide sampling, In *Proceedings of CoNEXT06*, Lisboa, Portugal, December, 2006.

[24] M. R. Garey and D. S. Johnson. "Computers and Intractability: A Guide to the Theory of NP-Completeness". *W.H. Freeman Publishing Company*, 1979.

[25] Y. Bejerano abd R. Rastogi, "Robust monitoring of link delays and faults in IP networks". IEEE/ACM Trans. on Networking, Vol. 14, No. 5, pp 1092-1103, 2006.

[26] U. Feige, "A threshold of ln *n* for approximating set-cover", *Proceedings of the 28th Annual ACM Symposium on Theory of Computing, 314-318*, 1996.

[27] B. M. Waxman. "Routing of Multipoint Connections". *IEEE Journal on Selected Areas in Communications*, 6(9):1617-1622, December 1988.

3

A Multilevel Greedy Algorithm for the Satisfiability Problem

Noureddine Bouhmala[1] and Xing Cai[2]
[1]*Vestfold University College,*
[2]*Simula Research Laboratory,*
Norway

1. Introduction

The *satisfiability* (SAT) problem is known to be NP-complete [3] and plays a central role in many domains such as computer-aided design, computing theory, and artificial intelligence. Generally, a SAT problem is defined as follows. Given a propositional formula $\Phi = \bigwedge_{j=1}^{m} C_j$ with m clauses and n boolean variables, where each variable has value of either ***True*** or ***False***. Negation of boolean variable x_i is denoted by $\overline{x}_i$. Each clause C_j has the following form:

$$C_j = \bigvee_{k \in \mathcal{I}_j} \vee \bigvee_{k \in \overline{\mathcal{I}}_j},$$

where $\mathcal{I}_j$, $\overline{\mathcal{I}}_j$ are two sets of literals. The literals in $\mathcal{I}_j$ are from a subset of the n boolean variables, and the literals in $\overline{\mathcal{I}}_j$ are from a subset of the negation of the n boolean variables. Moreover, we have $\mathcal{I} \cap \overline{\mathcal{I}}_j = 0$. The task is to determine whether Φ evaluates to true. Such an assignment of the n boolean variables, if it exists, is called a satisfying assignment for Φ (and Φ is called satisfiable). Otherwise Φ is said to be unsatisfiable. Most SAT solvers use a *conjunctive normal form* (CNF) representation of the formula Φ. In CNF, the formula Φ is represented as a conjunction of clauses, each clause is a disjunction of literals, where a literal is a boolean variable or its negation. For example, $P \vee Q$ is a clause containing the two literals P and Q. The clause $P \vee Q$ is satisfied if either P is true or Q is true. When each clause in Φ contains exactly k literals, the restricted SAT problem is called k-SAT. In the numerical experiments of this chapter, we will focus on the 3-SAT problem, where each clause contains exactly 3 literals. Since we have two choices for each boolean variable, and taken over n variables, the size of the search space S is $|S| = 2^n$. The chapter is organized as follows. In Section 2 we review various algorithms for SAT problems. Section 3 explains the basic greedy GSAT algorithm. In Section 4, the multilevel paradigm is described. Section 5 presents the multilevel greedy algorithm. In Section 6, we look at the results from testing the new approach on a test suit of problem instances. Finally in Section 7 we give a summary of the work.

2. Methods for SAT

The SAT problem has been extensively studied due to its simplicity and applicability. The simplicity of the problem coupled with its intractability makes it an ideal platform for exploring new algorithmic techniques. This has led to the development of many algorithms for solving SAT problems which usually fall into two main categories: systematic algorithms and local search algorithms. Systematic search algorithms are guaranteed to return a solution to a SAT problem if at least one exists or prove it insoluble otherwise.

2.1 Systematic search algorithms

The most popular and efficient systematic search algorithms for SAT are based on the Davis-Putnam (DP) [4] procedure which enumerates all possible variable assignments. This is achieved by setting up a binary search tree and proceeding until it either finds a satisfying truth assignment or concludes that no such assignment exists. In each recursive call of the algorithm the propositional formula Φ is simplified by unit propagation. A boolean variable x_i is selected according to a predefined rule among the n boolean variables. Next, find all the clauses that include the literal x_i and delete it from all these clauses. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ be the set of k(<= m) clauses resulting from this process. Similarly, let $\mathcal{D} = \{D_1, D_2, \ldots, D_r\}$ denote the set of r(<= m) clauses resulting from deleting the literal $\bar{x}_i$. Moreover, let $\mathcal{R} = \{R_1, R_2, \ldots, R_{(m-k-r)}\}$ represent the set of $m - k - r$ clauses that include neither of these two literals. Finally, the original propositional formula is reduced to

$$\Phi_{\text{simpler}} = \left(\bigwedge_{i=1}^{k} \bigwedge_{j=1}^{r} (C_i \vee D_j) \right) \bigwedge_{l=1}^{m-r-k} R_l.$$

Note that the propositional formula Φ_{simpler} does not contain the boolean variable x_i since none of the clauses set $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{R}$ include x_i. If thus an empty clause is obtained, the current partial assignment can not be extended to a satisfying one and backtracking is used to proceed with the search; if an empty formula is obtained, i.e., all clauses are satisfied, the algorithm returns a satisfying assignment. If neither of these two situations occur, an unassigned variable is chosen and the algorithm is called recursively after adding a unit clause containing this variable and its negation. If all branches are explored and no satisfying assignment has been reached, the formula is found to be unsatisfiable. For efficiency reasons, the search tree is explored in depth-first search manner. Since we are only interested in whether the SAT problem is satisfiable or not, we stop as soon as the first solution is found. The size of the search tree depends on the branching rule adopted (how to select the branch variable) thereby affecting the overall efficiency of DP. This has led to the development of various improved DP variants which differ in the schemes employed to maximize the efficiency of unit propagation in their branching rules.

2.2 Stochastic local search algorithms

Due to their combinatorial explosion nature, large and complex SAT problems are hard to solve using systematic algorithms. One way to overcome the combinatorial explosion is to give up completeness. Stochastic local search (SLS) algorithms are techniques which use this strategy. SLS algorithms are based on what is perhaps the oldest optimization method: *trial*

and error. Typically, they start with an initial assignment of the variables, either randomly or heuristically generated. Satisfiability can be formulated as an optimization problem in which the goal is to minimize the number of unsatisfied clauses. Thus, the optimum is obtained when the value of the objective function equals zero, which means that all clauses are satisfied. During each iteration, a new solution is selected from the neighborhood of the current one by performing a move. Choosing a good neighborhood and a search method are usually guided by intuition, because very little theory is available as a guide. Most SLS algorithms use a 1-flip neighborhood relation for which two truth value assignments are neighbors if they differ in the truth value of one variable. If the new solution provides a better value in light of the objective function, the new solution replaces the current one. The search terminates if no better neighbor solution can be found.

One of the most popular local search methods for solving SAT is GSAT [9]. The GSAT algorithm operates by changing a complete assignment of variables into one in which the maximum possible number of clauses are satisfied by changing the value of a single variable. An extension of GSAT referred as random-walk [10] has been realized with the purpose of escaping from local optima. In a random walk step, an unsatisfied clause is randomly selected. Then, one of the variables appearing in that clause is flipped, thus effectively forcing the selected clause to become satisfied. The main idea is to decide at each search step whether to perform a standard GSAT or a random-walk strategy with a probability called the walk probability. Another widely used variant of GSAT is the WalkSAT algorithm originally introduced in [12]. It first picks randomly an unsatisfied clause and then in a second step, one of the variables with the lowest break count appearing in the selected clause is randomly selected. The break count of a variable is defined as the number of clauses that would be unsatisfied by flipping the chosen variable. If there exists a variable with break count equals to zero, this variable is flipped, otherwise the variable with minimal break count is selected with a certain probability (noise probability). The choice of unsatisfied clauses combined with the randomness in the selection of variables enable WalkSAT to avoid local minima and to better explore the search space.

Recently, new algorithms [12] [13] [14] [15] [16] have emerged using history-based variable selection strategy in order to avoid flipping the same variable. Apart from GSAT and its variants, several clause weighting based SLS algorithms [17] [18] have been proposed to solve SAT problems. The key idea is associate the clauses of the given CNF formula with weights. Although these clause weighting SLS algorithms differ in the manner how clause weights should be updated (probabilistic or deterministic) they all choose to increase the weights of all the unsatisfied clauses as soon as a local minimum is encountered. Clause weighting acts as a diversification mechanism rather than a way of escaping local minima. Finally, many other SLS algorithms have been applied to SAT. These include techniques such as Simulated Annealing [19], Evolutionary Algorithms [20], and Greedy Randomized Adaptive Search Procedures [21].

3. The GSAT greedy algorithm

This section is devoted to explaining the GSAT greedy algorithm and one of its variants before embedding it into the multilevel paradigm. Basically, the GSAT algorithm begins with a randomly generated assignment of the variables, and then uses the steepest descent

heuristic to find the new truth value assignment which best decreases the number of unsatisfied clauses. After a fixed number of moves, the search is restarted from a new random assignment. The search continues until a solution is found or a fixed number of restart is performed. As with any combinatorial optimization , local minima or plateaus (i.e., a set of neighboring states each with an equal number of unsatisfied clauses) in the search space are problematic in the application of greedy algorithms. A local minimum is defined as a state whose local neighborhood does not include a state that is strictly better. The introduction of an element of randomness (e.g., noise) into a local search methods is a common practice to increase the success of GSAT and improve its effectiveness through diversification [2].

```
Procedure GSAT Random Walk

Input: A set of clauses, MAX_TRIES, MAX_FLIPS, walking probability p;

Output: A satisfying truth assignment of the clauses, if found;

Begin
    /* Initialization */
    For i := 1 To MAX_TRIES Do
        T ← a random truth assignment;
        For j := 1 To MAX_FLIPS Do
            If T satisfies all the clauses Then return T;
            r ← a randomly generated number between 0 and 1;
            If r ≤ p Then
              PossFlips ← a randomly selected variable with the
              largest decrease in the number of unsatisfied clauses;
            Else
              PossFlips ← a random variable in some unsatisfied clause;
            EndIf
            v ← Pick (PossFlips);
            T ← T with v flipped;
        EndFor
    EndFor
    return "no solution found"
End
```

Fig. 1. The GSAT Random Walk Algorithm.

The algorithm of GSAT Random Walk, which is shown in Figure 1, starts with a randomly chosen assignment. Thereafter two possible criteria are used in order to select the variable to be flipped. The first criterion uses the notion of a "noise" or walk-step probability to randomly select a currently unsatisfied clause and flip one of the variables appearing in it also in a random manner. At each walk-step, at least one unsatisfied clause becomes satisfied. The other criterion uses a greedy search to choose a random variable from the set PossFlips. Each variable in this set, when flipped, can achieve the largest decrease (or the least increase) in the total number of unsatisfied clauses. The walk-step strategy may lead to an increase in the total number of unsatisfied clauses even if improving flips would have been possible. In consequence, the chances of escaping from local minima of the objective function are better compared with the basic GSAT [11].

4. The multilevel paradigm

The *multilevel* paradigm is a simple technique which at its core applies recursive coarsening to produce smaller and smaller problems that are easier to solve than the original one. Figure 2 shows the generic multilevel paradigm in pseudo-code. The multilevel paradigm consists of three phases: coarsening, initial solution, and multilevel refinement. During the coarsening phase, a series of smaller problems is constructed by matching pairs of vertices of the input original problem in order to form clusters, which define a related coarser problem. The coarsening procedure recursively iterates until a sufficiently small problem is obtained. Computation of an initial solution is performed on the coarsest level (the smallest problem). Finally, the initial solution is projected backward level by level. Each of the finer levels receives the preceding solution as its initial assignment and tries to refine it by some local search algorithm. A common feature that characterizes multilevel algorithms is that any solution in any of the coarsened problems is a legitimate solution to the original problem. This is always true as long as the coarsening is achieved in a way that each of the coarsened problems retains the original problem's global structure.

```
Procedure Multilevel Paradigm

Input: SAT problem P_0

Output: Solution S_final(P_0)

Begin
    level := 0
    /* Coarsening Phase */
    While (the desired number of levels is not reached)
        P_level+1 := Coarsen (P_level)
        level := level + 1
    end
    /* Initial Solution is computed at the lowest level */
    S(P_level) = Initial Solution (P_level)
    /* Uncoarsening and Refinement phase */
    While (level > 0)
        S_start(P_level-1) := Uncoarsen (S_final(P_level))
        S_final(P_level-1) := Refine (S_start(P_level-1))
        level := level - 1
    end
End
```

Fig. 2. The Multilevel Generic Algorithm.

The key success behind the efficiency of the multilevel techniques is the use of the multilevel paradigm, which offers two main advantages enabling local search techniques to become much more powerful in the multilevel context. First, by allowing local search schemes to view a cluster of vertices as a single entity, the search becomes restricted to only those configurations in the solution space in which the vertices grouped within a cluster are assigned the same label. During the refinement phase a local refinement scheme applies a local transformation within the neighborhood (i.e., the set of solutions that can be reached

from the current one) of the current solution to generate a new one. As the size of the clusters varies from one level to another, the size of the neighborhood becomes adaptive and allows the possibility of exploring different regions in the search space. Second, the ability of a refinement algorithm to manipulate clusters of vertices provides the search with an efficient mechanism to escape from local minima.

Multilevel techniques were first introduced when dealing with the graph partitioning problem (GPP) [1] [5] [6] [7] [8] [22] and have proved to be effective in producing high quality solutions at lower cost than single level techniques. The traveling salesman problem (TSP) was the second combinatorial optimization problem to which the multilevel paradigm was applied and has clearly shown a clear improvement in the asymptotic convergence of the solution quality. Finally, when the multilevel paradigm was applied to the graph coloring problem, the results do not seem to be in line with the general trend observed in GPP and TSP as its ability to enhance the convergence behaviour of the local search algorithms was rather restricted to some problem classes.

5. A multilevel framework for SAT

- **Coarsening:** The original problem P_0 is reduced into a sequence of smaller problems P_0, P_2, . . . , P_m. It will require at least $\mathcal{O}(\log n/n')$ steps to coarsen an original problem with n variables down to n' variables. Let $\mathcal{V}_i^v$ denote the set of variables of P_i that are combined to form a single variable v in the coarser problem P_{i+1}. We will refer to v as a multivariable. Starting from the original problem P_0, the first coarser problem P_1 is constructed by matching pairs of variables of P_0 into multivariables. The variables in P_0 are visited in a random order. If a variable has not been matched yet, then we randomly select another unmatched variable, and a multivariable consisting of these two variables is created. Unmatchable variables are simply copied to the coarser level. The new multivariables are used to define a new and smaller problem. This coarsening process is recursively carried out until the size of the problem reaches some desired threshold.
- **Initial solution:** An initial assignment $\mathcal{A}_m$ of P_m is easily computed using a random assignment algorithm, which works by randomly assigning to each multivariable of the coarsest problem P_m the value of true or false.
- **Projection:** Having optimized the assignment $\mathcal{A}_{k+1}$ for P_{k+1}, the assignment must be projected back to its parent P_k. Since each multivariable of P_{k+1} contains a distinct subset of multivariables of P_k, obtaining $\mathcal{A}_k$ from $\mathcal{A}_{k+1}$ is done by simply assigning the set of variables $\mathcal{V}_k^v$ the same value as $v \in P_{k+1}$ (i.e., $A_k[u] = A_{k+1}[v]$, $\forall u \in \mathcal{V}_k^v$).
- **Refinement:** At each level, the assignment from the previous coarser level is projected back to give an initial assignment and further refined. Although $\mathcal{A}_{k+1}$ is a local minimum of P_{k+1}, the projected assignment $\mathcal{A}_k$ may not be at a local optimum with respect to P_k. Since P_k is finer, it may still be possible to improve the projected assignment using a version of GSAT adapted to the multilevel paradigm. The idea of GSAT refinement as shown in Figure 3 is to use the projected assignment of P_{k+1} onto P_k as the initial assignment of GSAT. Since the projected assignment is already a good one, GSAT will hopefully converge quickly to a better assignment. During each level, GSAT is allowed to perform MAXFLIPS iterations before moving to a finer level. If a solution is not

found at the finest level, a new round of coarsening, random initial assignment, and refinement is performed.

```
Procedure GSAT Refine
Input: P_i, A_i, MAX_FLIPS;
Output: A satisfying truth assignment of the clauses, if found;
Begin
    For j := 1 To MAX_FLIPS Do
        If A_i satisfies all the clauses of P_i Then return A_i;
        PossFlips ← a randomly selected multivariable with the
        largest decrease in the number of unsatisfied clauses;
        v ← Pick (PossFlips);
        A_i ← A_i with v flipped;
    EndFor
End
```

Fig. 3. The GSAT Refinement Algorithm.

6. Experimental results

6.1 Benchmark instances

To illustrate the potential gains offered by the multilevel greedy algorithm, we selected a benchmark suite from different domains including benchmark instances of SAT competition Beijing held in 1996. These instances are by no means intended to be exhaustive but rather as an indication of typical performance behavior. All these benchmark instances are known to be hard and difficult to solve and are available from the SATLIB website (http://www.informatik.tudarmstadt. de/AI/SATLIB). All the benchmark instances used in this section are satisfiable and have been used widely in the literature in order to give an overall picture of the performance of different algorithms. Due to the randomization of the algorithm, the time required for solving a problem instance varies between different runs. Therefore, for each problem instance, we run GSAT and MLVGSAT both 100 times with a max-time cutoff parameter set to 300 seconds. All the plots are given in logarithmic scale showing the evolution of the solution quality based on averaged results over the 100 runs.

6.1.1 Random-3-SAT

Uniform Random-3-SAT is a family of SAT problems obtained by randomly generating 3-CNF formula in the following way: For an instance with n variables and k clauses, each of the k clauses is constructed from 3 literals which are randomly drawn from the $2n$ possible literals (the n variables and their negations), such that each possible literal is selected with the same probability of $1/2n$. Clauses are not accepted for the construction of the problem instance if they contain multiple copies of the same literal or if they are tautological (i.e., they contain a variable and its negation as a literal).

6.1.2 SAT-encoded graph coloring problems

The graph coloring problem (GCP) is a well-known combinatorial problem from graph theory: Given a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = v_1, v_2, \ldots, v_n$ is the set of vertices and $\mathcal{E}$ the set of

edges connecting the vertices, the goal is to find a coloring $\mathcal{C} : \mathcal{V} \rightarrow N$, such that two vertices connected by an edge always have different colors. There are two variants of this problem: In the optimization variant, the goal is to find a coloring with a minimal number of colors, whereas in the decision variant, the question is to decide whether for a particular number of colours, a coloring of the given graph exists. In the context of SAT-encoded graph coloring problems, we focus on the decision variant.

6.1.3 SAT-encoded logistics problems

In the logistics planning domain, packages have to be moved between different locations in different cities. Within cities, packages are carried by trucks while between cities they are transported by planes. Both trucks and airplanes are of limited capacity. The problem involves 3 operators (load, unload, move) and two state predicates (in, at). The initial and goal state specify locations for all packages, trucks, and planes; the plans allow multiple actions to be executed simultaneously, as long as no conflicts arise from their preconditions and effects. The question in the decision variant is to decide whether a plan of a given length exists. SAT-based approaches to logistics planning typically focus on the decision variant.

6.1.4 SAT-encoded block world planning problems

The Blocks World is a very well-known problem domain in artificial intelligence research. The general scenario in Blocks World Planning comprises a number of blocks and a table. The blocks can be piled onto each other, where the down-most block of a pile is always on the table. There is only one operator which moves the top block of a pile to the top of another pile or onto the table. Given an initial and a goal configuration of blocks, the problem is to find a sequence of operators which, when applied to the initial configuration, leads to the goal situation. Such a sequence is called a (linear) plan. Blocks can only be moved when they are clear, i.e., no other block is piled on top of them, and they can be only moved on top of blocks which are clear or onto the table. If these conditions are satisfied, the move operator always succeeds. SAT-based approaches to Blocks World Planning typically focus on the decision variant where the question is to decide whether a plan of a given length exists.

6.1.5 SAT-encoded quasigroup problems

A quasigroup is an ordered pair $(\mathcal{Q}, \cdot)$, where $\mathcal{Q}$ is a set and $\cdot$ is a binary operation on $\mathcal{Q}$ such that the equations $a \cdot x = b$ and $y \cdot a = b$ are uniquely solvable for every pair of elements a, b in $\mathcal{Q}$. The cardinality of the set $\mathcal{Q}$ is called the order of the quasigroup. Let N be the order of the quasigroup $\mathcal{Q}$ then the multiplication table $\mathcal{Q}$ is a table $N \times N$ such that the cell at the coordinate (x, y) contains the result of the operation $x \cdot y$. The multiplication of the quasigroup must satisfy a condition that there are no repeated result in each row or column. Thus, the multiplication table defines a Latin square. A complete Latin square is a table that is filled completely. The problem of finding a complete Latin square can be stated as a satisfiability problem.

6.2 Experimental results

Figures 4-15 show individual results which appear to follow the same pattern within each application domain. Overall, at least for the instances tested here, we observe that the search

pattern happens in two phases. In the first phase, both MLVGSAT and GSAT behave as a hill-climbing method. This phase is short and a large number of the clauses are satisfied. The best assignment climbs rapidly at first, and then flattens off as we mount the plateau, marking the start of the second phase. The plateau spans a region in the search space where flips typically leave the best assignment unchanged. The long plateaus become even more pronounced as the number of flips increases, and occurs more specifically in trying to satisfy the last few remaining clauses. The transition between two plateaus corresponds to a change to the region where a small number of flips gradually improve the score of the current solution ending with an improvement of the best assignment. The plateau is rather of short length with MLVGSAT compared with that of GSAT. For MLVGSAT the projected solution from one level to its finer predecessor offers an elegant mechanism to reduce the length of the plateau as it consists of more degrees of freedom that can be used for further improving the best solution. The plots show a time overhead for MLVGSAT specially for large problems due mainly to data structures settings at each level. We feel that this initial overhead, which is a common feature in multilevel implementations is more susceptible to further improvements, and will be considerably minimized by a more efficient implementation. Comparing GSAT and MLVGSAT for small problems (up to 1500 clauses) and as can be seen from the left sides of Figures 6,8, both algorithms seem to be reaching the optimal quality solutions. It is not immediately clear which of the two algorithms converges more rapidly. This is probably very dependent on the choice of the instances in the test suite. For example the run time required by MLVGSAT for solving instance flat100-239 is more than 12 times higher than the mean run-time of GSAT (25sec vs 2sec). The situation is reversed when solving the instance block-medium (20sec vs 70sec). The difference in convergence behavior of the two algorithms becomes more distinctive as the size of the problem increases. All the plots show a clear dominance of MLGSAT over GSAT throughout the whole run. MLVGSAT shows a better asymptotic convergence (to around 0.008%−0.1%) in excess of the optimal solution as compared with GSAT which only reach around (0.01%- 11%). The performance of MLVGSAT surpasses that of GSAT although few of the curves overlay each other closely, MLVGSAT has marginally better asymptotic convergence.

The quality of the solution may vary significantly from run to run on the same problem instance due to random initial solutions and subsequent randomized decisions. We choose the Wilcoxon Rank test in order to test the level of statistical confidence in differences between the mean percentage excess deviation from the solution of the two algorithms. The test requires that the absolute values of the differences between the mean percentage excess deviation from the solution of the two algorithms are sorted from smallest to largest and these differences are ranked according to the absolute magnitude. The sum of the ranks is then formed for the negative and positive differences separately. As the size of the trials increases, the rank sum statistic becomes normal. If the null hypothesis is true, the sum of ranks of the positive differences should be about the same as the sum of the ranks of the negative differences. Using two-tailed P value, significance performance difference is granted if the Wilcoxon test is significant for $P < 0.05$.

Looking at Table 1, we observe that the difference in the mean excess deviation from the solution is significant for large problems and remains insignificant for small problems.

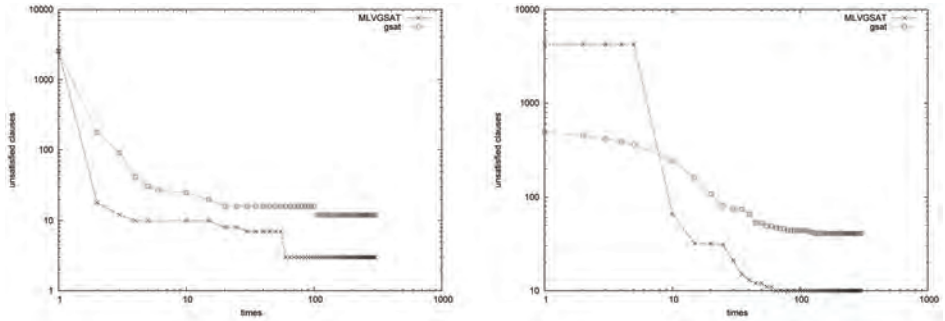


Fig. 4. Log-Log plot:Random:(Left) Evolution of the best solution on a 600 variable problem with 2550 clauses (f600.cnf). Along the horizontal axis we give the time in seconds , and along the vertical axis the number of unsatisfied clauses. (Right) Evolution of the best solution on a 1000 variable problem with 4250 clauses. (f1000.cnf).Horizontal axis gives the time in seconds, and the vertical axis shows the number of unsatisfied clauses.

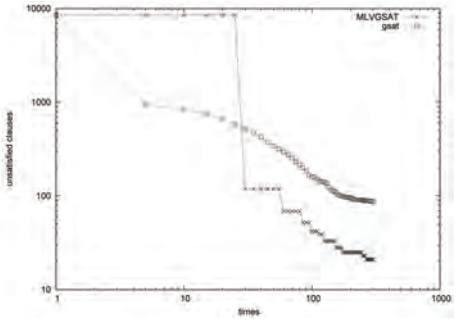


Fig. 5. Log-Log plot: Random:Evolution of the best solution on a 2000 variable problem with 8500 clauses (f2000.cnf). Along the horizontal axis we give the time in seconds , and along the vertical axis the number of unsatisfied clauses.

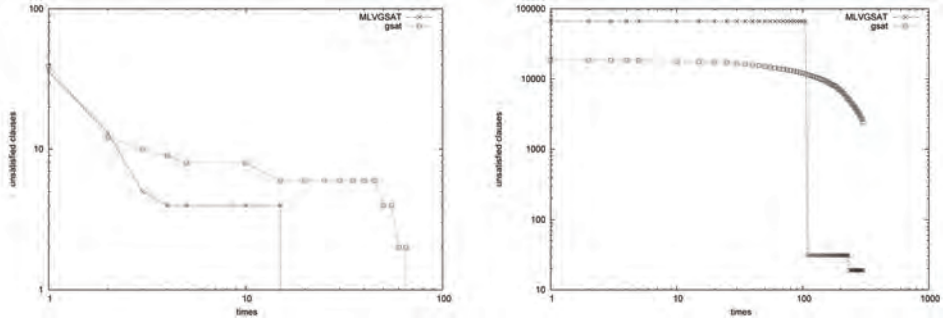


Fig. 6. Log-Log plot: SAT-encoded graph coloring:(Left) Evolution of the best solution on a 300 variable problem with 1117 clauses (flat100.cnf). Along the horizontal axis we give the time in seconds, and along the vertical axis the number of unsatisfied clauses. (Right) Evolution of the best solution on a 2125 variable problem with 66272 clauses (g125-17.cnf). Horizontal axis gives the time in seconds, and the vertical axis shows the number of unsatisfied clauses.

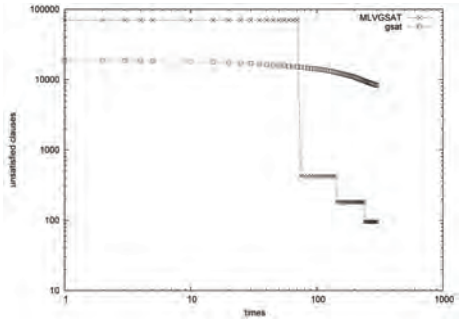


Fig. 7. Log-Log plot: SAT-encoded graph coloring:Evolution of the best solution on a 2250 variable problem with 70163 clauses (g125-18.cnf). Along the horizontal axis we give the time in seconds , and along the vertical axis the number of unsatisfied clauses.

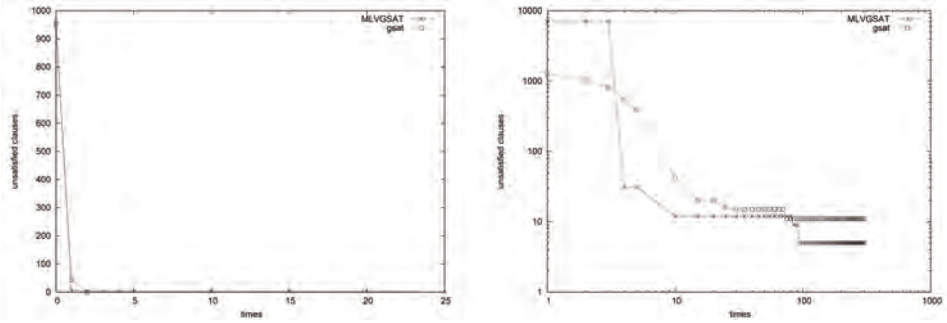


Fig. 8. SAT-encoded block world:(Left) Evolution of the best solution on a 116 variable problem with 953 clauses (medium.cnf). Along the horizontal axis we give the time in seconds , and along the vertical axis the number of unsatisfied clauses. Log-Log plot (Right) Evolution of the best solution on a 459 variable problem with 7054 clauses (huge.cnf). Horizontal axis gives the time in seconds, and the vertical axis shows the number of unsatisfied clauses.

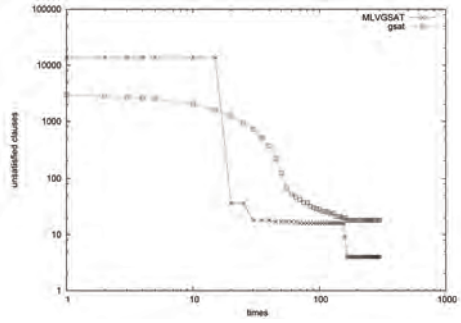


Fig. 9. Log-Log plot: SAT-encoded block world:Evolution of the best solution on a 1087 variable problem with 13772 clauses (bw-largeb.cnf). Along the horizontal axis we give the time in seconds, and along the vertical axis the number of unsatisfied clauses.

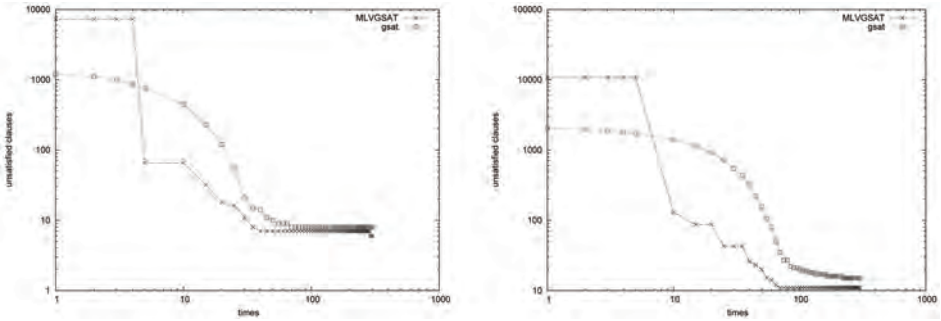


Fig. 10. Log-Log plot: SAT-encoded Logistics:(Left) Evolution of the best solution on a 843 variable problem with 7301 clauses (logisticsb.cnf). Along the horizontal axis is the time in seconds , and along the vertical axis the number of unsatisfied clauses. (Right) Evolution of the best solution on a 1141 variable problem with 10719 clauses (logisticsc.cnf). Horizontal axis gives the time in seconds, and the vertical axis shows the number of unsatisfied clauses.

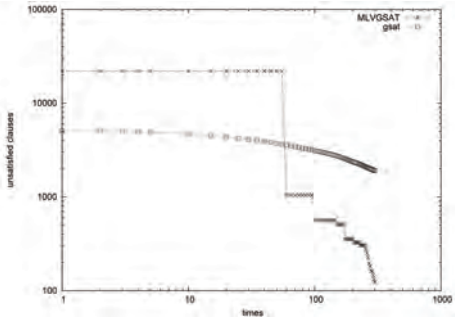


Fig. 11. Log-Log plot:SAT-encoded logistics:Evolution of the best solution on a 4713 variable problem with 21991 clauses (logisticsd.cnf). Along the horizontal axis we give the time in seconds, and along the vertical axis the number of unsatisfied clauses.

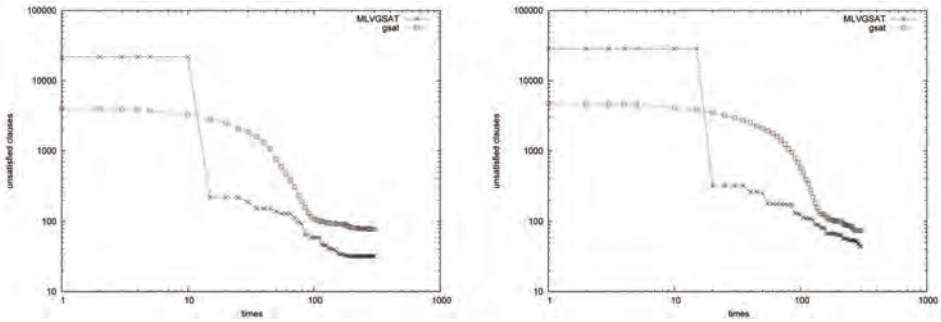


Fig. 12. Log-Log plot:SAT-encoded quasigroup:(Left) Evolution of the best solution on a 129 variable problem with 21844 clauses (qg6-9.cnf). Along the horizontal axis we give the time in seconds , and along the vertical axis the number of unsatisfied clauses.(Right) Evolution of the best solution on a 729 variable problem with 28540 clauses (qg5.cnf). Horizontal axis gives the time in seconds, and the vertical axis shows the number of unsatisfied clauses.

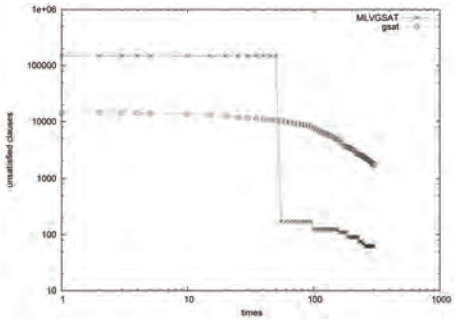


Fig. 13. Log-Log plot:SAT-encoded quasigroup:Evolution of the best solution on a 512 variable problem with 148957 clauses (qg1-8.cnf). Along the horizontal axis we give the time in seconds , and along the vertical axis the number of unsatisfied clauses.

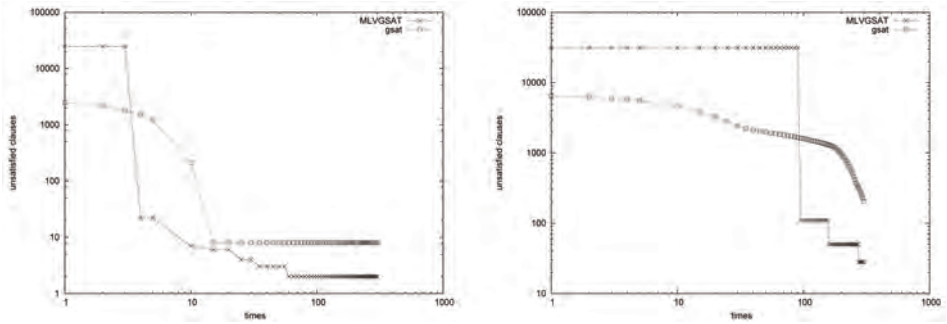


Fig. 14. Log-Log plot:SAT competition Beijing: (Left) Evolution of the best solution on a 410 variable problem with 24758 clauses (4blockb.cnf). Along the horizontal axis we give the time in seconds, and along the vertical axis the number of unsatisfied clauses. (Right) Evolution of the best solution on a 8432 variable problem with 31310 clauses (3bitadd-31.cnf). Horizontal axis gives the time in seconds, and the vertical axis shows the number of unsatisfied clauses.

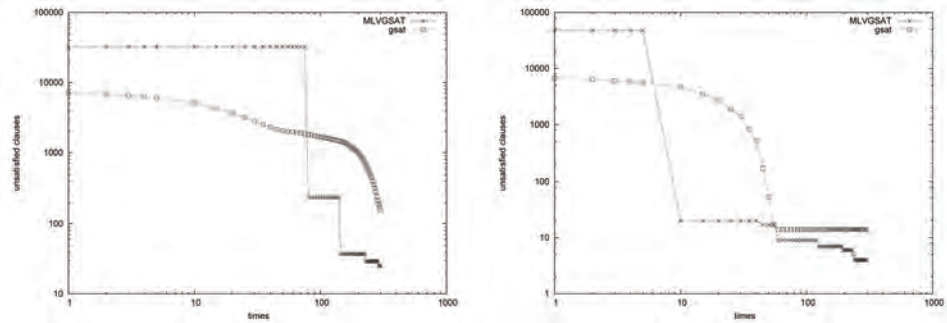


Fig. 15. Log-Log plot:SAT competition Beijing:(Left) Evolution of the best solution on a 8704 variable problem with 32316 clauses (3bitadd32.cnf). Along the horizontal axis we give the time in seconds, and along the vertical axis the number of unsatisfied clauses. (Right) Evolution of the best solution on a 758 variable problem with 47820 clauses (4blocks.cnf). Horizontal axis gives the time in seconds, and the vertical axis shows the number of unsatisfied clauses.

Problem	%EX: MLVGSAT	%EX: GSAT	Null Hypotheis
f600	0.001	0.004	Accept
f1000	0.002	0.009	Accept
f2000	0.002	0.01	Accept
flat100	0	0	Reject
g125-17	0.0002	0.001	Accept
g125-18	0.001	0.11	Accept
logistic-b	0.00008	0.0001	Accept
logistic-c	0.001	0.004	Accept
logistic-d	0.005	0.08	Accept
bw-medium	0	0	Reject
bw-huge	0.0007	0.001	Accept
bw-large-b	0.0002	0.001	Accept
qg6-9	0.001	0.003	Accept
qg5-9	0.001	0.002	Accept
qg1-8	0.0004	0.011	Accept
4block	0.00008	0.0008	Accept
3bitadd-31	0.0008	0.006	Accept
3bitadd-32	0.0007	0.004	Accept
4blocksb	0.00008	0.0002	Accept

Table 1. Wilcoxon statistical test.

7. Conclusions

In this chapter, we have described and tested a new approach to solving the SAT problem based on combining the multilevel paradigm with the GSAT greedy algorithm. The resulting MLVGSAT algorithm progressively coarsens the problem, provides an initial assignment at the coarsest level, and then iteratively refines it backward level by level. In order to get a comprehensive picture of the new algorithm's performance, we used a benchmark set consisting of SAT-encoded problems from various domains. Based on the analysis of the results, we observed that within the same computational time, MLVGSAT provides higher quality solution compared with that of GSAT. Other conclusions that we may draw from the results are that the multilevel paradigm can either speed up GSAT or even improve its asymptotic convergence. Results indicated that the larger the instance, the higher the difference between the mean percentage excess deviation from the solution. An obvious subject for further work would be the use of efficient data structures in order to minimize the overhead during the coarsening and refinement phases. It would be of great interest to further validate or contradict the conclusions of this work by extending the range of problem classes. Finally, obvious subjects for further work include designing different coarsening strategies and tuning the refinement process.

8. References

[1] S.T. Barnard and H.D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6(2):101-17, 1994.

[2] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268-308, 2003.

[3] S.A. Cook. The complexity of theorem-proving procedures. *Proceedings of the Third ACM Symposium on Theory of Computing*, pages 151-158, 1971.

[4] M. Davis and H.Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201-215, 1960.

[5] R. Hadany and D. Harel. A Multilevel-Scale Algorithm for Drawing Graphs Nicely. *Tech.Rep.CS99-01*, Weizmann Inst.Sci, Faculty Maths.Comp.Sci, 1999.

[6] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. In S. Karin, editor, *Proc.Supercomputing'95, San Diego*, 1995. ACM Press, New York.

[7] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J.Sci. Comput.*, 20(1):359-392, 1998.

[8] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *J.Par.Dist.Comput.*, 48(1):96-129, 1998.

[9] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. *Proceedings of AAA'92*, pages 440-446. MIT Press, 1992.

[10] B. Selman, Henry A. Kautz, and B. Cohen. Noise strategies for improving local search. *Proceedings of AAAI'94*, pages 337-343. MIT Press, 1994.

[11] B. Selman and H.K. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In R. Bajcsy, editor, *Proceedings of the international Joint Conference on Artificial Intelligence*, volume 1, pages 290-295. Morgan Kaufmann Publishers Inc., 1993.

[12] D. McAllester, B. Selman, and H. Kautz. Evidence for invariants in local search. *Proceedings of AAAI'97*, pages 321-326. MIT Press, 1997.

[13] F. Glover. Tabu search – Part I. *ORSA Journal on Computing*, 1(3):190-206, 1989.

[14] P. Hansen and B. Jaumand. Algorithms for the maximum satisfiability problem. *Computing*, 44:279-303, 1990.

[15] I. Gent and T. Walsh. Unsatisfied variables in local search. In J. Hallam, editor, *Hybrid Problems, Hybrid Solutions*, pages 73-85. IOS Press, 1995.

[16] L.P. Gent and T.Walsh. Towards an understanding of hill-climbing procedures for SAT. *Proceedings of AAAI'93*, pages 28-33. MIT Press, 1993.

[17] B. Cha and K. Iwama. Performance tests of local search algorithms using new types of random CNF formula. *Proceedings of IJCAI'95*, pages 304-309. Morgan Kaufmann Publishers, 1995.

[18] J. Frank. Learning short-term clause weights for GSAT. *Proceedings of IJCAI'97*, pages 384- 389. Morgan Kaufmann Publishers, 1997.

[19] W.M. Spears. Simulated Annealing for Hard Satisfiability Problems. *Technical Report*, Naval Research Laboratory, Washington D.C., 1993.

[20] A.E. Eiben and J.K. van der Hauw. Solving 3-SAT with adaptive genetic algorithms. *Proceedings of the 4th IEEE Conference on Evolutionary Computation*, pages 81-86. IEEE Press, 1997.

[21] D.S. Johnson and M.A. Trick, editors. Cliques, Coloring, and Satisfiability, *Volume 26 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.

[22] C. Walshaw and M. Cross. Mesh partitioning: A multilevel balancing and refinement algorithm. *SIAM J.Sci. Comput.*, 22(1):63-80, 2000.

4

A Multi-start Local Search Approach to the Multiple Container Loading Problem

Shigeyuki Takahara
Kagawa Prefectural Industrial Technology Center
Japan

1. Introduction

This paper is concerned with the multiple container loading problem with rectangular boxes of different sizes and one or more containers, which means that the boxes should be loaded into the containers so that the waste space in the containers are minimized or the total value of the boxes loaded into the containers is maximized. Several approaches have been taken to solve this problem (Ivancic et al., 1989; Mohanty et al., 1994; George, 1996; Bortfeldt, 2000; Eley, 2002; Eley, 2003; Takahara, 2005; Takahara, 2006). The aim of this paper is to propose an efficient greedy approach for the multiple container loading problem and to contribute to develop a load planning software and applications.

The problem considered in this paper is the three-dimensional packing problem, therefore it is known to NP-hard. This implies that no simple algorithm has been found so far. In this paper this problem is solved by a relatively simple method using a two-stage strategy. Namely, all boxes are numbered and for the sequence of the numbers a greedy algorithm of loading boxes into containers is considered. This greedy algorithm is based on *first-fit concept* (Johnson et al., 1974) and determines the arrangement of each box. This algorith try to load the boxes all kind of containers and select the best arrangement result. It's requires a box loading sequence and a set of orientation orders of each box type for each container type.

Each box is tried to load according to these dynamic parameters. Moreover, a static parameter overhang ratio is introduced here. This is a percentage of the bottom face area of the loading box that isn't supported with other boxes that are below. As shown by Takahara (Takahara, 2006), if this parameter is different, the arrangement given this algorithm is different in spite of using a same pair of the loading sequence and the orientation orders. Some initial pairs of solution, that is a box loading sequence and a set of orientation orders of each box type, are selected among some rules, which are given beforehand. This solution is altered by using a multi-start local search approach. The local search approach is used to find a good pair of solutions that is brought an efficient arrangement by the greedy loading algorithm. The arrangement obtained in the iterations is estimated by volume utilization or total value of the loaded boxes.

The effectiveness of the proposed approach is shown by comparing the results obtained with the approaches presented by using benchmark problems from the literature. Finally, the layout examples by the application using the proposed approach for container loading are illustrated.

2. Problem description

The multiple container loading problem discussed in this paper is to load a given set of several boxes of varying size in one or more different containers so as to minimize the total volume of required containers or to maximize the total value of loaded boxes. This problem includes two kinds of problem, one is the problem with one container type and the other is the problem with different container types.

Before presenting the problem formulation, some notations used this paper are defined.

Let n be the number of types of boxes and let P = {1,..., n} . The box, the volume, the value per unit volume and the number of a box of type i , $i \in P$, are denoted by b_i, v_i , c_i and m_i, respectively. Each box of type corresponds to integer i , $i \in P$, and all permutations of the numbers ρ ={ $\sigma : \sigma$ (b_1,..., b_i,..., b_n), $i \in P$} are considered. The number of boxes is denoted by T , i.e. $T = \sum_{i=1}^{n} m_i$.

Let N be the number of types of containers and let Q = {1,..., N}. The container, the volume and the number of a container of type h , $h \in Q$, are denoted by C_h, V_h and M_h Thus, if N =1, then this problem is the one container type problem.

The boxes can be rotated. In consequence, up to six different orientations are allowed. Hence the rotation variants of a box are indexed from 1 to 6, an orientation order of each box type i to load to the container type h is denoted by $r_i^h = (u_{i1}^h,...,u_{i6}^h)$. If an edge (i.e. length, width or height) placed upright, the box of type i can take two orientations among them. Each box of type i is tried to load to the container of type h according to this orientation order r_i^h. Here, a set of orientation orders $\lambda = \{\mu : \mu = (r_1^h,...,r_n^h), h \in Q\}$ is considered.

For each $\sigma \in \rho$ and $\mu \in \lambda$, an algorithm that will be described below is applied, and loading positions of boxes are determined. The optimal solution is found by changing this permutation and this set of orders.

Practical constraints which have to be taken into account are load stability, weigh distribution, load bearing strength of boxes and so on. In this paper, in order to consider the load stability and vary the arrangement, overhang parameter γ is introduced. This is a percentage of the bottom face area of the loading box that isn't supported with other boxes that are below. Therefore, a box isn't allowed to be loaded to the position in which this parameter isn't satisfied. Generally, γ takes the value from 0% to 50%. Suppose that other constraints aren't taken into account here.

A loading algorithm A and a criterion F must be prepared for solving this multiple container loading problem. A is greedy algorithm and determines the arrangement of loading position of each box X according to the sequence σ and the set of orientation orders μ within the range of γ . This algorithm try to load the boxes to all kind of containers and select the container provided the best result. When all boxes are loaded by this algorithm and the arrangement is determined, the criterion F can be calculated. The arrangement is estimated by volume utilization and the total value of the loaded boxes. If the result required the number of container of type h is M_h' and the number of the loaded box of type i is m_i', the volume utilization is represented by

$$U = \sum_{i=1}^{n} v_i m_i' \Big/ \sum_{h=1}^{N} V_h M_h' \tag{1}$$

Moreover, the total value of the boxes loaded into the containers is represented by

$$W = \sum_{i=1}^{n} v_i c_i \tag{2}$$

In this paper, the objective is to maximize the volume utilization and the total value of loaded boxes. Therefore, the criterion F is denoted by

$$F(\sigma, \mu, \gamma) = \alpha U + \beta W \tag{3}$$

where α, β are constant number that depend on the problem. The optimality implies that this factor is maximized. Thus, the multiple container loading problem is denoted by

$$\max_{\sigma \in \rho, \mu \in \lambda, 0 \le \gamma \le 50} F(\sigma, \mu, \gamma) \tag{4}$$

When the calculation method for $F(\sigma, \mu, \gamma)$ is specified, the multiple container loading problem can be formulated as above combinatorial optimization problem. It naturally is adequate to use local search approach to obtain the optimal pair of the box sequence σ and the set of orientation orders μ . It is supposed that the overhang parameter γ is given before simulation. It should be noticed that the greedy loading algorithm and the multi-start local search can separately be considered.

3. Proposed approach

3.1 Greedy loading algorithm

"*Wall-building approach* (George & Robinson, 1980)" and "*Stack-building approach* (Gilmore & Gomory, 1965)" are well-known as heuristic procedure for loading boxes in container.

In this paper, *first-fit concept* is used as basic idea. Therefore, the proposed greedy loading algorithm consecutively loads the type of boxes, starting from b_1 to the last b_n. Since the total number of boxes is T, the box $b'_k(k=1,..,T)$ is loaded to the position p_k in the container. Therefore, the arrangement of boxes X is denoted by

$$X(\sigma, \mu, \gamma) = \{p_k : k = 1,...,T\} \tag{5}$$

If the box $b'_k(k=1,..,T)$ is loaded in the container C^i_h, $h \in Q$, $i = 1,..., M_h$,that means $p_h \in C^i_h$. The loading position p_k is selected from a set of potential loading areas S_k, which consists of the container floor and the top surface of the boxes that have been already loaded. Therefore, the set of potential loading areas here is

$$S_k = (s_1,...,s_{g_k}) \tag{6}$$

where g_k is the number of potential loading areas after loading k-1 boxes. The potential loading area is defined by a base point position, the length and the width. The base point is a point near the lower far left corner of the container. For example, in case of k =1, the number of potential loading areas is 1 and s_1 is the container floor.

In order to solve this multiple container loading problem, the method can be divided by two parts. The first part is single container part, and the other part is different container part.

The single container part algorithm is to solve single container loading problem.

The single container part algorithm SCA uses the following steps.

[Algorithm SCA]

Step 1: Set the sequence $\sigma = (b_1,..., b_n)$, the set of orientation orders $\mu = (r_1^h,...,r_n^h)$ and the overhang parameter γ for loading boxes;

decide the set of initial potential loading area S_1;
set $i = 1$, $k = 1$, and $j = 1$ as index for the current box type, the current box, and the current orientation, respectively.

Step 2: If $i \leq n$, then take the box of type b_i for loading, else stop.
Step 3: If all boxes of type b_i are already loaded, then set $i = i + 1$, $j = 1$ and go to Step 2.
Step 4: Scan the set of loading area S_k and find the loading position.
If a position that can be loaded is not found, then go to Step 6.
Step 5: Load the box on the selected position by Step 4.
Set $k = k + 1$ and update the set of loading area S_k.
If the boxes of type b_i are remained, then go to Step 2, else go to Step 7.
Step 6: If $j < 6$, then $j = j + 1$ and go to Step 4.
Step 7: If $k \leq T$, then set $i = i + 1$, $j = 1$ and go to Step 2, else stop.

This algorithm uses the orientation order of each box. The box of type b_i is arranged in the orientation of $\mu_{b_i j}$ in Step 3. If the current orientation is not permitted, skip Step 3 and go to Step 5 to take next orientation that is determined by r_{b_i}. Each box has a reference point in any of the six orientations. This point is set to the lower far left corner of the box and is used for scanning the potential loading areas in Step 4. A position that the box is loaded means a position in which the reference point of the box is put. For scanning each potential loading area, the reference point is set on the base point and is moved to the direction of the width and the length. The potential loading area is examined by the order of S_k, that is, from s_1 to s_{g_k}. The position in which the box can be loaded is where it doesn't overlap with other boxes that are already loaded and the overhang parameter of this box is γ or less.

The update procedure of the set of potential loading area in Step 5 is as follows:

1. Update current area
   If the box is loaded to the base point of the selected loading area, the area is deleted. When it is loaded to other positions, the area changes the dimension of the length or the width, and remains.
2. Create new loading area
   New loading areas, that is, a top face of the box, a right area of the box and a front area of the box, are generated if the container space and the loading area exist.
3. Combine area
   If the height of the base point of new loading area is same as the height of the base point of an existing loading area, and the new area is adjacent to the existing area, they are combined into one area.
4. Sort area
   In order to determine the order by which the loading area is examined, the existing loading areas are rearranged.

Generally, lower and far position in the container is selected as the loading position at first. In this paper, therefore the loading areas are sorted into the far lower left order of the base point position. Namely, the loading priority is given in the order of a far position, a lower position and a left position.

This algorithm is denoted by SCA(σ, μ, γ, h, i), where $h \in Q$, $i = 1,...,M_h$, and the arrangement result of SCA(σ, μ, γ, h, i) is denoted by R_h^i.

$$R_h^i = \{p_k : p_k \in C_h^i, k = 1,..., T\} \tag{7}$$

The criterion of this result R_h^i is represented by

$$F'(R_h^i) = \alpha \sum_{k=1}^{T} v_k' / V_h + \beta \sum_{k=1}^{T} v_k' c_k' \ (p_k \in C_h^i) \tag{8}$$

where v_k' is the volume of the box b_k', and c_k' is the value of the box b_k'. This is a partial criterion value.

The different container part algorithm is to try to load the boxes to all kind of left containers so as to determine the type of containers that is used. Therefore the q -th best arrangement R^q is denoted by

$$F'(R^q) = \max_{h=1,...,N} F'(R_l^{M'_{hq}+1}) \ (M'_{hq} = 0,...,M_h - 1) \tag{9}$$

where M'_{hq} is the number of required containers of type h before he q -th arrangement. The different container part algorithm DCA uses the following steps.

[Algorithm DCA]

Step 1: Set the sequence σ = (b_1,..., b_n), the set of orientation orders $\mu = (r_1^h,...,r_n^h)$ and the overhang parameter γ for loading boxes;
set h = 1 and q =1 as index for the current container type and number of required containers.

Step 2: If $h \le N$, then take the container of type $h \in Q$, else go to Step 5 .

Step 3: If the containers of type h aren't remained, then set $h = h + 1$ and go to Step 2.

Step 4: Solve SCA(σ, μ, γ, h, M'_{lq} +1).
Set $h = h + 1$ and go to Step 2.

Step 5: Select the best result R^q .
If all boxes are loaded, then set $q = q_{\max}$ and stop,
else if $q < \sum_{i=1}^{N} M_i$ set h = 1, $q = q$ +1 and go to Step 2.
else stop.

This algorithm is denoted by DCA(σ, μ, γ) and determines the container that loads the boxes one by one. This is also greedy algorithm and $q_{\max}$ is the number of required containers of this multiple container loading problem. The final arrangement result is represented by

$$X(\sigma,\mu,\gamma) = \{R^q : q = 1,..,q_{\max}\} \tag{10}$$

Therefore, the criterion of the result X (σ, μ, γ) is F(σ, μ, γ).

3.2 Multi-start local search

The multi-start local search procedure is used to obtain an optimal pair of the box sequence σ and the set of orientation orders μ. This procedure has two phases. First phase is decision of initial solutions using heuristics, and second phase is optimization of this solution using local search. Let *msn* be the number of initial solution and let *lsn* be the iteration number in local search. This procedure follows the original local search for each initial solution without any interaction among them and the random function is different at each local search.

At first phase, a good pair as an initial solution is selected. Therefore a heuristic rule that is related to decision of the box sequence σ is prepared. Another solution μ is selected at random.

At second phase, an optimal pair is found by using local search. The neighborhood search is a fundamental concept of local search. For a given σ, the neighborhood is denoted by $N_1(\sigma)$. Here, for $\sigma = (b_1,...,b_n)$,

$$\mathrm{N}_1(\sigma) = \{\tau : \tau = (b_1,..,b_{k-1},b_l,b_{k+1},..,b_{l-1},b_k,b_{l+1},..,b_n), |l-k| \le nsb\} \tag{11}$$

for all combinations of $1 \le k,l \le n$, $k \ne l$ and nsb is a neighborhood size. That is, two numbers b_k ,b_l of a permutation σ are taken, and they are exchanged. If $nsb = 1$, this neighborhood only swap two adjacent numbers. For a given μ, the neighborhood is denoted by $N_2(\mu)$. Here, for $\mu = (r_1^h,..., r_n^h)$,

$$\begin{aligned} &\mathrm{N}_2(\mu) = \{\upsilon : \upsilon = (r_1^h,...,r_i^h,...,r_n^h), h \in Q\} \\ &r_i^h = (u_{i1}^h,..,u_{i(k-1)}^h,u_{il}^h,u_{i(k+1)}^h,...,u_{i(l-1)}^h,u_{ik}^h,u_{i(l+1)}^h,...,u_{i6}^h) \end{aligned} \tag{12}$$

for all combinations of $1 \le i \le n$, $1 \le h \le N$, $1 \le k,l \le 6$, $k \ne l$. Thus, one order r_i^h of a set of orders μ is taken; two numbers u_{ik}^h, u_{il}^h of the selected order are taken, and they are exchanged. In this neighborhood, suppose that a neighborhood size nsr is the number of the exchange operation.

The local search in the neighborhood consists of generating a fixed number of pair of solutions $\tau_1,...,\tau_{lsn}$ in $N_1(\sigma)$ and $\upsilon_1,...,\upsilon_{lsn}$ in $N_2(\mu)$, and finding the best pair of solution $\hat{\tau}$ and $\hat{\upsilon}$:

$$F(\hat{\tau},\hat{\upsilon},\gamma) = \max_{k=1,...,lsn} F(\tau_k,\upsilon_k,\gamma) \tag{13}$$

Assume that the pair of j -th initial solutions are denoted by σ_j, μ_j, γ_j, the optimal solution is found by

$$F(\hat{\sigma},\hat{\mu},\hat{\gamma}) = \max_{j=1,...,msn} F(\hat{\sigma}_j,\hat{\mu}_j,\gamma_j) \tag{14}$$

The method of multi-start local search MSLS are describes as follows.

[Algorithm MSLS]

Step 1: Set j =1, i =1.

Step 2: If $j \le msn$, then set random seed rs_j and decide the sequence σ_j^i, the set of orientation orders μ_j^i and the overhang parameter γ_j, else stop.

Step 3: Set $\sigma^* = \sigma_j^i$, $\mu^* = \mu_j^i$.

Step 4: If $i \le lsn$, then solve DCA(σ_j^i, μ_j^i, γ_j), else go to Step 7.

Step 5: If $F(\sigma^*, \mu^*, \gamma_j) < F(\sigma_j^i, \mu_j^i, \gamma_j)$, then set $\sigma^* = \sigma_j^i$, $\mu^* = \mu_j^i$.

Step 6: Set $i = i + 1$and select $\sigma_j^i \in N_1(\sigma^*)$ and $\mu_j^i \in N_2(\mu^*)$.
Go back to Step4.

Step 7: If j =1, then set $\hat{\sigma} = \sigma^*$, $\hat{\mu} = \mu^*$ and $\hat{\gamma} = \gamma_j$;
else if $F(\hat{\sigma},\hat{\mu},\hat{\gamma}) < F(\sigma^*, \mu^*, \gamma_j)$, then $\hat{\sigma} = \sigma^*$, $\hat{\mu} = \mu^*$ and $\hat{\gamma} = \gamma_j$.

Step 8: Set $j = j + 1$, i =1 and go to Step 2.

In this algorithm, σ_j^1 and μ_j^1 (j =1,...,msn) are initial solutions and optimal arrangement result is represented by $X(\hat{\sigma},\hat{\mu},\hat{\gamma})$. The j-th local search uses the random seed rs_j to

determine the initial solution and the next solution in the neighborhood. The move strategy of this local search is first admissible move strategy as shown by this algorithm.

4. Computational experiments

4.1 Configuration

The tested method of multi-start local search procedure is shown below. At first phase, only one rule is required. The following rules are used to decide an initial solution of σ_j^1.

(R1) Sorting in decreasing order of the box volume v_i, $i \in P$.
(R2) Sorting in decreasing order of the box value c_i, $i \in P$.

These rules are used properly by problem type. If the object of the problem is to maximize the volume utilization, the rule R1 is used, and if the object of the problem is to maximize the total value of the loaded boxes, the rule R2 is used. Therefore the initial solution of box sequence is all same at any j, $j = 1,..., msn$.
Another initial solution of μ_j^1, which is the order of orientation $r_i^h = (u_{i1}^h,...,u_{i6}^h)$ for each box of type i are selected using a random function initialized random seed rs_j. The random function used here is linear congruential method.
The static parameter overhang ratio γ is important factor as shown by Takahara (Takahara, 2006). However, γ is fixed here for the simplification.
In order to show the effectiveness of the approach above described, the test cases from the literature were taken. Three kinds of test cases, that is the bin packing type multiple container loading problem with one container type, the bin packing type multiple container loading problem with different container type and the knapsack type multiple container loading problem, are taken.
As the bin packing type multiple container loading problem with one container type, the 47 examples of Ivancic et al. (Ivancic et al., 1989) are used here. The objective of the problems of Ivancic et al. is to find a minimal number of required containers load all given boxes. Each problem consists of two to five box types where only one container type is available. The problems of Ivancic et al. are denoted by IMM01 to IMM47.
As the bin packing type multiple container loading problem with different container type, the 17 examples of Ivancic et al. (Ivancic et al., 1989) are used here. The objective of these problems is to find the arrangement to maximize the volume utilization to load all given boxes. The problems of these are denoted by IMM2-01 to IMM2-17.
In these two bin packing type problems, the constant numbers are $\alpha = 1$ and $\beta = 0$ in the criterion (3) and the rule R1 is used to determine the initial solution of box sequence.
As the knapsack type multiple container loading problem, the 16 examples of Mohanty et al. (Mohanty et al., 1994) are used here. The objective of this problem is to maximize the total value of the loaded boxes. The problems are denoted by MMI01 to MMI16. In this knapsack type problem, the constant numbers are $\alpha = 0$ and $\beta = 1$ in the criterion (3) and the rule R2 is used to determine the initial solution of box sequence.
The algorithm was implemented in C using MS Visual C++ 6.0. The results of this approach were calculated on Xeon PC with a frequency 3.0GHz and 3GB memory.

4.2 Comparison with other methods

In order to show effectiveness of the approach above described, this approach MSLS has compared with other approaches. The following approaches were included the results of three kinds of test problems:

- IV_1989, a heuristic approach (Ivancic et al., 1989);
- MO_1994, a heuristic approach (Mohanty et al., 1994);
- B&R_1995, a heuristic approach (Bischoff & Ratcliff, 1995);
- BO_2000, a heuristic approach (Bortfeldt, 2000);
- EL_2002, a tree search approach (Eley, 2002);
- EL_2003, a bottleneck approach (Eley, 2003);
- TA_2006, a meta-heuristic approach (Takahara, 2006);

Table 1 presents the results for the 47 IMM problem classes. The parameters that were used in MSLS are $\gamma = 50$, $msn = 5$, $lsn = 500$, $nsb = 1$ and $nsr = 3$. The last column shows the total number of required containers. The results show that MSLS could be obtained the minimum number of total required containers than any other approach. But the result of the proposed approach is same as the result of the SA-Combined of TA_2006. This is because the greedy loading algorithm SCA is almost same as the loading algorithm that uses in TA_2006. However, the performance of computational time of MSLS has been improved from that of TA_2006 by 50%.

Table 2 shows the results for the 17 examples of three dimensional bin packing problem with different container types. The parameters that were used in MSLS are $\gamma = 50$, $msn = 10$, $lsn = 1000$, $nsb = 1$ and $nsr = 3$. The proposed approach obtained second highest average of volume utilization. For the test cases of IMM2-04 and IMM2-17, best arrangements ware found among these four methods. Fig.1 shows the best results found by the proposed approach.

Table 3 shows the results for the 16 MMI examples of knapsack type problem. The parameters that were used in MSLS are $\gamma = 50$, $msn = 10$, $lsn = 1000$, $nsb = 2$ and $nsr = 6$. The proposed approach obtained third highest average of volume utilization. For the test cases of MMI08 and MMI15, best arrangements ware found among these four methods. Fig.2 shows the best results found by the proposed approach.

4.3 Effect of neighborhood size

The neighborhood sizes, that is *nsb* and *nsr* , are key parameters of the proposed approach. *nsb* is the box sequence element ranges that can be exchanged. *nsr* is the number of iterations in which the orientation order elements are exchanged. In order to show the effect of these parameters, the following experiments have been done. The other parameters that were used here are $\gamma = 50$, $msn = 10$, $lsn = 1000$. Table 4 shows the Ivancic et al. with different container types test problems results. The value in this table is volume utilization. If the neighborhood size *nsb* grows, the effect of the neighborhood size *nsr* becomes small. In the case of $nsr = 3$ or $nsr = 4$, the better results are obtained. Table 5 shows Mohanty et al. test problems results. The value is the percentage of bounds. If the neighborhood size *nsb* becomes small, the effect of the neighborhood size *nsr* becomes small. In the case of $nsr = 5$ or $nsr = 6$, the better results are obtained.

	IV_1989	B&R_1995	BO_2000	EL_2002	EL_2003	TA_2006	MSLS
	Number of Containers	Number of Containers	Number of Containers	Number of Containers	Number of Containers	Number of Containers	Number of Containers
IMM01	26	27	25	26	25	25	25
IMM02	11	11	10	10	10	10	10
IMM03	20	21	20	22	20	20	20
IMM04	27	29	28	30	26	26	26
IMM05	65	61	51	51	51	51	51
IMM06	10	10	10	10	10	10	10
IMM07	16	16	16	16	16	16	16
IMM08	5	4	4	4	4	4	4
IMM09	19	19	19	19	19	19	19
IMM10	55	55	55	55	55	55	55
IMM11	18	19	18	18	17	16	16
IMM12	55	55	53	53	53	53	53
IMM13	27	25	25	25	25	25	25
IMM14	28	27	28	27	27	27	27
IMM15	11	11	11	12	11	11	11
IMM16	34	28	26	26	26	26	26
IMM17	8	8	7	7	7	7	7
IMM18	3	3	2	1	2	2	2
IMM19	3	3	3	2	3	3	3
IMM20	5	5	5	2	5	5	5
IMM21	24	24	21	26	20	20	20
IMM22	10	11	9	9	8	9	9
IMM23	21	22	20	21	20	20	20
IMM24	6	6	6	6	6	5	5
IMM25	6	5	5	5	5	5	5
IMM26	3	3	3	3	3	3	3
IMM27	5	5	5	5	5	5	5
IMM28	10	11	10	10	10	10	10
IMM29	18	17	17	18	17	17	17
IMM30	24	24	22	23	22	22	22
IMM31	13	13	13	14	13	13	13
IMM32	5	4	4	4	4	4	4
IMM33	5	5	5	5	5	5	5
IMM34	9	9	8	9	8	8	8
IMM35	3	3	2	2	2	2	2
IMM36	18	19	14	14	14	14	14
IMM37	26	27	23	23	23	23	23
IMM38	50	56	45	45	45	45	45
IMM39	16	16	15	15	15	15	15
IMM40	9	10	9	9	8	9	9
IMM41	16	16	15	15	15	15	15
IMM42	4	5	4	4	4	4	4
IMM43	3	3	3	3	3	3	3
IMM44	4	4	3	4	4	3	3
IMM45	3	3	3	3	3	3	3
IMM46	2	2	2	2	2	2	2
IMM47	4	3	3	3	3	3	3
Total	763	763	705	716	699	698	698

Table 1. Results obtained for the problems from Ivancic et al. with one container type

	IV_1989		BO_2000		EL_2003		MSLS	
	Volume utilization [%]	Number of containers for every type	Volume utilization [%]	Number of containers for every type	Volume utilization [%]	Number of containers for every type	Volume utilization [%]	Number of containers for every type
IMM2-01	71.8	26/0	74.7	25/0	74.7	25/0	74.7	25/0
IMM2-02	87.3	2/5	87.3	2/5	88.5	1/7	88.5	1/7
IMM2-03	74.3	1/8	92.2	2/0	92.2	2/0	92.2	2/0
IMM2-04	84.1	1/14	89	0/15	89	0/15	89.5	2/11
IMM2-05	97.6	7/13/6	95.1	1/19/11	99.9	2/13/17	99.7	7/15/1
IMM2-06	97.6	4/6/1	99.7	4/1/2	99.7	7/4/0	99.6	7/1/0
IMM2-07	85.8	10/1/7	86.8	16/0/2	87.4	2/0/14	87.1	9/0/8
IMM2-08	95.8	3/0/26	97.9	7/0/23	99.4	3/0/25	98.7	1/1/25
IMM2-09	92.2	7/6/1	96.6	8/5/0	96.6	6/9/0	96.6	7/7/0
IMM2-10	90.6	1/0/2	90.6	1/0/2	90.6	1/0/2	90.6	1/0/2
IMM2-11	81.2	3/3/11	85.9	0/4/10	88.4	9/2/5	87.7	1/6/4
IMM2-12	75	5/1/0	90.2	2/1/1	93.5	3/0/1	92.7	5/0/0
IMM2-13	85.3	9/1/5	87.8	14/1/1	86.3	13/1/2	85	5/11/2
IMM2-14	88.7	0/2/4	94	2/1/1	94	2/1/1	94	2/1/1
IMM2-15	76.3	3/2/11	79.1	3/3/10	79.2	2/3/11	78.7	3/1/11
IMM2-16	82.7	4/0/0	91.6	2/1/0	91.6	2/1/0	82.9	1/1/1
IMM2-17	77.1	26/0	84.7	0/0/3	84.7	0/0/3	91.6	0/1/1
Average	84.9		89.6		90.3		90.0	

Table 2. Results obtained for the problems from Ivancic et al. with different container types

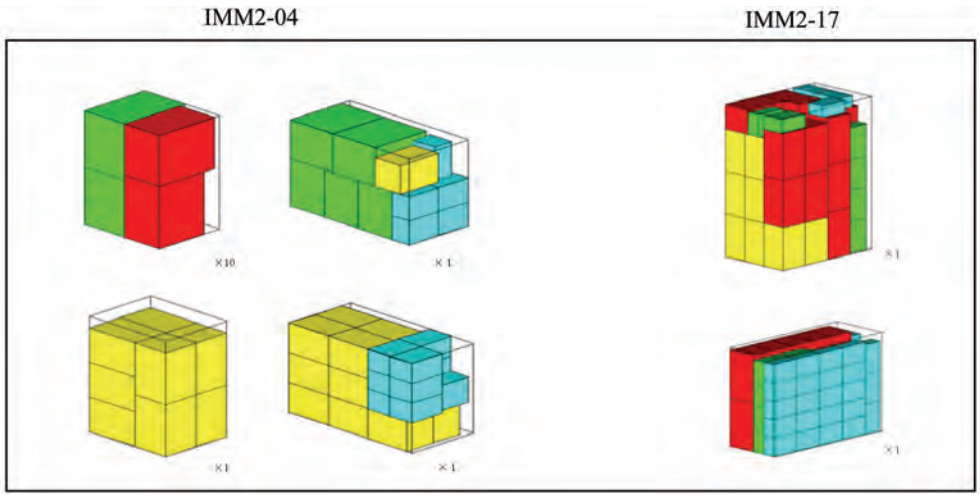


Fig. 1. Layout results of Ivancic et al. with different container types test problems

	Bound	MO_1994		BO_2000		EL_2003		MSLS	
		Absolute	In % of bound	Absolute	In % of bound	Absolute	In % of bound	Absolute	In % of bound
MMI01	11112.0	8640.0	77.7	8640.0	77.7	8640.0	77.7	8640.0	77.7
MMI02	86016.0	83494.4	97.1	85120.0	99.0	85376.0	99.3	84224.0	97.9
MMI03	53500.0	53262.5	99.6	53262.5	99.6	53262.5	99.6	52350.0	97.9
MMI04	2720640.0	2333440.0	85.8	2333440.0	85.8	2307840.0	84.8	23334400.0	85.8
MMI05	653750.0	495500.0	75.8	581250.0	88.9	583750.0	89.3	579250.0	88.6
MMI06	143424.0	138240.0	96.4	139584.0	97.3	141216.0	98.5	137952.0	96.2
MMI07	20203.2	16668.0	82.5	17409.0	86.2	17004.0	84.2	17262.0	85.4
MMI08	77986.8	65741.0	84.3	68645.6	88.0	69121.2	88.6	69747.2	89.4
MMI09	139356.0	119772.0	85.9	128952.0	92.5	133632.0	95.9	128556.0	92.3
MMI10	15360.0	15360.0	100.0	15360.0	100.0	15360.0	100.0	15360.0	100.0
IMMI11	68353.2	49995.0	73.1	53202.8	77.8	52873.6	77.4	53202.8	77.8
MMI12	24964.0	23529.0	94.3	24235.2	97.1	23673.0	94.8	23990.4	96.1
MMI13	36556.8	36556.8	100.0	36556.8	100.0	36556.8	100.0	36556.8	100.0
MMI14	71552.0	56492.8	78.9	65316.8	91.3	68723.2	96.0	68723.2	96.0
MMI15	42922.8	37558.8	87.5	39727.2	92.6	39382.2	91.8	40590.0	94.6
MMI16	666829.6	556458.0	83.5	595770.0	89.3	591535.0	88.7	571290.0	85.7
Average			87.7		91.4		91.7		91.3

Table 3. Results obtained for the problems from Mohanty et al.

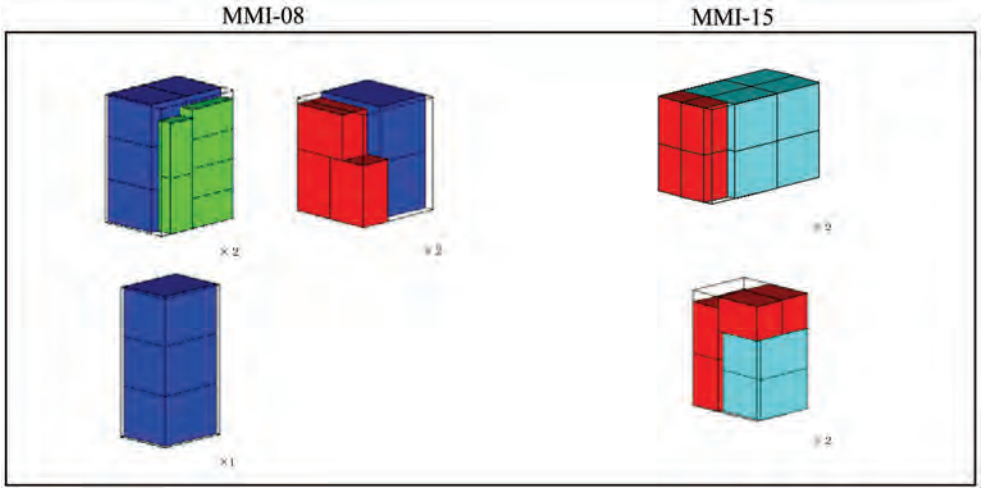


Fig. 2. Layout results of Mohanty et al. test problems

	nsb = 1 nsr = 2	nsb = 1 nsr = 4	nsb = 1 nsr = 8	nsb = 2 nsr = 2	nsb = 2 nsr = 4	nsb = 2 nsr = 8	nsb = 3 nsr = 2	nsb = 3 nsr = 4	nsb = 3 nsr = 8
IMM2-01	74.67	74.67	74.67	74.67	74.67	74.67	74.67	74.67	74.67
IMM2-02	87.34	87.34	87.34	86.18	86.18	87.34	86.18	87.16	87.34
IMM2-03	92.16	92.16	92.16	92.16	92.16	92.16	92.16	92.16	92.16
IMM2-04	89.48	89.48	89.48	89.72	89.72	89.48	89.72	89.72	84.07
IMM2-05	99.73	99.73	99.73	99.73	99.73	99.73	99.73	99.73	99.73
IMM2-06	99.65	99.65	99.65	99.65	99.65	99.65	99.65	99.65	99.65
IMM2-07	87.09	87.09	87.09	87.09	87.09	87.09	87.09	87.09	87.09
IMM2-08	98.68	98.68	98.68	98.68	98.68	98.68	98.68	98.68	98.68
IMM2-09	96.63	96.63	93.30	96.63	96.06	96.63	96.63	96.06	96.63
IMM2-10	90.57	90.57	90.57	90.57	90.57	90.57	90.57	90.57	90.57
IMM2-11	86.35	87.73	88.35	86.35	87.73	86.35	86.28	86.35	87.73
IMM2-12	90.98	90.98	90.98	92.73	92.73	90.98	90.23	90.98	92.73
IMM2-13	85.96	86.29	85.84	86.29	85.02	85.88	85.84	85.88	86.29
IMM2-14	94.03	94.03	94.03	94.03	94.03	94.03	94.03	94.03	94.03
IMM2-15	78.75	78.75	78.75	78.75	78.75	78.75	78.75	78.75	78.75
IMM2-16	82.87	82.87	82.87	82.87	82.87	82.87	82.87	82.87	82.87
IMM2-17	91.59	91.59	91.59	91.59	91.59	91.59	91.59	91.59	91.59
Average	89.79	89.89	89.71	89.86	89.83	89.79	89.68	89.70	89.68

Table 4. Ivancic et al. with different container types test problems results for different neighborhood size

	nsb = 1 nsr = 2	nsb = 1 nsr = 6	nsb = 1 nsr = 8	nsb = 2 nsr = 2	nsb = 2 nsr = 6	nsb = 2 nsr = 8	nsb = 3 nsr = 2	nsb = 3 nsr = 6	nsb = 3 nsr = 8
MMI01	77.75	77.75	77.75	77.75	77.75	77.75	77.75	77.75	77.75
MMI02	97.92	97.92	97.92	97.92	97.92	97.92	97.92	97.92	97.92
MMI03	97.85	97.85	97.85	97.85	97.85	97.85	97.85	97.85	97.85
MMI04	85.77	85.77	85.77	85.77	85.77	85.77	85.77	85.77	85.77
MMI05	87.23	88.60	86.77	87.23	88.60	88.60	87.23	88.60	88.60
MMI06	96.18	96.52	96.85	96.18	96.18	96.18	96.18	96.18	96.18
MMI07	85.44	85.44	85.44	85.44	85.44	85.44	85.44	85.44	85.44
MMI08	89.43	89.43	89.43	89.43	89.43	89.43	89.43	89.43	89.43
MMI09	92.59	91.91	92.25	91.91	92.25	92.10	92.59	91.91	92.25
MMI10	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
MMI11	77.84	77.84	77.84	77.84	77.84	77.84	77.84	77.84	77.84
MMI12	93.20	93.62	94.89	92.64	96.10	93.91	92.64	96.10	93.91
MMI13	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
MMI14	96.05	96.05	94.60	96.05	96.05	96.05	96.05	96.05	96.05
MMI15	94.57	94.57	94.57	94.57	94.57	94.57	94.57	94.57	94.57
MMI16	85.67	85.67	85.67	85.67	85.67	85.67	85.67	85.67	85.67
Average	91.09	91.18	91.10	91.02	91.34	91.19	91.06	91.32	91.20

Table 5. Mohanty et al. test problems results for different neighborhood size

5. Conclusion

The presented multi-start local search approach for the multiple container loading problem is suitable for solving various kind of problem, because the proposed approach is based on greedy loading algorithm and hardly uses problem-specific operators and heuristic rules. Hence it is easy to improve and manage by users. Its good performance and superiority compared with the other approaches were shown in the test results.

In this paper, only the weakly heterogeneous problems are taken. Thus, further studies include integration of greedy loading algorithm and multi-start approach, dealing with the strongly heterogeneous problems, and development of an efficient container loading software.

6. References

Bischoff, E. E. & Ratcliff, M. S. W. (1995). Issues in the development of approaches to container loading, *Omega*, 23, pp.377-390.

Bortfeldt, A. (2000). Eine heuristik für multiple containerladeprobleme, *OR Spektrum*, 22 pp.239-262.

Eley, M. (2002). Solving container loading problems by block arrangement, *EJOR*, 141, pp.393-402.

Eley, M. (2003). A bottleneck assignment approach to the multiple container loading problem, *OR Spectrum*, 25, pp.45-60.

George, J. A. & D. F. Robinson, D. F. (1980). A heuristic for packing boxes into a container, *Computers Opns Res.*, 7, pp.147-156.

George, J. A. (1996). Multiple container packing: a case study of pipe packing, *Journal of the Operational Research Society*, 47, pp.1098-1109.

Gilmore, P. C. & Gomory, R. E. (1965). Multistage cutting stock problems of two and more dimensions, *Opns Res.*, 13, pp.94-120.

Ivancic, N.J., Mathur, K. & Mohanty, B.B. (1989). An integer-programming based heuristic approach to the three dimensional packing problem, *Journal of Manufacturing and Operation Management*, 2, pp.268-298.

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R. & Graham, R. L. (1974). Worst-case performance bounds for simple one-dimensional packing algorithms, *SIAM J. Comput.*, 3, pp.299-325.

Mohanty, B. B., Mathur, K. & Ivancic, N.J. (1994). Value considerations in three-dimensional packing – A heuristic procedure using the fractional knapsack problem, *EJOR*, 74, pp.143-151.

Takahara, S. (2005). Loading problem in multiple containers and pallets using strategic search method, *Lecture Notes in Artificial Intelligence 3558*, pp.448-456.

Takahara, S. (2006). A simple meta-heuristic approach for the multiple container loading problem, *Proceedings of 2006 IEEE International Conference on Systems, Man and Cybernetics*, pp.2328-2333, Taipei, Taiwan.

5

A Partition-Based Suffix Tree Construction and Its Applications

Hongwei Huo[1] and Vojislav Stojkovic[2]
[1] *School of Computer Science and Technology, Xidian University, Xi'an*
[2] *Computer Science Department, Morgan State University, Baltimore*
[1]*China*
[2]*USA*

1. Introduction

A suffix tree (also called suffix trie, PAT tree or, position tree) is a powerful data structure that presents the suffixes of a given string in a way that allows a fast implementation of important string operations. The idea behind suffix trees is to assign to each symbol of a string an index corresponding to its position in the string. The first symbol in the string will have the index 1, the last symbol in the string will have the index n, where n = number of symbols in the string. These indexes instead of actual objects are used for the suffix tree construction. Suffix trees provide efficient access to all substrings of a string. They are used in string processing (such as string search, the longest repeated substring, the longest common substring, the longest palindrome, etc), text processing (such as editing, free-text search, etc), data compression, data clustering in search machines, etc.
Suffix trees are important and popular data structures for processing long DNA sequences. Suffix trees are often used for efficient solving a variety computational biology and/or bioinformatics problems (such as searching for patterns in DNA or protein sequences, exact and approximate sequence matching, repeat finding, anchor finding in genome alignment, etc).
A suffix tree displays the internal structure of a string in a deeper way. It can be constructed and represented in time and space proportional to the length of a sequence. A suffix tree requires affordable amount of memory. It can be fitted completely in the main memory of the present desktop computers. The linear construction time and space and the short search time are good features of suffix trees. They increase the importance of suffix trees. A suffix tree construction process is space demanding and may be a fatal in the case of a suffix tree to handle a huge number of long DNA sequences. Increasing the number of sequences to be handled, due to random access, causes degrades of the suffix tree construction process performance that uses suffix links. Thus, some approaches completely abandon the use of suffix link and give up the theoretically superior linear construction time for a quadratic time algorithm with better locality of reference.

2. Previous work

Weiner [1] gave the first linear time algorithm for suffix tree construction. McCreight [2] built a more space efficient algorithm for suffix tree construction in linear time. It has a

readable account for suffix tree construction while processing a string from right to left. Ukkonon [3] developed a conceptually different linear-time algorithm for suffix tree construction that includes all advantages of McCreight's algorithm but also allows a much simpler explanation. It is a left-to-right on-line algorithm. Ukkonon's algorithm maintains at each step a suffix tree for a string S, where S is $c_1 ... c_i ... c_n$, as the index i is increasing from 1 to n. Many improvements in suffix tree construction have been done during the last decades. The early algorithms for suffix tree construction have been focused on developing algorithms in linear space. These algorithms are adapted to a small input size and the entire-complete suffix tree can be constructed in the memory. Unfortunately, these algorithms are less space efficient, because they suffer from a poor locality of memory reference. Cache processor architectures have a hard job to store memory references in the secondary memory. One moment there are too many data to be loaded into the memory that causes the missing a lot of cache and more disk swapping. Thus, how to develop a practical algorithm for suffix tree construction is still an important problem.

Suffix trees are not only used in the substring processing problems. They are used also in the complex genome-scale computational problems. For example, MUMmer [4, 5] is a system for the genome alignment, which uses as its main structure suffix trees to align two closely relative genomes. Due to the advantages of suffix trees, MUMmer provides the faster, simpler, and more systematic way to solve the hard genome alignment problem. REPuter [6, 7] is another popular software tool for the efficient computing of exact repeats and palindromes in the entire genome. It uses an efficient and compact suffix tree to locate exact repeats in linear time and space.

Although suffix trees have these superior features, they are not widely used in the real string processing software. The main reason for that is that the space consumption of a suffix tree is still quite large despite the asymptotically linear space [3]. Therefore, several researchers/scientists have developed the alternative index structures, which store less information than suffix trees, but they are more space efficient [8]. The most known index structures are suffix arrays, level compressed tries, suffix binary search trees, [4]. Index structures have to be tailed for some string matching problems and cannot be adapted to other kinds of problems without loss of performance. Also, the traditional string methods cannot be directly used in the DNA sequences because they are too complex to be treated. The reducing the space requirement of suffix trees is still an important problem in the genome processing.

In order to overcome these disadvantages, we propose a new algorithm for suffix tree construction for DNA sequences based on the partitioning strategies and use of the common prefixes to construct the independent subtrees [9]. The experiments show that the proposed algorithm is more memory-efficient and it has a better performance on the average running time.

3. Suffix tree

3.1 Definition

Definition 1. A suffix tree for a string S of n-characters, where n>=1, is a tree with n leaves numbered from 0 to n-1. Each internal node, other than the root, has at least two children. Each edge has an edge-label that is a nonempty substring of the string S. All edges exit from a same node have edge-labels beginning with different characters.

The most important characteristic of a suffix tree for a string S is that for each leaf i, where 0 <= i <= n-1, the concatenation of edge-labels on the path from the root to the leaf i is the ith suffix of the string S. The ith suffix of a string S is the suffix of the string S that starts at the position i.

Suffix trees can be constructed in linear time and space [1~3]. Some suffix tree construction algorithms that use suffix links require $O(n)$ construction time, where n is the length of a string. A suffix link is a link from one internal node to another internal node. Often, leaves of a suffix tree are labeled by leaf-labels. A leaf-label is the starting position of the suffix that ends at this leaf.

The Fig. 1 shows the suffix tree for the string S = *ATTAGTACA*\$. The \$ character represents the end of the string S and it is count as the part of the string S. Dashed lines represent suffix links.

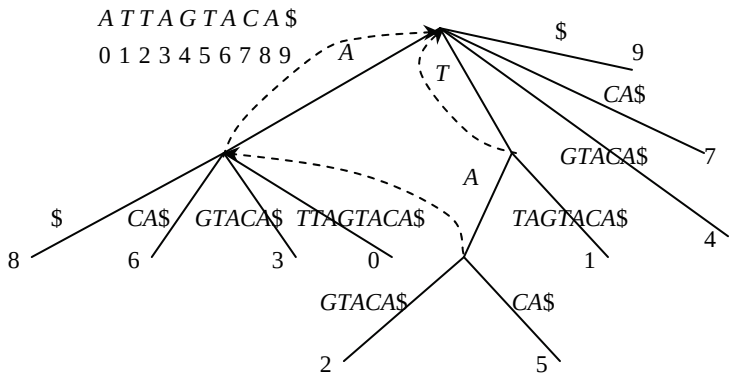


Fig. 1. The suffix tree for the string S = *ATTAGTACA*\$

3.2 Space requirements

The important characteristic of the suffix tree T for a string S, $T(S)$, is that $T(S)$ can be stored in $O(n)$ space, where n is the length of the string S.

The idea is the following:

- $T(S)$ has exactly n leaves, where n is the length of the string S.
- Since each internal node of $T(S)$ is a branching node, $T(S)$ has at most n internal nodes.
- Since in/at each node, except the root, enters/ends exactly one edge, $T(S)$ has at most $2n$ edges.
- Since each edge-label is a substring of S\$, it can be represented in constant space by a pair (start, end) points into S\$.

4. Partition-based suffix tree construction

4.1 Analysis

If a memory access mechanism has temporal and/or spatial locality features then the processor may use one or more caches to speed up access to the memory. Linear time, suffix tree construction algorithms, such as McCreight's algorithm [2] and Ukkonen's algorithm

[3], require many random accesses to the memory for suffix trees and links. In Ukkonen's algorithm, cache misses happen, when the algorithm makes a traversal via suffix links to reach another new subtree to check its children nodes. Such traversals cause random memory accesses at the very distant memory locations. In addition, each memory access visits memory with the higher probability because the address space span is too large to fit into the memory.

Kurtz's algorithm [8, 14], optimizes the space requirements for the McCreight's algorithm. Kurtz's algorithm divides the internal nodes into large nodes and small nodes to store the suffix tree information based on the relation of head position values. During the construction of internal nodes, there are many short or long small-large chains, which are sequences of small nodes followed by one large node. In a small-large chain, values of head position, the depth and suffix link of all small nodes can be derived from the large node at the end of chain. Therefore, with the bit optimization technique, Kurtz's algorithm uses four integers for one large node, two integers for one small node and one integer for each leaf node. Therefore, what a small-large chain is longer than more space is saved.

After analyzing, we find that a small-large chain is formed only if all nodes in the chain are series of new nodes created consecutively while series of suffixes are added into the suffix one by one.

DNA sequences are not only well known for their repetitive structures but also they are well known for their small-sized alphabet sequences that have high possibility of repetition. Therefore, applying Kurtz's algorithm on DNA sequences may not get advantage on small nodes but produces more large nodes.

4.2 Algorithm

Based on the properties of suffix trees, we can:

- in advance put together some suffixes of a branching node
- during the top-down suffix tree construction merge step by step the common prefixes of suffixes
- generate the internal branching nodes with the common prefix such as an edge-label and the responding leaf nodes
- finish the construction of the various branching nodes under the branch.

We propose the new ST-PTD (Suffix Tree Partition and Top-Down) algorithm for construction of a suffix tree for a DNA sequence. The ST-PTD algorithm uses partition and top-down techniques. Due to partition, a large input to the suffix tree construction is allowable. The construction of each subtree in the memory is independent.

The ST-PTD algorithm consists of two phases: partition suffixes and subtree construction. The algorithm is shown in Fig. 2.

Algorithm ST-PTD (*S*, *prefixlen*)
// Phase 1: Preprocessing
1. Scan the string *S* and partition suffixes based on the first *prefixlen* symbols of each suffix
// Phase 2. Construct suffix tree
2. **for** each partition P_i **do**
3. $R \leftarrow$ sorting(P_i)
4. **do**
5. **if** $|R| = 1$ **then**
6. create a leaf *l*

```
7.                $ST_i \leftarrow ST_i \cup \{l\}$
8.              else
9.                $lcp$ = finding-LCP($R$)
10.               create a branch node in the $ST_i$
11.               add the $X$ to $R$, $X$ being the set of remaining suffixes from $R$ after splitting
                  off the longest common prefix
12.               sorting($R$)
13.       while (!empty($R$))
14. Merge{$ST_i$}
```

Fig. 2. The ST-PTD algorithm

In the preprocessing step, the suffixes of the input string S is partitioned into $|\sum|^{prefixlen}$ parts, where $\sum$ is an alphabet and $|\sum|$ is the size of the alphabet $\sum$. In the case of DNA sequences $\sum$ = {A, C, G, T} and $|\sum|$ = 4. *prefixlen* is the depth of partitioning. The partition procedure is as follows. First, we scan the input string from left to right. At each index position *i*, the *prefixlen* subsequent characters are used to determine one of the $|\sum|^{prefixlen}$ partitions and the index *i* is then recorded to the calculated partition. At the end of the scan, each partition will contain the suffix pointers for suffixes that all have the same prefix of size *prefixlen*. In the case of DNA sequences, we can assume that the internal nodes close to the root are highly repetitive and have the small alphabet - we can take value of *prefixlen* to be the $\log_4$ (*Seqlen*–1). However, when the value of *prefixlen* is large than 7, the running time for partition phase for large dataset, such as genome, is costly and can not bring the obvious advantages to the algorithm, thus we take the value of *prefixlen* to be the ($\log_4$(*Seqlen*–1))/2. In the suffix tree construction step, for each partition, the algorithm performs an independent construction of the respective suffix tree branch. The algorithm does not need to start at the root of the suffix tree but directly in the node that is found at some depth.

4.3 Space requirements

The space requirement measures how many bytes one character uses on average.
We use the DNA sequences from the NCBI web site to compare the space requirement of the Kurtz's algorithm [8] with the space requirement of the ST-PTD algorithm. The numbers given in the Table 1 refer to the space required for the construction. They do not include the *n* bytes used to store the input string.

Name	Length	Kurtz's algorithm	The ST-PTD algorithm	Saving
AC008583	122493	12.62	11.79	0.0658
AC135393	38480	12.39	11.85	0.0436
BC044746	4897	12.61	11.72	0.0706
J03071	11427	12.32	13.68	-0.1104
M13438	2657	12.50	11.59	0.0728
M26434	56737	12.52	12.03	0.0391
M64239	94647	12.62	11.72	0.0713
V00662	16569	12.69	11.74	0.0749
X14112	152261	12.58	11.87	0.0564
ecoli	4668239	12.56	11.72	0.0669
[Average]	516841	12.541	11.971	0.0451

Table 1. The space requirements of Kurtz's algorithm and the ST-PTD algorithm

Table 1 shows the space requirement for each sequence.
- The first column of the Table 1 contains the names of DNA sequences.
- The second column of the Table 1 contains the lengths of DNA sequences.
- The third column of the Table 1 contains the space requirement of Kurtz' algorithm.
- The fourth column of the Table 1 contains the space requirement of the ST-PTD algorithm.
- The fifth column of the Table 1 contains the savings.

The ST-PTD algorithm compared with Kurtz's algorithm saves about 4.55% in space.
There is no relationship between space needs and the length of sequence. However, the DNA sequence, such as J03071, has a great effect on the space demand.

4.4 Running time

Kurtz's algorithm and the ST-PTD algorithm have been implemented in the C programming language and compiled with the GCC compiler. To learn and show the impact of the memory on the algorithms, we ran/executed the programs on two different platforms config1 and config2. Config1 consisted of the Intel Pentium 4.3 GHZ processor, 512M RAM, and the Red Hat Linux 9 operating system. Config2 consisted of the Intel Pentium III 1.3 GHHZ processor, 128 RAM, and the Fedora 4 operating system.
The experimental results are shown in Table 2. The running time is in seconds and throughout is the ratio of time multiplied by 10^6 and sequence length. The dark shaded areas show the better throughout. '-' shows the running time more than 1 hour.
We used in both algorithms arrays as the main data structures to get the higher efficiency in time. Unfortunately, arrays limit the size of data they deal with. However, we still used arrays, because Kurtz's algorithm in which we used linked lists to implement DNA sequences takes 1176.02 seconds (about 20 minutes) for the sequence B_anthracis_Mslice of 317k length and over four hours for the sequence ecoil of 4.6M length.
Although Kurtz's algorithm requires $O(n)$ time in the worst case and the ST-PTD algorithm requires $O(n^2)$ time, the ST-PTD algorithm is a little faster than Kurtz's algorithm on the average running time. This shows that the locality of memory reference has the great influence on the running time of both algorithms. The partition strategies and the sequence structure also had the impact on the performance of both algorithms. For example, the difference induced by the unbalanced partitions on the sequence influenza slice is obvious.
The ST-PTD algorithm has greater advantages on Kurtz's algorithm for the lower configuration due to its partition phase. The partition phase decreases the size of the set of problems we are processing so that we can deal with the larger size of data.
Comparing the running time of both algorithms in different configurations, we can see that memory is still one of the bottlenecks affecting the performances of both algorithms. Suffix trees are indeed very space greedy. In addition, compared with Kurtz's algorithm, the ST-PTD algorithm is easier to understand and implement. Also, the ST-PTD algorithm is easier to be parallelized because the construction of each subtree is independent.

5. Some applications of suffix trees

5.1 Exact string matching

The *exact string matching problem* is: Given a string/sequence S and a pattern string P. Find all positions of the pattern P in *the string S*.

		Config 1				Config 2			
		Kurtz's algorithm		The ST-PTD algorithm		Kurtz's algorithm		The ST-PTD algorithm	
Sequence	Length	time	tput	time	tput	time	tput	time	tput
J03071	11427	0.06	5.25	0.10	8.75	0.06	5.25	0.12	10.50
V00662	16569	0.01	0.60	0.02	1.21	0.01	0.60	0.02	1.21
AC135393	38480	0.2	5.20	0.94	24.43	0.26	6.76	1.54	40.02
M26434	56737	0.04	0.71	0.05	0.88	0.06	1.06	0.07	1.23
M64239	94647	0.07	0.74	0.08	0.85	0.11	1.16	0.12	1.27
AC008583	122493	0.09	0.73	0.11	0.90	0.14	1.14	0.15	1.22
X14112	152261	0.11	0.72	0.14	0.92	0.20	1.31	0.21	1.38
B_anthracis_Mslice	317829	0.34	1.07	0.31	0.98	0.46	1.45	0.45	1.42
H.sapiens chr.10 slice1	1119913	1.28	1.14	1.27	1.13	1.59	1.42	2.70	2.41
H.sapiens chr.10 slice2	2099930	2.62	1.25	2.53	1.20	3.41	1.62	5.32	2.53
H.sapiens chr.10 slice3	3149930	3.98	1.26	3.98	1.26	23.31	7.40	8.45	2.68
H.sapiens chr.10 slice4	4199930	5.56	1.32	5.13	1.22	-	-	11.74	2.80
ecoli	4668239	7.19	1.54	5.79	1.24	-	-	13.69	2.93
H.sapiens chr.10 slice5	4899930	6.25	1.28	6.08	1.24	-	-	14.08	2.87
H.sapiens chr.10 slice6	5250000	6.62	1.26	7.74	1.47	-	-	15.39	2.93
H.sapiens chr.10 slice7	5600000	7.03	1.26	7.04	1.26	-	-	16.61	2.97
influenza slice	5918744	5.16	0.87	46.07	7.78	-	-	71.15	12.02
H.sapiens chr.10 slice8	6019975	7.66	1.27	21.94	3.64	-	-	38.44	6.39
H.sapiens chr.10 slice9	6300000	8.2	1.30	7.92	1.26	-	-	18.78	2.98
H.sapiens chr.10 slice10	6999930	9.67	1.38	9.04	1.29	-	-	21.30	3.04
H.sapiens chr.10 slice11	8400000	10.71	1.28	11.52	1.37	-	-	26.55	3.16
H.sapiens chr.10 slice12	9100000	12.92	1.42	13.53	1.49	-	-	28.65	3.15
Arabidopsis thaliana chr. 4	9835812	44.01	4.47	30.33	3.08	-	-	-	-
H. sapiens chr. 10 slice13	10500000	79.13	7.54	25.89	2.47	-	-	-	-
[Average]		8.42	2.13	7.98	2.02				

Table 2. The running time and throughout of Kurtz's algorithm and ST-PTD

The exact string matching problem can be solved using the suffix tree on the following elegant way:

- Construct the suffix tree for the string S, $T(S)$.
- Traverse – top-down pass through $T(S)$ from the root further into $T(S)$, guided by the characters of P, as long as there is a continuation in $T(S)$ that corresponds to the letters of P.
- If this search stops before the end of P is reached, P does not occur in S.
- If P can be spelled out completely, then P occurs in S. Moreover, the numbers at the leaves below the end point of this search tell all the positions in S where P occurs.

Suppose that $S = ATTAGTACA\$$ is a string. The suffix tree for the string S, $T(S)$, is shown in Fig. 1.

Suppose that $P = TAA$ is a pattern. After reading the first two characters of P, T and A, we will arrive to the branching node $\underline{TA}$. Because, the edge A is not outgoing from the branch node $\underline{TA}$, we cannot continue with the matching P against $T(S)$. In other words, P does not occur in $T(S)$. Therefore, P is not the substring of S.

Suppose that $P = ATA$ is a pattern. Follow the first edge from the root to the node $\underline{A}$. The node $\underline{A}$ has the edge $TTAGTACA\$$ leading to the leaf $\underline{0}$. The next character to be read in P is the last character in P - $\underline{A}$. $\underline{A}$ does not match the next character $\underline{T}$ of the edge $TTAGTACA\$$. Therefore, P does not occur in $T(S)$ that is P is not the substring of S.

If we can find that P occurs in $T(S)$, then we can also find the positions in S where P occurs. Suppose that $P = TA$ is a pattern and assume $T(S)$ of Fig. 1. Following the second edge from the root, we will reach to the branching node $\underline{TA}$. Therefore, P is the substring of S. The leaf numbers in the subtree below the branching node $\underline{TA}$ are 2 and 5. Therefore, TA starts in S at positions 2 and 5.

The time complexity of this algorithm is as follows. The construction of $T(S)$ takes $O(n)$ time, where n is the length of S. The search for occurrences of P takes $O(m + z)$ time, where m is the length of P and z is the number of occurrences of P in S. (Note that z can be larger than m, but not larger than n.) Hence, the asymptotic time for the complete search is the same as for the optimal on-line string matching algorithms such as Boyer-Moore or Knuth-Morris-Pratt, $O(n + m)$.

5.2 Exact set matching

The *exact set matching problem* is: Given an array of strings/sequences $(S_k) = S_1, S_2, \ldots, S_k$ and a pattern string P. Find all positions of the pattern P in the sequence (S_k).

The exact set matching problem can be solved using the suffix trees on the following straightforward way:

- Concatenate the strings $S_1, S_2, \ldots, S_k$ separated by the unique separator symbols $\$_i$, where $i = 1, \ldots, k\text{-}1$, into the string S, $S = S_1\$_1S_2\$_2 \ldots \$_{k-1}S_k$.
  The string S is called the generalized string of the strings $S_1, S_2, \ldots, S_k$.
- Construct the suffix tree for the generalized string S, $T(S)$.

The suffix tree for the generalized string S, $T(S)$, is called the generalized suffix tree for the generalized string S. Leaves of the generalized suffix tree are labeled with pairs of the form (sequence number, start position). Often, the labels of leaf-edges are cut of after the first separator symbol. The pattern search is performed as in the standard suffix tree. Again, the pattern search takes $O(m + z)$ time to find all z occurrences of a pattern P of the length m.

Suppose that $S = BABAB\$_1AAB\$_2$ is a generalized string. The corresponding generalized suffix tree is shown in Fig. 2.

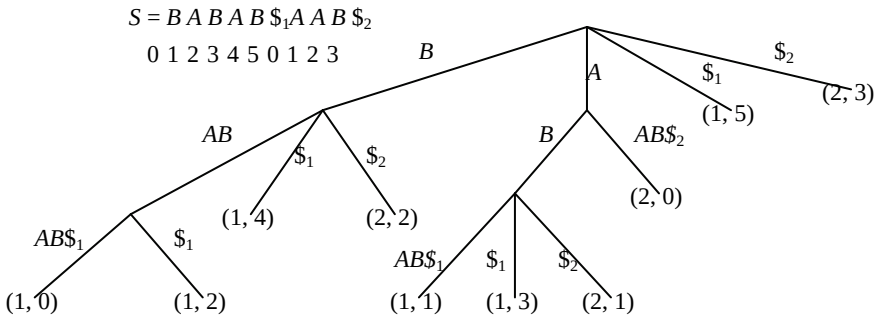


Fig. 2. The generalized suffix tree for the string S_1=*BABAB*\$$_1$ and the string S_2=*AAB*\$$_2$.

The advantage of constructing an index becomes clear when several searches are performed on the same string, as the following table shows (z denotes the output size).

-	- search for one pattern of length m	- search for k patterns of length m
- on-line algorithms	- $O(n + m)$	- $O(k(n + m))$
- suffix-tree algorithm	- $O(n + m)$	- $O(n + km + z)$

5.3 Minimal unique substrings

The *minimal unique substrings problem* is: Given a string S and a constant (a positive integer) l. Find - enumerate all substrings u of the string S that satisfy the following properties:

- u occurs exactly once in S (uniqueness)
- all proper prefixes of u occur at least twice in S (minimality)
- the length of u is greater or equal than l.

Suppose that S = *ATTAGTACA*\$ is a string and l = 2 is a constant. The minimal unique substrings of S are *TAG* and *AT* (see Fig. 1). The substring *ATT,* for example, is not the minimal unique substring of S, since the proper prefix *AT* of *ATT* is already unique, that is the minimality condition is not satisfied.

To solve the minimal unique substrings problem, exploit the following two properties of the suffix tree for the string S:

- if a string w occurs at least twice in the string S, there are at least two suffixes in S for which w is a proper prefix. Therefore, in the suffix tree $T(S)$, w corresponds to a path ending with an edge to a branching node;
- if a string w occurs only once in the string S, there is only one suffix in S for which w is a prefix. Therefore, in the suffix tree $T(S)$, w corresponds to a path ending with an edge to a leaf.

According to the second property, we can find the unique strings by looking at the paths ending on edges to a leaf. So, if we have reached a branching node, say w, then we only have to enumerate the leaf edges outgoing from w. Suppose w->y is the edge from the branching

node w leading to the leaf y, labeled av where a is the first character on that edge. Then wa occurs only once in S, i.e. it is unique. Moreover, w corresponds to the path leading to w and by the second property, w occurs at least twice. Finally, we only have to check if the length of wa is larger or equal than l.
The suffix tree based algorithm to solve the minimal unique substrings problem is very simple.
Let us apply the algorithm to the suffix tree of Fig. 1. Assume that $l = 2$. We can skip the root, since it would result in strings, which are too short. Let us consider the branching node reached by the edge from the root labeled TA. Then $w = TA$ and with the first character G of the label of the second edge we obtain the minimal unique substring TAG. The other solution AT can be found by looking at the other branching node reached by the label A from the root together with its zero-numbered edge.
The running time of the minimal unique substrings algorithm is linear in the number of nodes and edges in the suffix tree, since we have to visit each of these only once and for each we do a constant amount of work. The algorithm runs in linear time since the suffix tree can be constructed in linear time and there are $O(n)$ nodes and edges in the suffix tree. This is optimal, since the running time is linear in the size of its input.
The minimal unique substrings problem has applications in primer design.

5.4 Maximal unique match

The standard dynamic programming algorithm to compute the optimal alignment of two sequences of the length m and the length n requires $O(mn)$ steps. This is too slow for the cases when sequences have hundreds of thousands or millions characters.
There are algorithms that allow aligning of two genomes under the assumption that genomes are similar. Genomes are similar if they have long identical subsequences. Identical subsequences, called MUMs (Maximal Unique Matches), are almost certainly part of high quality and efficient alignment of two genomes. The first step of the maximal unique match algorithm is to find MUMs. MUMs are taken as the fixed part of alignment. The remaining parts of genomes (the parts not included in MUMs) are aligned with traditional dynamic programming methods.
In this section, we will show how to compute MUMs in linear time. This is very important for the applicability of the maximal unique match algorithm, the MUM algorithm. We do not consider how to compute the final alignment.
The *maximal unique match problem, the MUM problem,* is: Given two sequences S, $S' \in \Sigma^*$ (the genomes) and a constant (a positive integer) l. Find all subsequences u with the following properties:

- $|u| \geq l$.
- u occurs exactly once in S and exactly once in S' (uniqueness).
- for any character a neither ua nor au occurs both in S and in S' (maximality).

Suppose that $S = CCTTCGT$ *is a string*, $S' = CTGTCGT$ *is another string*, and $l = 2$ is a constant. There are two maximal unique matches CT and $TCGT$. Consider an optimal alignment of these two sequences (assuming the same costs for insertions, deletions, and replacements):
CCT-TCGT
-CTGTCGT

Clearly, two MUMs *CT* and *TCGT* are parts of this alignment.
To compute MUMs, we first have to construct the suffix tree for the concatenation of the two sequences S and S'. To prevent any match that occurs on the borderline between S and S', we put the unique symbol # between S and S', i.e. we construct the suffix tree for the string X, where $X = S\#S'$. A MUM, say u, must occur exactly twice in X, once in S and once in S'. Therefore, u corresponds to the path in the suffix tree $T(X)$ ending with an edge to a branching node. Since u is the right-maximal by definition (i.e. for any symbol a, ua does not occur both in S and in S'), u must even correspond to the branching node. In other words, for each MUM u there is the branching node $\underline{u}$ in the suffix tree for the string X.
Since u occurs twice in X, there are exactly two leaves in the subtree below u. The subtree can contain no branching node, hence there are two leaf edges outgoing from u. One edge must lead to the leaf, say v, that corresponds to a suffix starting in S and the other edge must lead to the leaf, say w, that corresponds to a suffix starting in S'. For the given branching node u, the existence of exactly two such leaf edges can be checked easily. What remains is to verify left-maximality, i.e. to check if there is the character a such that au occurs both in S and in S'. Suppose that the leaf v has the leaf number i and the leaf w has the leaf number j. Then u is left maximal, if and only if $i = 0$ or $X_{i-1} \neq X_{j-1}$. In other words, we only have to look at the two positions immediately to the left of two positions in X where u starts.

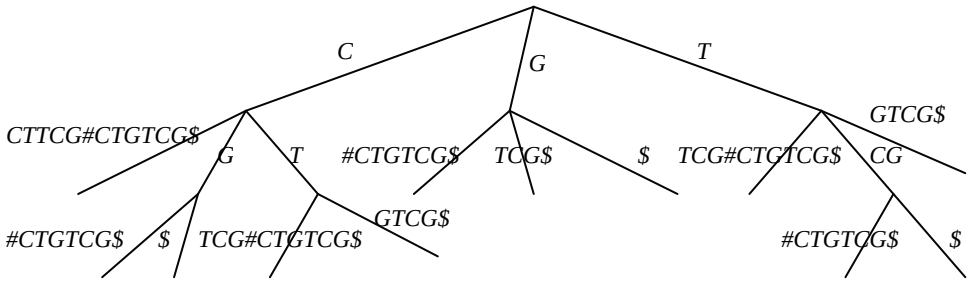


Fig. 3. The suffix tree for *CCTTCG#CTGTCG*$ without the leaf edges from the root

Suppose that $S = CCTTCG$ is a string, $S' = CTGTCG$ is another string, and $l = 2$ is a constant. Consider the suffix tree for the string $S\#S'$. This is shown in Fig. 3. Obviously, the string *TCG* occurs once in S and S', since there are two corresponding edges from the branching node $\underline{TCG}$. Comparing the character G and the character T immediately to the left of the occurrences of *TCG* in S and S' verifies the left maximality. The string *CT* also occurs once in S and once in S', as verified by the two leaf edges from $\underline{CT}$. The left-maximality is obvious, since the character *C* and the character # to the left of the occurrences are different.

5.5 Maximal repeat

If a sequence S may be represented by the array $S_0 \ldots S_{n-1}$, then the sequence S is indexed from 0 to n-1. If a sequence S is indexed, then a subsequence $S_i \ldots S_j$ of the string S may be represented by the pair (i, j). A pair (l, r), where $l = (i, j)$ and $r = (i', j')$, is a repeat, if $i < i'$ and $S_i \ldots S_j = S_{i'} \ldots S_{j'}$. l is the left instance of the repeat and r is the right instance of the repeat.

Note that the left instance of the repeat and the right instance of the repeat may overlap.
Suppose that $S = GAGCTCGAGC$ is a string. The string S contains the following repeats of the length $l \geq 2$:

-	((0, 3), (6, 9))	-	*GAGC*
-	((0, 2), (6, 8))	-	*GAG*
-	((0, 1), (6, 7))	-	*GA*
-	((1, 3), (7, 9))	-	*AGC*
-	((2, 3), (8, 9))	-	*GC*

The example shows that shorter repeats are often contained in longer repeats. To remove redundancy, we restrict to maximal repeats. A repeat is the maximal if it is the left maximal and the right maximal. These notions are formally defined as follows: The repeat $((i, j), (i', j'))$ is the *left maximal* if and only if $i\text{-}1 < 0$ or $S_{i-1} \neq S_{i'-1}$. The repeat $((i, j), (i', j'))$ is the *right maximal* if and only if $j'+1 > n\text{-}1$ or $S_{j+1} \neq S_{j'+1}$.
From now, we will restrict ourselves to the maximal repeats. All repeats, which are not the maximal repeats, can be obtained from the maximal repeats. In the example above, the last four repeats can be extended to the left or to the right. Hence, only the first repeat is maximal.
In the following, we will present an algorithm to compute all maximal repeats of a given sequence. It works in two phases. In the first phase, the leaves of the suffix tree are annotated. In the second phase, the repeats are output while simultaneously the branching nodes are annotated.
We will show how the algorithm to compute all maximal repeats works for the string $\$_1GCGC\$_2GGCG\$_3$. The corresponding suffix tree (with some unimportant edges left out) is shown in Fig. 4.
Suppose that:

- a string S of the length n over the alphabet $\sum$ such that the first and the last character of S both occur exactly once and
- the suffix tree for a string S is given (as in Fig. 4).

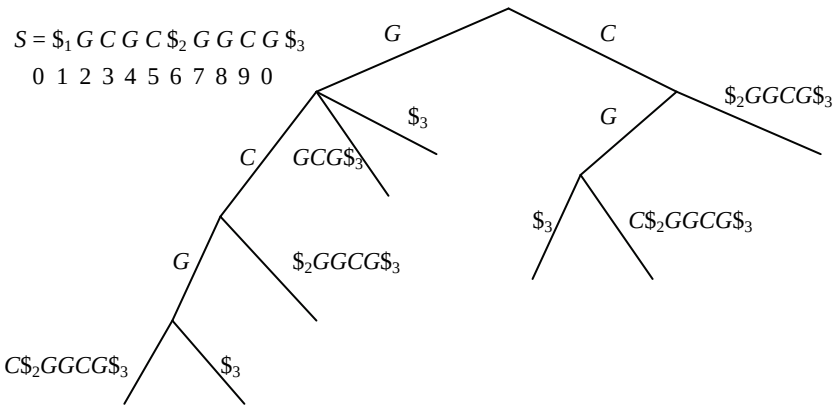


Fig. 4. The suffix tree for the string $\$_1GCGC\$_2GGCG\$_3$.

We can ignore leaf edges from the root, since the root corresponds to the repeats of the length zero, and we are not interested in these.

In the first phase, the algorithm annotates each leaf of the suffix tree: if $v = S_i \ldots S_n$, then the leaf $\underline{v}$ is annotated by the pair (a, i), where i is the starting position of the suffix v and $a = S_{i-1}$ is the character to the immediate left to the position i. (a, i) is the *leaf annotation* of $\underline{v}$ and it is denoted $A(\underline{v}, S_{i-1}) = \{i\}$. We assume that $A(\underline{v}, c) = \phi$ (the empty set) for all characters $c \in \Sigma$ different from S_{i-1}. This assumption holds in general (also for branching nodes), whenever there is no annotation (c, j) for some j.

The leaf annotation of the suffix tree for the string S in Fig. 4, is shown in Fig. 5.

Leaf edges from the root are not shown. These edges are not important for the algorithm.

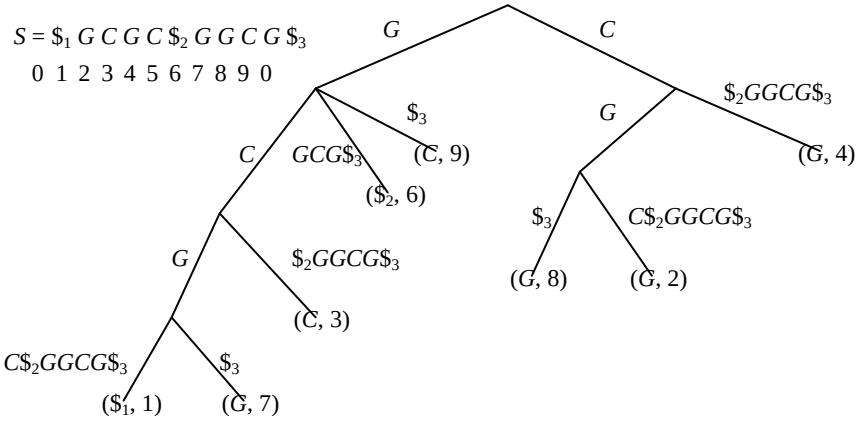


Fig. 5. The suffix tree for the string $\$_1GCGC\$_2GGCG\$_3$ with leaf annotation.

The leaf annotation gives the character upon which we decide the left-maximality of a repeat, plus the position where the repeated string occurs. We have only to combine this information at the branching nodes appropriately. This is done in the second phase of the algorithm: In a bottom-up traversal, the repeats are output and simultaneously the annotation for the branching nodes is computed. A bottom-up traversal means that a branching node is visited only after all nodes in the subtree below that node have been visited. Each edge, say $\underline{w} \to \underline{v}$ with a label au, is processed as follows: At first repeats (for w) are output by combining the annotation already computed for the node $\underline{w}$ with the complete annotation stored for $\underline{v}$ (this was already computed due to the bottom-up strategy). In particular, we output all pairs $((i, i + q\text{-}1), (j, j + q\text{-}1))$, where

- q is the depth of node $\underline{w}$, i.e. $q = |w|$,
- $i \in A(\underline{w}, c)$ and $j \in A(\underline{v}, c')$ for some characters $c \neq c'$,
- $A(\underline{w}, c)$ is the annotation already computed for $\underline{w}$ w.r.t. character c and $A(\underline{v}, c')$ is the annotation stored for node $\underline{v}$ w.r.t. character c'.

The second condition means that only those positions are combined which have different characters to the left. It guarantees the left-maximality of repeats. Recall that we consider

processing the edge $\underline{w} \rightarrow \underline{v}$ with the label au. The annotation already computed for $\underline{w}$ was inherited along edges outgoing from $\underline{w}$, which are different from $\underline{w} \rightarrow \underline{v}$ with the label au. The first character of the label of an edge, say c, is different from a. Now since w is the repeated substring, c and a are characters to the right of w. Consequently only those positions are combined which have different characters to the right. In other words, the algorithm also guarantees the right maximality of repeats.

As soon as for the current edge the repeats are output, the algorithm computes the union $A(\underline{w}, c) \cup A(\underline{v}, c)$ of all characters c, i.e. the annotation is inherited from the node $\underline{v}$ to the node $\underline{w}$. In this way, after processing all edges outgoing from $\underline{w}$, this node is annotated by the set of positions where w occurs, and this set is divided into (possibly empty) disjoint subsets $A(\underline{w}, c_1),\ldots, A(\underline{w}, c_r)$, where $\Sigma = \{c_1,\ldots,c_r\}$.

Fig. 6 shows the annotation for a large part of the previous suffix tree and some repeats. The bottom up traversal of the suffix tree for the string $\$_1GCGC\$_2GGCG\$_3$ begins with the node $\underline{GCG}$ of depth 4, before it visits the node $\underline{GC}$ of depth 2. The maximal repeats for the string $\underline{GC}$ are computed as follows: The algorithm starts by processing the first edge outgoing from $\underline{GC}$. Since initially, there is no annotation for $\underline{GC}$, no repeat is output, and $\underline{GC}$ is annotated by (C, 3). Then the second edge is processed. This means that the annotation ($\$_1$, 1) and ($G$, 7) for $\underline{GCG}$ is combined with the annotation (C, 3). This give repeats ((1, 2), (3, 4)) and ((3, 4), (7, 8)). The final annotation for $\underline{GC}$ becomes (C, 3), (G, 7), ($\$_1$, 1) which also can be read as $A(\underline{GC}, C) = \{3\}$, $A(\underline{GC}, G) = \{7\}$, and $A(\underline{GC}, \$_1) = \{1\}$.

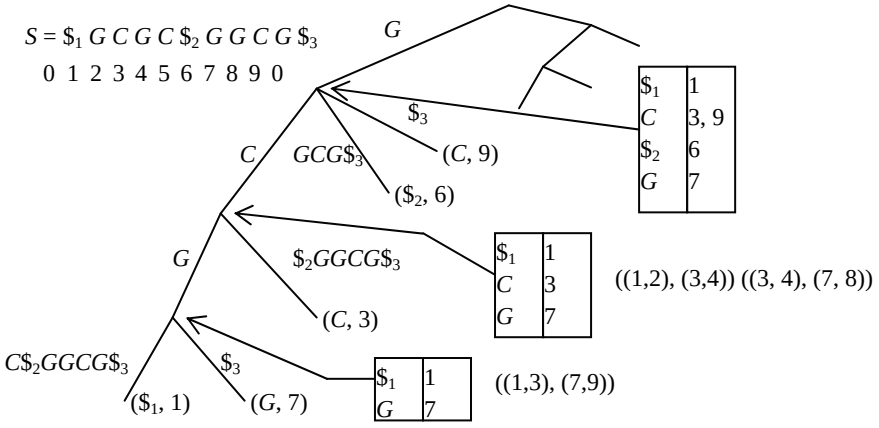


Fig. 6. The annotation for a large part of the suffix tree of Fig. 5 and some repeats.

Let us now consider the running time of the maximal repeats algorithm. Traversing the suffix tree bottom-up can be done in time linear in the number of nodes. Follow the paths in the suffix tree, each node is visited only once. Two operations are performed during the traversal: the output of repeats and the combination of annotations. If the annotation for each node is stored in linked lists, then the output operation can be implemented such it runs in time linear in the number of repeats. Combining annotations only involves linking

lists together. This can be done in time linear in the number of nodes visited during the traversal. Recall, that the suffix tree can be constructed in $O(n)$ time. Therefore, the algorithm requires $O(n + z)$ time, where n is the length of the input string and z is the number of repeats.

To analyze the space consumption of the maximal repeats algorithm, the annotations for all nodes do not have to be stored all at once. As soon as a node and its father have been processed, the annotations are no longer needed. The consequence is - the annotation only requires $O(n)$ space. Therefore, the space consumption of the algorithm is $O(n)$.

The maximal repeats algorithm is optimal, since its space and time requirement are linear in the size of the input plus the output.

6. References

[1] P. Weiner, "Linear Pattern Matching Algorithms," Proc. 14th IEEE Annual Symp. on Switching and Automata Theory, pp1-11, 1973

[2] E. M. McCreight, "A Space-Economical Suffix Tree Construction Algorithm," *Journal of Algorithms*, Vol. 23, No. 2, pp262-272, 1976

[3] E. Ukkonen, "On-line Construction of Suffix Trees," Algorithmica, Vol. 14, No. 3, pp249-260, 1995

[4] Arthur L. Delcher, Simon Kasif, Robert D. Fleischmann, Jeremy Peterson, Owen White and Steven L. Salzberg, "Alignment of whole genomes," *Nucleic Acids Research*, Vol. 27, pp. 2369–2376, 1999

[5] Aurthur L. Delcher, Adam Phillippy, Jane Carlton and Steven L. Salzberg, "Fast algorithms for large-scale genome alignment and comparison," *Nucleic Acids Research*, Vol. 30, pp. 2478–2483, 2002

[6] S. Kurtz and Chris Schleiermacher, "REPuter fast computation of maximal repeats in complete genomes," *Bioinformatics*, Vol. 15, No. 5, pp.426-427, 1999

[7] Stefan Kurtz, Jomuna V. Choudhuri, Enno Ohlebusch, Chris Schleiermacher, Jens Stoye and Robert Giegerich, "REPuter the manifold applications of repeat analysis on a genomic," *Nucleic Acids Research*, Vol. 29, No.22, pp. 4633–4642, 2002

[8] Stefan Kurtz, "Reducing the space requirement of suffix trees," *Software Pract. Experience*, Vol. 29, pp. 1149-1171, 1999

[9] Hongwei Huo and Vojislav Stojkovic, "A Suffix Tree Construction Algorithm for DNA Sequences," IEEE 7th International Symposium on BioInformatics & BioEngineering. Harvard School of Medicine, Boston, MA, October 14-17, Vol. II, pp. 1178-1182, 2007.

[10] Stefan Kurtz, "Foundations of sequence analysis," lecture notes for a course in the winter semester, 2001

[11] D. E. Knuth, J. H. Morris, and V. B. Pratt, "Fast pattern matching in strings," *SIAM Journal on Computing*, 1977, Vol. 6, pp. 323-350, 1997

[12] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM*, 1977,Vol. 20, pp. 762-772, 1997

[13] Yun-Ching Chen & Suh-Yin Lee, "Parsimony-spaced suffix trees for DNA sequences," *ISMSE'03*, Nov, pp.250-256, 2003.

[14] Giegerich, R., Kurtz, S., Stoye, J.,"Efficient implementation of lazy suffix trees," *Soft. Pract. Exp*. Vol. 33,1035–1049, 2003

[15] Schurmann, K.-B., Stoye, J.,"Suffix-tree construction and storage with limited main memory," Technical Report 2003-06, 2003, University of Bielefeld, Germany.

[16] Dan Gusfield, "Algorithms on strings, trees, and sequences," Cambridge University Press, 1997

6

Bayesian Framework for State Estimation and Robot Behaviour Selection in Dynamic Environments

Georgios Lidoris, Dirk Wollherr and Martin Buss
Institute of Automatic Control Engineering, Technische Universität München
D-80290 München, Germany

1. Introduction

One of the biggest challenges of robotics is to create systems capable of operating efficiently and safely in natural, populated environments. This way, robots can evolve from tools performing well-defined tasks in structured industrial or laboratory settings, to integral parts of our everyday lives. However such systems require complex cognitive capabilities, to achieve higher levels of cooperation and interaction with humans, while coping with rapidly changing objectives and environments.

In order to address these challenges a robot capable of autonomously exploring densely populated urban environments, is created within the Autonomous City Explorer (ACE) project (Lidoris et al., 2007). To be truly autonomous such a system must be able to create a model of its unpredictable dynamic environment based on noisy sensor information and reason about it. More specifically, a robot is envisioned that is able to find its way in an urban area, without a city map or GPS. In order to find its target, the robot will approach pedestrians and ask for directions.

Due to sensor limitations the robot can observe only a small part of its environment and these observations are corrupted by noise. By integrating successive observations a map can be created, but since also the motion of the robot is subject to error, the mapping problem comprises also a localization problem. This duality constitutes the Simultaneous Localization And Mapping (SLAM) problem. In dynamic environments the problem becomes more challenging since the presence of moving obstacles can complicate data association and lead to incorrect maps. Moving entities must be identified and their future position needs to be predicted over a finite time horizon. The autonomous sensory-motor system is finally called to make use of its self-acquired uncertain knowledge to decide about its actions.

A Bayesian framework that enables recursive estimation of a dynamic environment model and action selection based on these uncertain estimates is introduced. This is presented in Section 2. In Section 3, it is shown how existing methods can be combined to produce a working implementation of the proposed framework. A Rao-Blackwellized particle filter (RBPF) is deployed to address the SLAM problem and combined with recursive conditional

particle filters in order to track people in the vicinity of the robot. Conditional filters have been used in the literature for tracking given an a priori known map. In this paper they are modified to be utilized with incrementally constructed maps. This way a complete model of dynamic, populated environments can be provided. Estimations serve as the basis for all decisions and actions of robots acting in the real world. In Section 4 the behaviours of the robot are described. In Section 5 it is shown how these are selected so that uncertainty is kept under control and the likelihood of achieving the tasks of the system is increased. In highly dynamic environments decision making needs to be performed as soon as possible. However, optimal planning is either intractable or requires very long time to be completed and since the world is changing constantly, any plan becomes outdated quickly. Therefore the proposed behaviour selection scheme is based on greedy optimization algorithms.

Fig. 1. The Autonomous City Explorer (ACE) robotic platform

2. Bayesian framework for state estimation and behaviour selection

The problem of action selection has been addressed by different researchers in various contexts. The reviews of (Tyrrell, 1993) and (Prescott et al., 1999) cover the domains of ethology and neuroscience. (Maes, 1989) addresses the problem in the context of artificial

agents. In robotics, action selection is related to optimization. Actions are chosen so that the utility toward the goal of the robot is maximized. Several solutions have been proposed which can be distinguished in many dimensions. For example whether the action selection mechanism is competitive or cooperative (Arkin, 1998), or whether it is centralized or decentralized (Pirjanian, 1999). Furthermore, explicit action selection mechanisms can be incorporated as separate components into an agent architecture (Bryson, 2000). Reinforcement learning has been applied to selection between conflicting and heterogeneous goals (Humphrys, 1997). A distinction was made between selecting an action to accomplish a unique goal and choosing between conflicting goals.

However, several challenges remain open. Real-world environments involve dynamical changes, uncertainty about the state of the robot and about the outcomes of its actions. It is not clear how uncertain environment and task knowledge can be effectively expressed and how it can be incorporated into an action selection mechanism. Another issue remains dealing with the combinatorial complexity of the problem. Agents acting in dynamic environments cannot consider every option available to them at every instant in time, since decisions need to be made in real-time. Consequently, approximations are required.

The approach presented in this chapter addresses these challenges. The notion of behavior is used, which implies actions that are more complex than simple motor commands. Behaviors are predefined combinations of simpler actuator command patterns, that enable the system to complete more complex task objectives (Pirjanian, 1999). A Bayesian approach is taken, in order to deal with uncertain system state knowledge and uncertain sensory information, while selecting the behaviours of the system. The main inspiration is derived from the human cognition mechanisms. According to (Körding & Wolpert, 2006), action selection is a fundamental decision process for humans. It depends both on the state of body and the environment. Since signals in the human sensory and motor systems are corrupted by variability or noise, the nervous system needs to estimate these states. It has been shown that human behaviour is close to that predicted by Bayesian theory, while solving estimation and decision problems. This theory defines optimal behaviour in a world characterized by uncertainty, and provides a coherent way of describing sensory-motor processes.

Bayesian inference also offers several advantages over other methods like Partially Observable Markov Decision Processes (POMDPs) (Littman et al., 1995), which are typically used for planning in partially observable uncertain environments. Domain specific knowledge can be easily encoded into the system by defining dependences between variables, priors over states or conditional probability tables. This knowledge can be acquired by learning from an expert or by quantifying the preferences of the system designer.

A relationship is assigned between robot states and robot behaviours, weighted by the state estimation uncertainty. Behaviour selection is then performed based on greedy optimization. No policy learning is required. This is a major advantage in dynamic environments since learning policies can be computationally demanding and policies need to be re-learned every time the environment changes. In such domains the system needs to be able to decide as soon as possible. There is evidence (Emken et al., 2007) that also humans use greedy algorithms for motor adaptation in highly dynamic environments. However, the

optimality of this approach depends on the quality of the approximation of the true distributions. State of the art estimation techniques enable very effective and qualitative approximations of arbitrary distributions. In the remainder of this section the proposed Bayesian framework is going to be presented in more detail.

2.1 Bayesian Inference

In terms of probabilities the domain of the city explorer can be described by the joint probability distribution $p(S_t, B_t, C_t, Z_t \mid U_t)$. This consists of the state of the system and the model of the dynamic environment S_t, the set of behaviors available to the system B_t, a set of processed perceptual inputs that are associated with events in the environment and are used to trigger behaviors C_t, system observations Z_t and control measurements U_t that describe the dynamics of the system. In the specific domain, observations are the range measurements acquired by the sensors and control measurements are the odometry measurements acquired from the mobile robot. The behavior triggering events depend on the perceived state of the system and its goals. The state vector S_t is defined as

$$S_t = \{X_t, m, Y_t^1, Y_t^2, ..., Y_t^M\} \tag{1}$$

where X_t represents the trajectory of the robot, m is a map of the environment and Y^1_t, Y^2_t,..., Y^M_t the positions of M moving objects present at time t. Capital letters are used throughout this chapter to denote the full time history of the quantities from time point *0* to time point t, whereas lowercase letters symbolize the quantity only at one time step. For example z_t would symbolize the sensor measurements acquired only at time step t.

The joint distribution can be decomposed to simpler distributions by making use of the conjunction rule.

$$p(S_t, B_t, C_t, Z_t \mid U_t) = p_0 \prod_{j=1}^{t} \{p(s_j \mid S_{j-1}, U_j) p(z_j \mid S_j) p(b_j \mid B_{j-1}, C_j, S_j)\} \tag{2}$$

Initial conditions, $p(s_0, b_0, c_0, z_0, u_0)$, are expressed for simplicity by the term p_0. The first term in the product represents the dynamic model of the system and it expresses our knowledge about how the state variables evolve over time. The second one expresses the likelihood of making an observation z_t given knowledge of the current state. This is the sensor model or perceptual model. The third term constitutes the behaviour model. Behaviour probability depends on behaviours selected previously by the robot, on perceptions and the estimated state at the current time.

The complexity of this equation is enormous, since dependence on the whole variable history is assumed. In order to simplify it, Bayes filters make use of the Markov assumption. Observations z_t and control measurements u_t are considered to be conditionally independent of past measurements and control readings given knowledge of the state s_t. This way the joint distribution is simplified to contain first order dependencies.

$$p(S_t, B_t, C_t, Z_t \mid U_t) = p_0 \prod_{j=1}^{t} \{p(s_j \mid s_{j-1}, u_j) p(z_j \mid s_j) p(b_j \mid b_{j-1}, c_j, s_j)\} \tag{3}$$

As discussed previously, the goal of an autonomous system is to be able to choose its actions based only on its perceptions, so that the probability of achieving its goals is maximized. This requires the ability to recursively estimate all involved quantities. Using the joint distribution described above this is made possible. In the next subsection it will be analyzed how this information can be derived, by making use of Bayesian logic.

2.2 Prediction

The first step is to update information about the past by using the dynamic model of the system, in order to obtain a predictive belief about the current state of the system. After applying the Bayes rule and marginalizing irrelevant variables, the following equation is acquired.

$$p(s_t \mid Z_{t-1}, U_t) \propto \sum_{s_{t-1}} p(s_t \mid s_{t-1}, u_t)\, p(s_{t-1} \mid Z_{t-1}, U_{t-1}) \tag{4}$$

More details on the mathematical derivation can be found in (Lidoris et al., 2008). The first term of the sum is the system state transition model and the second one is the prior belief about the state of the system. Prediction results from a weighted sum over state variables that have been estimated at the previous time step.

2.3 Correction step

The next step of the estimation procedure is the correction step. Current observations are used to correct the predictive belief about the state of the system, resulting to the posterior belief $p(s_t \mid Z_t, U_t)$. During this step, all information available to the system is fused.

$$p(s_t \mid Z_t, U_t) \propto \sum_{s_{t-1}} p(z_t \mid s_t)\, p(s_t \mid Z_{t-1}, U_t) \tag{5}$$

It can be seen from (5) that the sensor model is used to update the prediction with observations. The behaviour of the robot is assumed not to have an influence on the correction step. The effect of the decision the robot will take at the current time step about its behaviour, will be reflected in the control measurements that are going to be received at the next time step. Therefore the behaviour and behaviour trigger variables have been integrated out of (5).

2.4 Estimation of behaviour probabilities

Finally, the behaviour of the system needs to be selected by using the estimation about the state of the system and current observations. That includes calculating the probabilities over the whole set of behaviour variables, $p(b_t \mid S_t, C_t, Z_t, U_t)$ for the current time step. The same inference rules can be used as before, resulting to the following equation

$$p(b_t \mid S_t, C_t Z_t, U_t) \propto \sum_{s_t} p(z_t \mid s_t)\, p(b_t \mid b_{t-1}, c_t, s_t)\, p(s_t \mid Z_{t-1}, U_t) \tag{6}$$

By placing (5) in (6) an expression is acquired which contains the estimated posterior.

$$p(b_t \mid S_t, C_t Z_t, U_t) \propto \sum_{s_t} p(b_t \mid b_{t-1}, c_t, s_t) p(s_t \mid Z_t, U_t) \tag{7}$$

As mentioned previously, system behaviours are triggered by processed perceptual events. These events naturally depend on the state of the system and its environment. Therefore the behaviour selection model $p(b_t \mid b_{t-1}, c_t, s_t)$ can be further analyzed to

$$p(b_t \mid b_{t-1}, c_t, s_t) = p(b_t \mid b_{t-1}, c_t) p(c_t \mid s_t) \tag{8}$$

and replacing equation (8) to (7) leads to

$$p(b_t \mid S_t, C_t Z_t, U_t) \propto \sum_{s_t} p(b_t \mid b_{t-1}, c_t) p(c_t \mid s_t) p(s_t \mid Z_t, U_t) \tag{9}$$

The behaviour model is weighted by the estimated posterior distribution $p(s_t \mid Z_t, U_t)$ for all possible values of the state variables and the probability of the behaviour triggers. The term $p(b_t \mid b_{t-1}, c_t)$ expresses the degree of belief that given the current perceptual input the current behaviour will lead to the achievement of the system tasks. This probability can be pre-specified by the system designer or can be acquired by learning.

3. Uncertainty representation and estimation in unstructured dynamic environments

In the previous section a general Bayesian framework for state estimation and decision making has been introduced. In order to be able to use it and create an autonomous robotic system, the related probability distributions need to be estimated. How this can be made possible, is discussed in this section. The structure of the proposed approach is presented in Fig. 2. Whenever new control (e.g. odometry readings) and sensor measurements (e.g. laser range measurements) become available to the robot, they are provided as input to a particle filter based SLAM algorithm. The result is an initial map of the environment and an estimate of the trajectory of the robot. This information is used by a tracking algorithm to obtain a model of the dynamic part of the environment. An estimate of the position and velocity of all moving entities in the environment is acquired, conditioned on the initial map and position of the robot. All this information constitutes the environment model and the estimated state vector s_t. A behaviour selection module makes use of these estimates to infer behaviour triggering events c_t and select the behaviour b_t of the robot. According to the selected behaviour, a set of actuator commands is generated which drives the robot toward the completion of its goals. In the following subsections each of the components and algorithms mentioned here are going to be further analyzed.

3.1 Simultaneous localization and mapping

The problem of simultaneous localization and mapping is one of the fundamental problems in robotics and has been studied extensively over the last years. It is a complex problem because the robot needs a reliable map for localizing itself and for acquiring this map it requires an accurate estimate of its location. The most popular approach (Dissanayake et al., 2002) is based on the Extended Kalman Filter (EKF). This approach is relatively effective

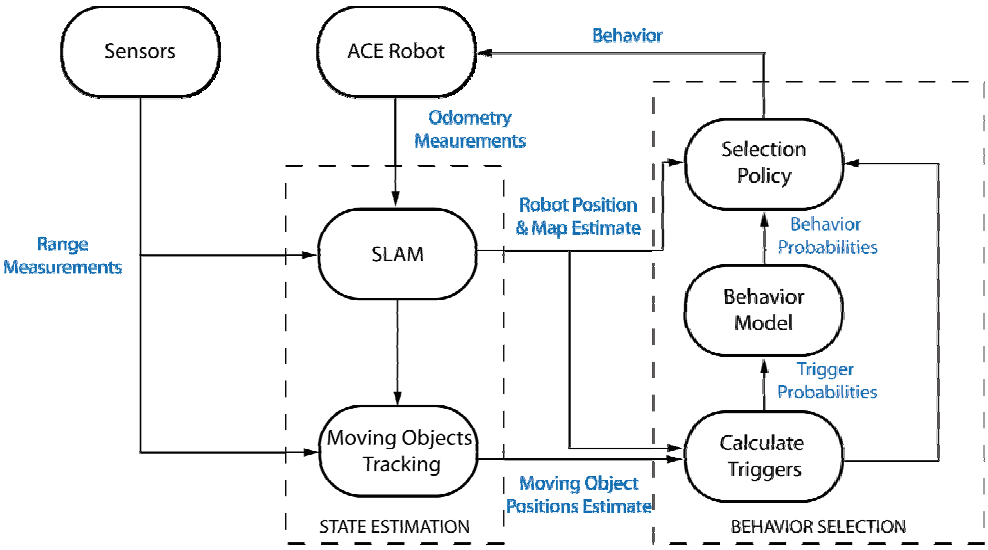


Fig. 2. Proposed approach for modelling dynamic environments and behaviour selection.

since the resulting estimated posterior is fully correlated about landmark maps and robot poses. Its disadvantage is that motion model and sensor noise are assumed Gaussian and it does not scale well to large maps, since the full correlation matrix is maintained. Another well known approach (Thrun et al., 2004) corrects poses based on the inverse of the covariance matrix, which is called information matrix and is sparse. Therefore predictions and updates can be made in constant time. Particle filters have been applied to solve many real world estimation and tracking problems (Doucet et al. 2000), (Murphy, 1999) since they provide the means to estimate the posterior over unobservable state variables, from sensor measurements. This framework has been extended, in order to approach the SLAM problem with landmark maps in (Montemerlo et al., 2002). In (Grisetti et al., 2005) a technique is introduced to improve grid-based Rao-Blackwellized SLAM. The approach described here is similar to this technique, with the difference that scan-matching is not performed in a per-particle basis but only before new odometry measurements are used by the filter.

This approach allows the approximation of arbitrary probability distributions, making it more robust to unpredicted events such as small collisions which often occur in challenging environments and cannot be modelled. Furthermore it does not rely on predefined feature extractors, which would assume that some structures in the environment are known. This allows more accurate mapping of unstructured outdoor environments. The only drawback is that the approximation quality depends on the number of particles used by the filter. More particles result to increased required computational costs. However if the appropriate proposal distribution is chosen, the approximation can be kept very accurate even with a small number of particles. In the remainder of this section the approach is briefly highlighted.

The idea of Rao-Blackwellization is that it is possible to evaluate (Doucet et al.,2000) some of the filtering equations analytically and some others by Monte Carlo sampling. This results in estimators with less variance than those obtained by pure Monte Carlo sampling.

In the context of SLAM the posterior distribution $p(X_t, m \mid Z_t, U_t)$ needs to be estimated. Namely the map m and the trajectory X_t of the robot need to be calculated based on the observations Z_t and the odometry measurements U_t, which are obtained by the robot and its sensors.
The use of the Rao-Blackwellization technique, allows the factorization of the posterior.

$$p(X_t, m \mid Z_t, U_t) = p(X_t \mid Z_t, U_t)\, p(m \mid X_t, Z_t) \tag{10}$$

The posterior distribution $p(X_t \mid Z_t, U_t)$ can be estimated by sampling, where each particle represents a potential trajectory. This is the localization step. Next, the posterior $p(m \mid X_t, Z_t)$ over the map can be computed analytically as described in (Moravec, 1989) since the history of poses X_t is known.
An algorithm similar to (Grisetti et al., 2005) is used to estimate the SLAM posterior. Only the main differences are highlighted here. Each particle i is weighted according to the recursive formula

$$w_t^i = \frac{p(z_t \mid m_{t-1}, x_t^i)\, p(x_t^i \mid x_{t-1}^i, u_t)}{q(X_t^i \mid X_{t-1}^i, Z_t, U_t)} \tag{11}$$

The term $p(x^i_t \mid x^i_{t-1}, u_{t-1})$ is an odometry-based motion model. The motion of the robot in the interval $(t\text{-}1,t]$ is approximated by a rotation δ_{rot1}, a translation δ_{trans} and a second rotation δ_{rot2}. All rotations and translations are corrupted by noise. An arbitrary error distribution can be used to model odometric noise, since particle filters do not require specific assumptions about the noise distribution.
The likelihood of an observation given a global map and a position estimate is denoted as $p(z_t \mid m_{t-1}, x^i_t)$. It can be evaluated for each particle by using the particle map constructed so far and map correlation. More specifically a local map, $m^i_{local}(x^i_t, z_t)$ is created for each particle i. The correlation to the most actual particle map, m^i_{t-1}, is evaluated as follows:

$$\rho = \frac{\sum_{x,y} (m^i_{x,y} - \overline{m}^i)(m^i_{x,y,local} - \overline{m}^i)}{\sqrt{\sum_{x,y} (m^i_{x,y} - \overline{m}^i)^2 \sum_{x,y} (m^i_{x,y,local} - \overline{m}^i)^2}} \tag{12}$$

Where $\overline{m}^i$ is the average map value at the overlap between the two maps. The observation likelihood is proportional to the correlation value.
An important issue for the performance and the effectiveness of the algorithm is the choice of the proposal distribution. Typically the motion model is used, because it is easy to compute. In this work, the basis for the proposal distribution is provided by the odometry motion model, but is combined with a scan alignment that integrates the newest sensor measurements and improves the likelihood of the sampled particles. More specifically, new odometry measurements are corrected based on the current laser data and the global map, before being used by the motion model, through scan matching. This way information from the more accurate range sensors is incorporated. It must be noted here, that this is not performed on a per particle basis like in other approaches (Grisetti et al. 2005), since no great improvement in the accuracy of the estimator has been observed, compared with the higher computational costs involved.

3.2 Conditional particle filters for tracking

The methods mentioned above focus on the aspects of state estimation, belief representation and belief update in static environments. More specifically, an estimate of the most likely trajectory X_t of the robot, relative to an estimated static map, m_t, is provided. To estimate the full state of the environment as defined by (1), the position of moving objects needs also to be estimated.

Until now, no complete Bayesian framework exists for the dynamic environment mapping problem. One of the first attempts was introduced in (Wang et al., 2003). However it is based on the restrictive assumption of independence between static and dynamic elements in the environment. In (Haehnel et al., 2003) scan registration techniques are used to match raw measurement data to estimated occupancy grids in order to solve the data association problem and the Expectation-Maximization algorithm is used to create a map of the environment. A drawback is that the number of dynamic objects must be known in advance. Particle filters have been used to track the state of moving objects in (Montemerlo et al., 2002). However the static environment is assumed known. Particle filters have also been used in (Miller & Campbell, 2007) to solve the data association problem for mapping but without considering robot localization.

A similar approach as in (Montemerlo et al., 2002) is used here, extended to handle unknown static maps. The full state vector can then be estimated by conditioning the positions of moving objects on the robot trajectory estimate provided by tackling the SLAM problem.

$$p(S_t \mid Z_t, U_t) = p(X_t, m_t \mid Z_t, U_t) \prod_{m=1}^{M} p(Y_t^m \mid X_t, Z_t, U_t) \tag{13}$$

Each conditional distribution $p(Y_t^m \mid X_t, Z_t, U_t)$ is also represented by a set of particles. The particles are sampled from the motion model of the moving object. Several dynamics models exist, including constant velocity, constant acceleration and more complicated switching ones (Wang et al., 2003). Since people move with relatively low speeds and their motion can become very unpredictable, a Brownian motion model is an acceptable approximation.

Every particle of each particle filter, $y^{m,i}_t$, is weighted according to the measurement likelihood.

$$w_t^{m,i} = p(z_t \mid x_t, y_t^{m,i}) \tag{14}$$

In order to calculate the likelihood, each sensor reading needs to be associated to a specific moving object. However, measurements can be erroneous, objects might be occluded and the environment model might not be accurate, therefore leading to false associations. Persons are modelled as cylindrical structures during data association of the 2D laser data. The radius of the cylinder has been chosen experimentally. A laser measurement is associated with a person if its distance from a person position estimate is smaller than a maximum gating distance. In this case it is additionally weighted according to its distance from the position estimate. Therefore if the gating regions of two persons overlap, the person closest to a laser point is associated with it.

4. Robot behaviour description

In this section the application of the proposed general Bayesian framework to the Autonomous City Explorer (ACE) robot is going to be presented. The set of available behaviours consists of *Explore*, *Approach*, *Reach Goal* and *Loop Closing*. A detailed description of each one of them follows.

4.1 Explore

The ability to explore its environment in order to find people to interact with and increase its map knowledge, is fundamental for the robot. The robot performs greedy optimization in order to choose its next goal so that a trade-off is achieved between maximizing its information gain and minimizing traveling costs. Given an occupancy grid map, frontier regions between known and unknown areas are identified, as described in (Yamauchi, 1998). The subset of cells of the grid m that belong to a frontier region f, are denoted by m_f. The expected information gain $I(m_f,x_t)$ acquired by reaching a frontier region from the current robot position x_t can be calculated as in (Stachniss et al., 2005).. The traveling costs associated with reaching a frontier region, $cost(m_f,x_t)$, are proportional to the path length to it. In order to achieve the aforementioned trade-off, the autonomous explorer chooses its next goal, on the frontier region that maximizes the following objective function

$$m_f^* = \arg\max_{m_f} \{I(m_f, x_t) - \alpha \cos t(m_f, x_t)\} . \quad (15)$$

The parameter a is used to define how much the path cost should influence the exploration process and it can be chosen experimentally.

4.2 Approach

In order to interact with a person the robot needs first to approach her. This behaviour generates a target within a safety distance to a person. The person nearest to the robot is chosen in case more than one person is present simultaneously. Estimated positions from the tracker are used.

4.3 Reach goal

If the robot has been instructed a target through interaction, it needs to navigate safely to the specified target. An A* based planner is utilized that takes into account the motions of moving objects. A more detailed description is given in (Rohrmuller et al., 2007).

4.4 Loop closing

As the robot moves, the uncertainty about its position and its map grows constantly, therefore increasing the risk of failure. It is necessary for the robot to find opportunities to close a loop, therefore correcting its estimates. A way to acquire an estimate for the pose uncertainty $H(p(X_t | Z_t,U_t))$ of the robot, is to average over the uncertainty of the different poses along the path as in (Stachniss et al., 2005).

Since the distribution of the particle set can be arbitrary, it is not possible to efficiently calculate its entropy. A Gaussian approximation $N(\mu_t, \Sigma_t)$ can be computed based on the weighted samples with covariance Σ_t. The entropy can then be calculated only as a function of the covariance matrix. Such an approximation is rather conservative but absolutely

eligible, since a Gaussian probability distribution has higher entropy than any other distribution with the same variance.

In order to detect and close a loop, an approach similar to the one described in (Stachniss et al., 2004) is chosen. Together with the occupancy grid map a topological map is simultaneously created. This topological map consists of nodes, which represent positions visited by the robot. Each of these nodes contains visibility information between itself and all other nodes, derived from the associated occupancy grid. For each node the uncertainty of the robot $H_{init}(p(x_t \mid z_t, u_t))$ when it entered the node for the first time is also saved. To determine whether or not the robot should activate the loop-closing behaviour the system monitors the uncertainty $H(p(x_t \mid z_t, u_t))$ about the pose of the robot at the current time step. The necessary condition for starting the loop-closing process is that the geometric distance of the robot and a node in the map is small, while the graph distance in the topological map is large. If such a situation is detected the node is called entry point. Then the robot checks the difference between its initial uncertainty at the entry point and its current uncertainty, $H(p(x_t \mid z_t, u_t)) - H_{init}(p(x_t \mid z_t, u_t))$. If this difference exceeds a threshold then the loop is closed. This is done by driving the robot to the nearest neighbour nodes of the entry point in the topological map. During this process the pose uncertainty of the vehicle typically decreases, because the robot is able to localize itself in the map built so far and unlikely particles vanish.

5. Behaviour selection

As seen in the previous section, each of the behaviours available to the system has an objective which contributes to the achievement of the overall system goal. The robot needs to efficiently combine these behaviours by deciding when to activate which one and for how long. The proposed behaviour selection scheme is based on (9). This equation can be further analyzed by using the results of the state estimation process as summarized in (13).

$$p(b_t \mid S_t, C_t, Z_t, U_t) \propto \sum_{x_t} \left\{ p(b_t \mid b_{t-1}, c_t) p(c_t \mid x_t) p(x_t, m_t \mid Z_t, U_t) \prod_{m=1}^{M} p(y_t^m \mid x_t, Z_t, U_t) \right\} \quad (16)$$

It must be noted that the summation is done only over the state of the robot, x_t, since both the states of the moving objects and the map are conditioned on it. Particle filters have been used to approximate the posterior distributions $p(x_t, m_t \mid Z_t, U_t)$ and $p(y_t^m \mid x_t, Z_t, U_t)$. Therefore they can be approximated according to their particle weights (Arulampalam et al., 2002), given in (11) and (14), leading to the following equation:

$$p(b_t \mid S_t, C_t, Z_t, U_t) \propto \sum_{i=1}^{N} \left\{ p(b_t \mid b_{t-1}, c_t) p(c_t \mid x_t) w_t^i \delta(x_t - x_t^i) \prod_{m=1}^{M} \sum_{j=1}^{K} w_t^{m,j} \delta(y_t^m - y_t^{m,j}) \right\} \quad (17)$$

δ is the Dirac delta function, N is the number of particles used by the Rao-Blackwellized SLAM algorithm, M is the number of persons tracked by the robot and K is the number of particles of each conditional particle filter. After the probability of each behaviour is calculated, the behaviour with the maximum posterior probability is chosen.

$$b_t^* = \arg\max_{b_t} p(b_t \mid S_t, C_t, Z_t, U_t) \quad (18)$$

Greedy optimization of task completion probability is performed. The order of calculation for this equation is *O(NMK)*, which is significantly lower than the complexity of existing methods for action selection under uncertainty, like POMDPs, that typically have complexity exponential to the number of states. This allows the system to take decisions more often, in order to cope with fast changes and the occurrence of unpredictable events in the environment. The behaviour selection scheme is described in the next section in more detail.

5.1 Behaviour selection model

The term $p(b_t \mid b_{t-1}, c_t)p(c_t \mid x_t)$ in equation (18) is the behaviour model and it plays a crucial role in the optimality of the behaviour selection. It depends on the previous behaviour of the robot, the perceptual events that activate system behaviours and the estimated system state. This model supplies an expert opinion on the applicability of each behaviour at the present situation, indicating if it is completely forbidden, rather unwise, or recommended. This is done according to the information available to the system.

The results of state estimation are used to evaluate if behaviour triggering events have occurred and how certain their existence is. During this step the term $p(c_t \mid x_t)$ in (16) is calculated. Triggers and behaviours can have *high*, *medium*, *low* probability or be inactive. These values are predefined in this implementation and encode the goals of the system. They can also be acquired by letting a human operator decide about which behaviour the robot should choose, according to the situation. These decisions are then modelled to probability distributions. Bayesian decision theory and decision modelling provide the theoretical background to achieve that. Interesting works in this direction are (Ahmed & Campbell, 2008) and (Hy et al., 2004).

Three triggers exist that are used to calculate the probabilities of the behaviour model. These are:

- The existence of a person in the vicinity of the robot denoted by *person*. If a person has been detected then this trigger is activated. Its probability, $p(person \mid x_t)$, increases as the robot comes closer to a person.
- The existence of a goal for the robot to reach, which is given through interaction with people, denoted by *goal*. The probability $p(goal \mid x_t)$ increases as the distance of the given target from the current most likely, estimated robot position decreases.
- The existence of a loop closing opportunity, *loop*. It depends as explained in Section 4.4 on the existence of an entry point for loop closing and the difference between current position uncertainty and the initial position uncertainty at the entry point. The probability $p(loop \mid s_t)$ increases as the difference in uncertainty from the current position to the initial uncertainty at the entry point position becomes larger.

It remains now to explain how $p(b_t \mid b_{t-1}, c_t)$ is constructed. At each time step the robot knows its previous behaviour b_{t-1} and the triggers that are active. Using Table 1, behaviours are proposed as recommended and are assigned high probability. The rest of the behaviours that are possible receive lower recommendations and some are prohibited (denoted by "-" in the table). For example, if the previous behaviour of the robot, b_{t-1}, was *Loop Closing*, the trigger *loop* has probability *low* and the robot has no goal assigned, then the most recommended behaviour for the current time step, b_t, will be *Explore*. No other behaviour is possible.

b_t \ b_{t-1}	*Explore*	*Loop Closing*	*Approach*	*Reach Goal*
Explore	¬*person*	$p(loop\|x_t)<medium$ & ¬*goal*	¬*person* || $p(person\|x_t)<medium$	$p(goal\|x_t)<medium$
Loop Closing	$p(loop\|x_t)>medium$	$p(loop\|x_t)>low$	-	$p(loop\|x_t)>medium$
Approach	*person*	-	$p(person\|x_t)<high$	¬*goal* & *person*
Reach Goal	-	$p(loop\|x_t)<medium$ & *goal*	*Goal*	*Goal*

Table I. Behaviour Selection Model

A recommended behaviour is assigned *high* probability value and all other possible behaviours a *low* value. Finally values are normalized. If only one behaviour is possible as in the example given, then it receives a probability of 1. This way, $p(b_t \mid b_{t-1}, c_t, s_t)$ is acquired and is used to calculate the behaviour that maximizes (15).

6. Results

In order to evaluate the performance of the proposed behaviour selection mechanism, experiments were carried out. The robot was called to find its way to a given room of the third floor of our institute, without any prior knowledge of the environment. The floor plan as well as the starting position of the robot and the given target room is shown in Fig. 3. The robot must interact with people in order to ask for directions.

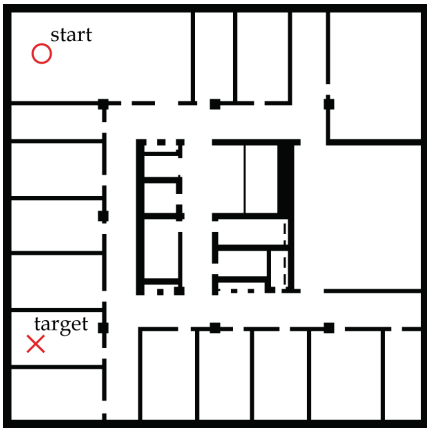


Fig. 3. Ground truth map of the third floor of the Institute of Automatic Control Engineering, Munich is illustrated. The robot starts without map knowledge and is required to reach the depicted final target location, which is acquired by interaction with humans.

All algorithms described in this paper have been implemented in C++ and have been tested on-line on the robot, using an AMD Athlon Dual Core 3800+ processor and 4GB of RAM. For the Rao-Blackwellized particle filter 200 particles were used and the conditional particle filters for people tracking used 30 particles each. Behaviour selection was performed at 1Hz. The SLAM and tracking module was running at 2Hz and the path planner at 1Hz. It has been found experimentally that at this frequency the tracker can track up to 15 moving objects.

In Fig. 4 the decisions taken by the robot in different situations during the experiment are illustrated. At first the robot decides to explore in order to acquire information about where the target room is. Two persons are detected and the robot decides to approach the one nearest to it in order to interact with. A goal position is acquired in the form of a waypoint "10m in the x direction and 3m in the y direction". The robot decides to reach this goal. After the intermediate goal is reached, a decision is made to explore in order to acquire new direction instructions. Another person is approached and new instructions are given which this time will lead to the final goal. As the robot moves its uncertainty grows. At some point an opportunity to close a loop is recognized. Therefore the robot decides to change its behaviour to *Loop Closing*, in order to reduce its uncertainty. After the loop is closed, the robot reaches its final goal.

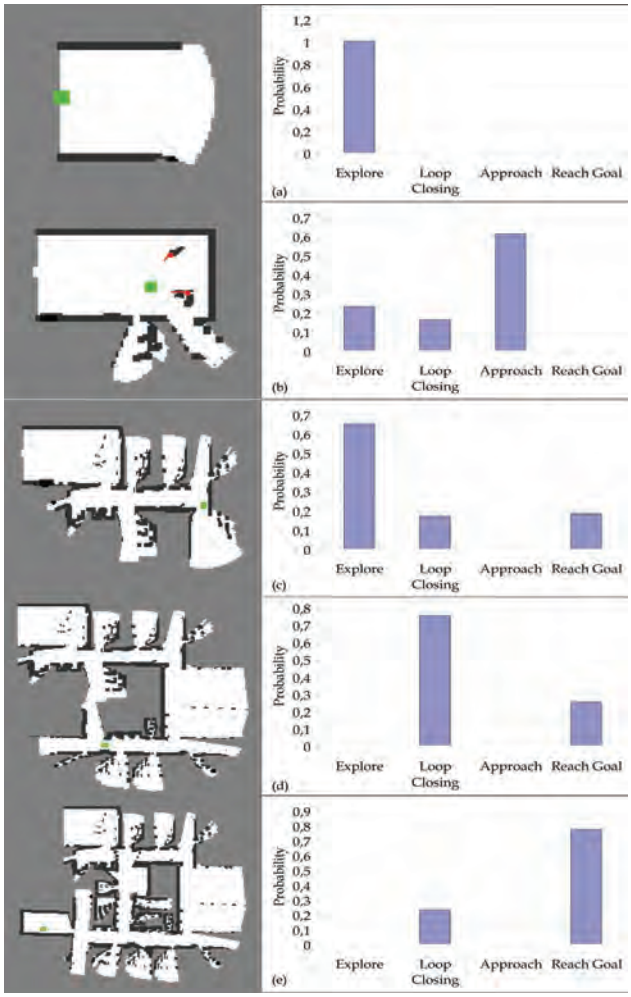


Fig. 4. The robot is called to find its way to a given goal, without prior map knowledge. All information is extracted by interaction. The decisions of the behaviour selection scheme are

shown in different situations. (a) The robot starts without any prior map information and decides to explore in order to find persons to interact with. (b) Two persons are found and the robot chooses the one closest to it in order to interact. (c) A goal was given to the robot by the first interaction and was reached by the robot. Now it chooses to explore in order to find a person to acquire a new target. (d) The robot has a target but its position uncertainty is high. It detects an opportunity to close a loop and decides to do so. (e) The robot reaches its final goal.

By taking uncertainty into account in action selection, the robot can anticipate unforeseen situations and increase the likelihood of achieving its goal. In Fig. 5 the overall uncertainty of the robot during this experiment is illustrated by the red line. The uncertainty of the robot trajectory when it reaches the target directly, without being controlled by the proposed scheme, is illustrated by the blue dashed line. It can be seen that at the early phases of the experiment the uncertainty of the system is larger with the proposed scheme, since the robot drives more complex trajectories in order to approach people, but it is not critical. At some point it decides to close the loop and its uncertainty is reduced notably. When it reaches its final goal the overall system uncertainty is much lower than without behaviour selection. Lower uncertainty is equivalent to safer navigation and increased task completion likelihood.

The presented system is capable of deciding when it should pursuit its given target, in which situation interaction with humans is needed in order to acquire new target information and finally when its overall uncertainty has reached a critical point. In this last case it tries to reduce it by taking actions that improve its state estimates.

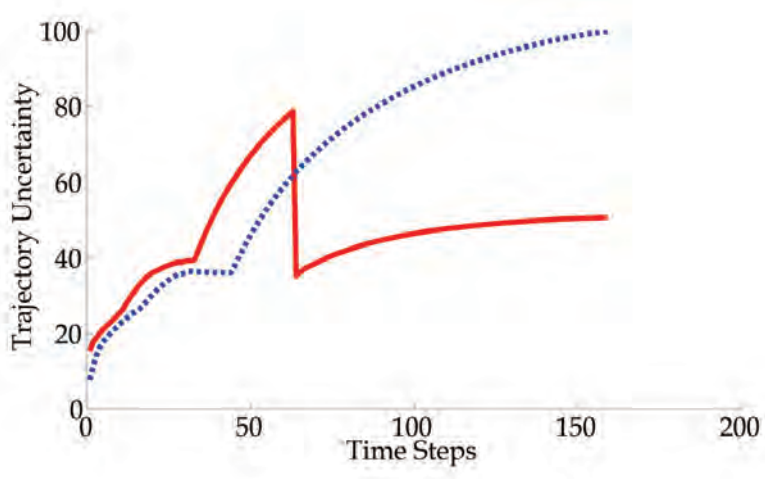


Fig. 5. Trajectory uncertainty as it evolves with the time. With red the uncertainty of the robot is illustrated, while it is controlled with the proposed behaviour selection scheme. The uncertainty of the robot trajectory when it reaches the target directly, without being controlled by the proposed scheme, is illustrated with blue dashed line.

7. Conclusion

In this Chapter a probabilistic framework has been introduced, that enables recursive estimation of a dynamic environment model and action selection based on these uncertain estimates. The proposed approach addresses two of the main open challenges of action selection. Uncertain knowledge is expressed by probability distributions and is utilized as a basis for all decisions taken from the system. At the same time the complexity of the proposed action selection mechanism is kept lower than of most state-of-the-art algorithms.
The probability distributions of all associated uncertain quantities are approximated effectively and no restrictive assumptions are made regarding their form. More specifically, a Rao-Blackwellized particle filter (RBPF) has been deployed to address the SLAM problem and conditional particle filters have been modified to be utilized with incrementally constructed maps for tracking people in the vicinity of the robot. This way a complete model of dynamic, populated environments is provided. The computational costs depend only on the required approximation accuracy and can be defined according to the requirements of the application domain.
The estimated uncertain quantities are used for coordinating the behaviours of the robot so that uncertainty is kept under control and the likelihood of achieving its goals is increased. A greedy optimization algorithm is used for behaviour selection, which is computationally inexpensive. Therefore the robot can decide quickly in order to cope with its rapidly changing environment. The decisions taken may not be optimal in the sense of POMDP policies, but are always responding to the current state of the environment and are goal oriented. The goals of the system are expressed by the behaviour selection model.
Results from the implementation of all proposed algorithms on the ACE robotic platform demonstrate the efficiency of the approach. The robot can decide when to pursue its given goal or when to interact with people in order to get more target information. If its uncertainty becomes large, it takes actions that improve its state estimates. It is shown that overall system uncertainty is kept low even if the robot is called to complete complex tasks.
Human decision making capabilities are remarkable. Therefore, future work will focus on learning the behaviour selection model from data provided by a human expert. This way the quality of the decisions taken by the system can be improved. Formal evaluation criteria for action selection mechanisms need to be developed. This is challenging since such criteria must consider many conflicting requirements and since in almost every study different physical robots are used in variable experimental conditions. Finally, more experiments are going to be conducted in unstructured, outdoor, dynamic environments.

8. Acknowledgements

This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems -- CoTeSys*, see also *www.cotesys.org*.

9. References

Ahmed, N.; Campbell, M. (2008). Multimodal Operator Decision Models. *IEEE American Control Conference (ACC),* Seattle, USA.

Arkin, R.C. (1998). Social behavior. *Behavior-Based Robotics*, MIT Press, Cambridge, MA.

Arulampalam, S.; Maskell, S.; Gordon, N.; Clapp, T. (2002). A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing,* 50, p. 174-188.

Bryson, J.J.; Stein, L.A. (2000). Architectures and Idioms: Making Progress in Agent Design. In: *The Seventh International Workshop on Agent Theories, Architectures and Languages*. Springer.

Dissanayake, G.; Newman, P.; Clark, S.; Durrant-Whyte, H.; Csorba, M. (2001). A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *IEEE Transactions on Robotics and Automation,* 17(3), p. 229-241.

Doucet, A.; de Freitas, J. F. G.; Gordon, N. J. (2000). Sequential Monte Carlo Methods in Practice. *Springer-Verlag*, New York.

Emken, J. L.; Benitez, R.; Sideris, A.; Bobrow J. E.; Reinkensmeyer D.J. (2007). Motor Adaptation as Greedy Optimization of Error and Effort. *Journal of Neurophysiology*. p. 3997-4006.

Grisetti, G.; Stachniss, C.; Burgard, W. (2005). Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling *International Conference of Robotics and Automation (ICRA)*, Barcelona, Spain.

Hähnel, D.; Triebel, R.; Burgard, W.; Thrun, S. (2003). Map Building with Mobile Robots in Dynamic Environments. *Proceedings of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Taipei, Taiwan.

Humphrys, M. (1997). Action Selection Methods Using Reinforcement Learning. *PhD Thesis*. University of Cambridge, Computer Laboratory, Cambridge, England.

Hy, R. L.; Arrigoni, A.; Bessiere, P.; Lebeltel, O. (2004). Teaching Bayesian behaviours to video game characters. *Robotics and Autonomous Systems*, 47, p. 177-185.

Körding, K.; Wolpert, D. (2006). Bayesian Decision Theory in Sensorimotor Control. *Trends in Cognitive Sciences*, 10(7), p. 319-326.

Lidoris, G.; Klasing, K.; Bauer, A.; Xu, T.; Kühnlenz, K.; Wollherr, D.; Buss, M. (2007). The Autonomous City Explorer Project: Aims and System Overview. *Proceedings of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, USA.

Lidoris, G.; Wollherr, D.; Buss, M. (2008). Bayesian State Estimation and Behavior Selection for Autonomous Robotic Exploration in Dynamic Environments. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, Sept, 22-26.

Littman, M.; Cassandra, A.; Kaelbling, L. (1995). Learning Policies for Partially Observable Environments: Scaling Up. *Proceedings of the 12th Intl. Conf. on Machine Learning*. p. 362-370, San Fransisko, USA.

Maes, P. (1989). How to do the right thing. *Connection Science Journal*, 1(3): p. 291-323.

Miller, I.; Campbell, M. (2007). Rao-Blackwellized Particle Filtering for Mapping Dynamic Environments. *IEEE International Conference on Robotics and Automation (ICRA),* Rome, Italy.

Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. (2002). FastSLAM: A Factored Solution to Simultaneous Localization and Mapping. *National Conf. on Artificial Intelligence (AAAI)*, Edmonton, Canada.

Montemerlo, M.; Whittaker, W.: Thrun, S. (2002). Conditional Particle Filters for Simultaneous Mobile Robot Localization and People-Tracking. *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Washington, DC, USA.

Moravec, H. (1989). Sensor fusion in certainty grids for mobile robots. *Sensor Devices and Systems for Robotics,* p. 243-276.

Murphy, K. (1999). Bayesian map learning in dynamic environments. *Advances in Neural Information Processing Systems (NIPS).* MIT Press, p. 1015-1021.

Pirjanian, P. (1999). Behavior coordination mechanisms -- state-of-the-art. *Technical Report IRIS-99-375*, Institute of Robotics and Intelligent Systems, School of Engineering, University of Southern California.

Prescott, T.J.; Redgrave P.; Gurney, K. (1999). Layered control architectures in robots and vertebrates. *Adaptive Behavior*, 7:99-127.

Rohrmüller, F.; Althoff, M.; Wollherr, D.; Buss, M. (2005). Probabilistic Mapping of Dynamic Obstacles Using Markov Chains for Replanning in Populated Environments. *IEEE Intl. Conf. on Robotics and Automation (IROS)*, Nice, France, Sept, 22-26.

Stachniss, C.; Haehnel, D.; Burgard, W. (2004). Exploration with Active Loop-Closing for FastSLAM. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Sendai, Japan.

Stachniss, C.; Grisetti, G.; Burgard, W. (2005). Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. *Robotics: Science and Systems (RSS)*, Philadelphia, USA, p. 65-72.

Thrun, S.; Liu, Y.; Koller, D.; Ng, A.; Ghahramani, Z.; Durrant-Whyte, H. (2004). Simultaneous Localization and Mapping with Sparse Extended Information Filters. *Int.J. Robotics Research,* 23(7-8), p. 693-716.

Tyrrel, T. (1993). Computational Mechanisms for Action Selection. *PhD Thesis*, University of Edinburgh.

Wang, C.; Thorpe, C.; Thrun, S. (2003). Online Simultaneous Localization And Mapping with Detection And Tracking of Moving Objects: Theory and Results from a Ground Vehicle in Crowded Urban Areas. *IEEE Int. Conf.on Robotics and Automation (ICRA)*, Taipei, Taiwan.

Yamauchi, B. (1998). Frontier-based Exploration Using Multiple Robots. *Second Intl. Conference on Autonomous Agents*, Minneapolis, USA.

7

Efficient Multi-User Parallel Greedy Bit-Loading Algorithm with Fairness Control For DMT Systems

Cajetan M. Akujuobi and Jie Shen
The Center of Excellence for Communication Systems Technology Research (CECSTR), Prairie View A&M University, Prairie View, Texas 77446 USA

1. Introduction

This chapter addresses the multi-user bit-loading algorithm for discrete multitone (DMT) modulation in digital subscriber line (DSL) systems. The widely deployed asymmetric digital subscriber line (ADSL) provides the high bit rate data transmission as well as plain old telephone service (POTS) on a single twisted-pair at the same time. DMT, the core of DSL systems, divides the frequency-selective channel into large number of narrow subchannels. If the number is large enough, each subchannel becomes flat in frequency response, although the responses may differ a lot among subchannels. One of the advantages of DMT is that the power spectral density (PSD) and bits allocated to each subchannel could be chosen according to the subchannel signal-to-noise ratio (SNR) in order to obtain the optimal performance (e.g. maximum data rate, or minimum power consumption). This process is called bit loading and is a critical issue in the design of DMT systems.

In the early days of DMT development, bit loading was studied only in single-user case, where only one pair of modems (transmitter and receiver) was considered. Compared with traditional telephone service, DSL systems always work on high frequency range, which causes the crosstalk interference among the twisted pairs in the same cable noticeable. The SNR of a subchannel is related not only with the PSD of its own transmitter, but also with the PSD of all other transmitters in the same cable that act as disturbers. The bit-loading algorithms need to be extended to multi-user scenario to obtain the global optimum performance among all users. The optimal algorithm for discrete multi-user bit loading is a natural extension of single-user greedy algorithm. A matrix of cost is calculated, with elements that represent the power increment to transmit additional bits for each subchannel. Then, the subchannel with minimum cost is found, and additional bits are assigned to it. The process continues until all subchannels are filled. A drawback of the multi-user greedy bit loading is the computation complexity. For a single iteration of the algorithm, only one subchannel on one user who has the minimum cost is selected to get additional bits.

The objective of this chapter is to propose an efficient greedy bit-loading algorithm for multi-user DMT systems. An improved parallel bit-loading algorithm for multi-user DMT will be discussed. The new algorithm is based on multi-user greedy bit loading. In a single

iteration, the bits were allocated to multiple users on the subchannels with same frequency. Thus, the number of iterations to allocate bits to all users decreased significantly. The adjustable cost elastic coefficient defined the range of power cost. In the bit-loading iteration, all subchannels for different users that have the power cost within the range have the chance to get additional bits assigned to them.

As a consequence of the greedy bit-loading algorithm, the user with better channel condition, which means it has smaller cost to transmit additional bits, will have more chance to get bits assigned to it until it meets the target data rate or exceeds the power budget. The user with worse channel condition is sacrificed in order to gain the maximum total data rate. However, in most real networks, users in the same cable are of equal priority. Their service quality is supposed to be as equal as possible. Fairness should be considered in the design of bit-loading algorithm. This chapter studied the possibility to improve the fairness among users in the same cable. We proposed a fairness control method in the bit-loading algorithm. A fairness coefficient is introduced so that the variance of total number of bits for all users could be controlled. The cost of fairness is that the better-condition loops have to reduce their data rate, because the worse-condition loops have little improvement in their data rate.

2. Bit loading algorithms for DMT

The application of discrete multi-tone (DMT), divides the frequency selective channel into large number of narrow subchannels, so that each subchannel could be used to modulate a fraction of information in parallel independently. The advantage of DMT is that the power spectral density (PSD) of transmitted signal could be chosen according to the subchannel signal-to-noise ratio (SNR) in order to obtain the optimal performance (e.g. maximum data rate, or minimum power consumption). The problem of how to allocate energy or information (bits) into subchannels optimally is a critical issue in the design of DMT systems. This problem is called bit loading. Many algorithms have been studied in the past for DMT systems such as (Akujuobi, C. M. & Shen, J. 2006), (Yu &Cioffi, 2001), (Hughes-Hartogs, 1987-1989), (Sonalker & Shively, 1998), (Chow & Cioffi, 1995), (Tu & Ciofi, 1990), (Leke & Cioffi, 1997) and (Campello, 1999). They are the foundations for the work discussed in this chapter. A review of several typical bit-loading algorithms are discussed in this Section.

2.1 Channel capacity

In information theory, the capacity of a band-limited white noise channel is: (Cover & Thomas, 1991)

$$C = \frac{1}{2}\log_2\left(1+\frac{P}{N_0 W}\right) \text{ bits per sample} \tag{1}$$

where the power of signal is P watts, the spectral density of white noise is $N_0/2$ watts/Hz, and the bandwidth is W Hz. The unit of capacity in Equation (1) is bits per sample, if we convert the unit to bits per second, the expression is the famous channel capacity formula proved by Claude Shannon in 1948 (Shannon, 1948) as shown in Equation (2):

$$C = W\log_2\left(1+\frac{P}{N_0 W}\right) \text{ bits per second} \tag{2}$$

In DMT systems, QAM is used as modulation method to map digital information into complex numbers. And we know that QAM is a two-dimensional modulation method, which means it has two basis functions as in-phase function and quadrature function. Therefore, the channel capacity in QAM DMT is

$$C_n = \log_2\left(1 + SNR_n\right) \tag{3}$$

where SNR_n refers to the signal-to-noise ratio for subchannel n, and C_n refers to the capacity of subchannel n. Channel capacity is the theoretic upper limit of achievable data rate for a channel with probability of error that tends to zero. In practical analysis, the probability of error can never be zero; instead, we expect an acceptable error probability P_e at some practical data rate. The reduced data rate could be expressed in a revised channel capacity formula by introducing a SNR gap Γ.

$$b_n = \log_2\left(1 + \frac{SNR_n}{\Gamma}\right) \tag{4}$$

When Γ = 1 (0 dB), b_i becomes the channel capacity. The selection of Γ depends on the error probability P_e and coding scheme. Higher P_e requires larger Γ. Complex coding scheme that guarantees reliable transmission can reduce the Γ. For the two-dimensional QAM system with bit error rate (BER) at 10^{-7}, the gap Γ is computed using the following formula: (Chow et al., 1995)

$$\Gamma = 9.8 + \gamma_m - \gamma_c \quad (dB) \tag{5}$$

where γ_m is the performance margin and γ_c is the code gain. If the system is uncoded (γ_c = 0dB) and performance margin is 0 dB, the gap is 9.8 dB.

3. Review of bit-loading algorithms

3.1 Water-filling algorithm

Water-filling algorithm is demonstrated in (Cover & Thomas, 1991) and (Gallager, 1968) as the optimal solution for the problem that distributes energy into parallel independent Gaussian channels with a common power constraint. Expand the SNR_n in Equation (4) to the ratio of received signal power $P_n H_n$ and noise power N_n, where P_n is the power allocated to subchannel n and H_n is the channel gain. The number of bits that transmits in a subchannel is expressed as in Equation (6).

$$b_n = \log_2\left(1 + \frac{P_n H_n}{\Gamma N_n}\right) \tag{6}$$

The problem of bit loading is an optimization problem that allocates power to subchannels. The target of the optimization is to maximize the aggregate number of bits transmitted on all N subchannels under the constraint that the total power should not exceed the power budget P.

$$\begin{aligned} &\text{maximize} \quad \sum_{n=1}^{N} b = \sum_{n=1}^{N} \log_2\left(1 + \frac{P_n H_n}{\Gamma N_n}\right) \\ &\text{subject to} \quad \sum_{n=1}^{N} P_n \le P \end{aligned} \tag{7}$$

Employ Lagrange multipliers method to solve the optimization problem of Equation (7) with the Lagrangian function $L(P_n)$ as:

$$L(P_n) = \sum_{n=1}^{N} \log_2 \left(1 + \frac{P_n H_n}{\Gamma N_n}\right) + \lambda \left(\sum_{n=1}^{N} P_n - P\right) \tag{8}$$

The λ in Equation (8) is called Lagrangian Multiplier. Take the derivative on $L(P_n)$ over the variable P_n and make it equal to 0,

$$\frac{\partial L}{\partial P_n} = \frac{1}{\ln 2 \left(1 + \frac{P_n H_n}{\Gamma N_n}\right)} \cdot \frac{H_n}{\Gamma N_n} + \lambda = 0 \tag{9}$$

Solve this equation to get the power allocation P_n as:

$$\begin{gathered} P_n = -\frac{1}{\lambda \ln 2} - \Gamma \frac{N_n}{H_n} = C_\lambda - \frac{\Gamma}{g_n} \\ \left(\text{where } C_\lambda = -\frac{1}{\lambda \ln 2}, \quad g_n = \frac{H_n}{N_n}\right) \end{gathered} \tag{10}$$

This equation could be rearranged to the following form:

$$P_n + \frac{\Gamma}{g_n} = \text{constant } C_\lambda \tag{11}$$

The variable g_n in Equation (11) expresses the signal-to-noise ratio when unit power is transmitted on subchannel n. It is a unified version of SNR, which is a measurement to indicate the quality of subchannels. We can see from Equation (11) that P_n has the form of "water-filling" distribution. That is, the summation of power transmitted in subchannel n and inverse unified signal-to-noise power ratio (multiplied by Γ) must equal to a constant (C_λ). The $\frac{\Gamma}{g_n}$ can be understood as the terrain of a bowl, power is poured into the bowl like water. The value of C_λ represents the resulting water level. Fig. 1 shows the idea of water-filling.

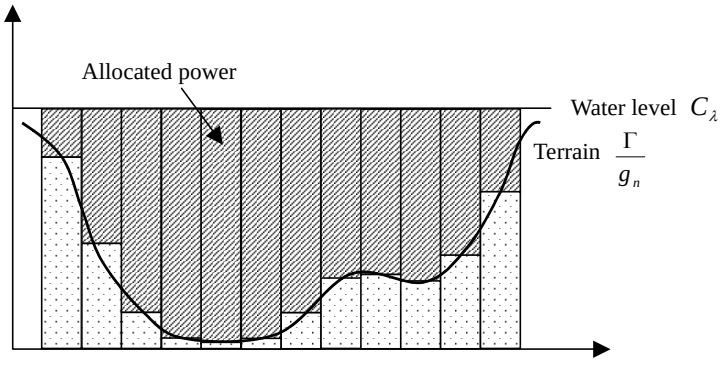


Fig. 1. Water-Filling

The formula to obtain the value of C_λ comes from the constraint of total power budget.

$$P = \sum_{n=1}^{N} P_n = \sum_{n=1}^{N} \left(C_\lambda - \frac{\Gamma}{g_n} \right) = NC_\lambda - \Gamma \sum_{n=1}^{N} \frac{1}{g_n} \tag{12}$$

So we get

$$C_\lambda = \frac{1}{N} \left(P + \Gamma \sum_{n=1}^{N} \frac{1}{g_n} \right) \tag{13}$$

The water-filling algorithm described in (Starr et al., 1999) and (Tu & Cioffi, 1990) starts from calculating the water-level C_λ, then calculate the power allocation by using,

$$P_n = C_\lambda - \frac{\Gamma}{g_n} \tag{14}$$

The subchannels are sorted in descendent order with respective to the value of g_n.
Therefore, if in any step the calculated P_n is negative, which means the subchannel's terrain is too high (signal-to-noise ratio is too small) to hold the power, the algorithm stops. The problem of water-filling algorithm is that in DMT systems, the bit loading for subchannels should be discrete numbers instead of arbitrary real numbers, which is assumed in this algorithm.

3.2 On/Off algorithm – chow's algorithm

The algorithm described in (Chow & Cioffi, 1995) utilizes the fact that if the same or nearly same subchannels are used, the difference of bit-loading result is very small (less than 2%, (Leke & Cioffi, 1997) between the proposed "on/off" algorithm and the traditional water-filling algorithm. In Chow's on/off algorithm, the power distribution is flat, that is, the power is same over all the selected subchannels. On the subchannels that are not selected, the power is simply set to zero, which means they are turned off.
The algorithm starts by sorting the SNRs in descendent order, so that the first subchannel been processed has the highest SNR. At the beginning, the number of subchannels turned on is zero. During each step, one more subchannel is turned on, which causes the total number of turned on subchannels to be K. The power allocated to each turned on subchannel is set to be

$$P_n = \frac{P_{budget}}{K}, \quad (n = 1 \cdots K) \tag{15}$$

where P_{budget} is power constraint. If the P_n is greater than the power mask at subchannel n, the power mask is used as P_n. With the power allocation from Equation (15), numbers of bits in subchannels 1 to n are then calculated using Equation (6). If the aggregated number of bits over all used subchannels becomes less than the value in previous step, the algorithm stops, and the bit allocation scheme obtained from previous step is used as the bit loading result. All the remaining subchannels are thus left in the off state.

The flat power allocation in Chow's algorithm satisfies the requirement of static spectrum management of ADSL power spectral density. The reason why flat power allocation causes very small difference from optimal water-filling algorithm is studied in detail in (Yu & Cioffi, 2001). The answer is because the logarithm operation in (6) is insensitive to the power actually allocated to subchannels, unless the SNR is small. If the subchannels with SNR less than a cut-off value are turned off, the power could be simply allocated to all other subchannels evenly without loss of much accuracy. An even simpler algorithm was proposed in (Yu & Cioffi, 2001), that save the complexity to find the cut-off point of SNR. When the cut-off SNR is found, power allocation is just to assign constant power to subchannels that has SNRs greater than cut-off value, and assign zero power to other subchannels.

3.3 Greedy algorithm – hughes-hartogs algorithm

In Hughes-Hartogs's patent (Hughes-Hartogs, 1987-1989), an algorithm based on greedy idea was proposed. That is, every incremental power to transmit one additional bit is allocated to the subchannel that can use it most economically. For example, assume considering only two subchannels A and B. Subchannel A bears N_A bits now, and the incremental power required to transmit N_A+1 bits is ΔP_A. For subchannel B that bears N_B bits, the incremental power is ΔP_B. If $\Delta P_A < \Delta P_B$, subchannel A will be selected to transmit the additional bit and gets the incremental power allocation. The power requirement for all subchannels to transmit all possible number of bits could be calculated in advance, and be saved in a matrix P as show in Fig. 2. The element $P_{m,n}$ in the matrix represents the power needed to transmit m bits in subchannel n. The values in first row are zeros obviously.
The incremental power required to transmit one additional bit is calculated by subtracting the first row from the second row. The result ΔP is same as the second row in the first iteration. The subchannel n that has minimum value in ΔP is selected to get the additional bit. In the next iteration, the elements of column n in matrix P are shifted upward for one position. The following subtractions are still performed on the row one and row two, until the power budget is fully consumed.

Subchannels	1	2	…	N
0 bit	0	0	…	0
1 bit	$P_{1,1}$	$P_{1,2}$		$P_{1,N}$
2 bits	$P_{2,1}$	$P_{2,2}$		$P_{2,N}$
…	…			
M_{max} bits	$P_{M,1}$	$P_{M,2}$		$P_{M,N}$

Fig. 2. Power Matrix for Hughes-Hartogs Algorithm

The Hughes-Hartogs algorithm was first invented for voice-band modem in 1987, ((Hughes-Hartogs, 1987-1989). For the modern DSL systems, the number of subchannels is usually much larger than the voice-band modems. The slow convergence rate of Hughes-Hartogs algorithm and some other constraints, such as the fixed SNR assumption, make this algorithm impractical in DSL systems. But it is still a good starting point for later improved algorithms.

3.4 Bit removal greedy algorithm

Compared with the bit filling greedy algorithm described in Section 3.3, a bit removal greedy algorithm was proposed in (Sonalker & Shively, 1998). Reversing Equation (6) we can get the power that is required to transmit b_n bits in subchannel n as,

$$P_n[b_n] = \left(2^{b_n} - 1\right)\frac{\Gamma N_n}{H_n} = \left(2^{b_n} - 1\right)\alpha_n \tag{16}$$

Therefore, the amount of power saved if one bit is removed from the subchannel is

$$\Delta P_n^R = P_n[b_n] - P_n[b_n - 1] = \left[\left(2^{b_n} - 1\right) - \left(2^{b_n - 1} - 1\right)\right]\alpha_n = 2^{b_n - 1}\alpha_n \tag{17}$$

The bit removal algorithm first allocates the maximum possible numbers of bits to all subchannels. The maximum number of bits in a subchannel is determined by either the power mask limit or the upper limit of allowable bit number – whichever is smaller. Most likely, this bit-loading scheme will exceed the power budget, and the total bits number will be greater than the desired target. Then, the bits are removed one bit per time from the subchannel that may save the power most significantly by removing it. The removing process stops until the power constraint is satisfied, or the data rate requirement is satisfied. Authors (Sonalker & Shively, 1998) made a computation load comparison between bit-removal and bit-filling greedy algorithms over several loops. It showed that in most cases, the bit-removal algorithm required much fewer computations.

3.5 Peter, chow, cioffi, and bingham's practical algorithm

Chow, Cioffi, and Bingham proposed a practical DMT loading algorithm based on the rounded bit number and performance margin adjustment in (Chow et al., 1995). The idea is to make a round operation on the resulting bit loading value, which is expressed as

$$b(n) = \log_2\left(1 + \frac{SNR(n)}{\Gamma + \gamma_{\text{margin}}(dB)}\right) \tag{18}$$

where γ_{margin} represents the performance margin, and has the initial value of 0. The zero value of γ_{margin} causes the $b(n)$ to get the maximum value. $b(n)$ is then rounded to the nearest integer value $\hat{b}(n)$. In regular condition, the summation of rounded values $\hat{b}(n)$, n from 1 to N, will exceed the target total bit number. In next step, the algorithm increases the margin by using the formula:

$$\gamma_{\text{margin}} = \gamma_{\text{margin}} + 10\log_{10}\left(2^{\frac{B_{\text{total}} - B_{\text{target}}}{\text{UsedCarriers}}}\right) \tag{19}$$

With the updated γ_{margin}, Equation (18) is calculated and rounded again. The process continues until the total number of bits reaches the target. If the process doesn't converge after the maximum number of iterations, the algorithm forces the convergence by adjusting the bit allocation according to the difference between $b(n)$ and $\hat{b}(n)$.

At last, the algorithm makes further adjustment on energy distribution so that the bit error rates (BER) on all used subchannels are equal. The analysis in (Chow et al., 1995) shows that only 10 iterations is enough for ADSL loops, which is much faster than the Hughes-Hartogs algorithm.

4. Efficient greedy bit loading with fairness control for multi-user DMT

The twisted-pairs inside a cable could be imagined as a multi-user communication channel because of crosstalk coupling. The received signal power of any user depends on the peering transmitted power, and at the same time is impaired by additive white Gaussian noise (AWGN) and the crosstalk from all other transmitters in the same cable. Fig. 3 shows the multi-user channel environment, where H_{ii} represents the main channel transfer function; H_{ij} ($i \neq j$) represents the crosstalk transfer function from user i to user j; σ_i represents the power of AWGN noise. Under DMT scenario, we need to study a step further besides the total power of each user. The power allocation over frequency spectrum of each user is of interest.

The power allocation problem relates with the bit-loading problem closely. Actually, they are the two aspects of a same problem, because the number of bits that can be transmitted on a subchannel is a function of the power allocated to that subchannel. Equation (6) gives the relationship between them. In Section 3, we gave a review of several major bit-loading algorithms. However, all those algorithms we discussed are applied to a single modem (or single user). In crosstalk environment, bit-loading algorithms need to be extended to consider mutual effects between multiple users so that the global optimal performance among all modems (users) in a cable could be obtained.

In Section 4, we explore the current available multi-user bit-loading algorithms, and then propose an improved efficient algorithm, which allocates bits to multiple users in parallel on subchannels that have same frequency. This new algorithm reduces the number of iterations dramatically. A new fairness coefficient is also introduced to improve the fairness of data rate among users.

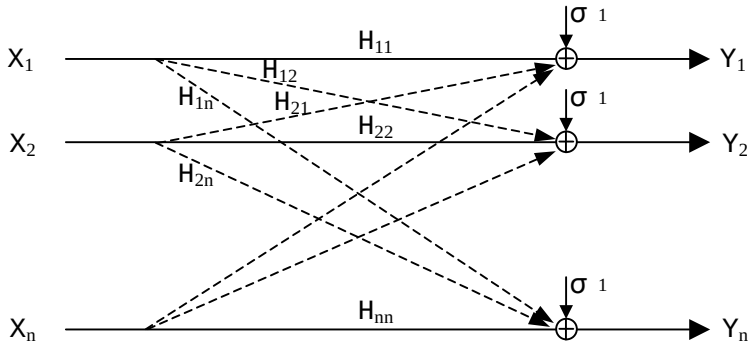


Fig. 3. Multi-user Channel With Crosstalk among Loops

4.1 Notation

The notations that are used in the later Sections of this chapter are defined here. As shown in Figure 4, there are M users in the same cable in which interference exists between users.

We use variables i or j (i, j = 1, … M) to indicate the indices of users. In DMT based systems, each user has N sub-channels. The index of sub-channel is indicated by variable n (n = 1, …, N). We use the term "subchannel (i, n)" to refer to the subchannel n of user i, where i = 1…M; n = 1…N. More variables are listed as below:

$b_i(n)$: Number of bits allocated to subchannel (i, n).

$\mathbf{b}(n) = [b_1(n), b_2(n) \cdots b_M(n)]^T$: The vector of bit allocations at subchannel n for M users.

$P_i(n)$: Power allocated to subchannel (i, n).

$\mathbf{P}(n) = [P_1(n), P_2(n) \cdots P_M(n)]^T$: The vector of power allocation at subchannel n for M users.

$H_{ij}(n)$: Channel gain transfer function from user i to user j at the subchannel n. When $i = j$, $H_{ii}(n)$ is the insertion loss transfer function for user i at subchannel n (ANSI Std. T1.417, 2001). When $i \neq j$, $H_{ij}(n)$ is the crosstalk transfer function from user i to user j at the subchannel n.

Γ : SNR gap margin to ensure the performance under unexpected noise. It is a function of target BER, modulation scheme and coding scheme.

$\sigma_i^2(n)$: The variance of white Gaussian noise on subchannel (i, n).

$P_{mask}(n)$: The power mask on subchannel n.

$P_{budget}(i)$: The power budget for user i.

Δb : The incremental unit of bits added to a subchannel in each iteration. In general, it should be an integer number.

T_{symbol} : The symbol period of the DMT system. For ADSL, it equals to 1/4000.

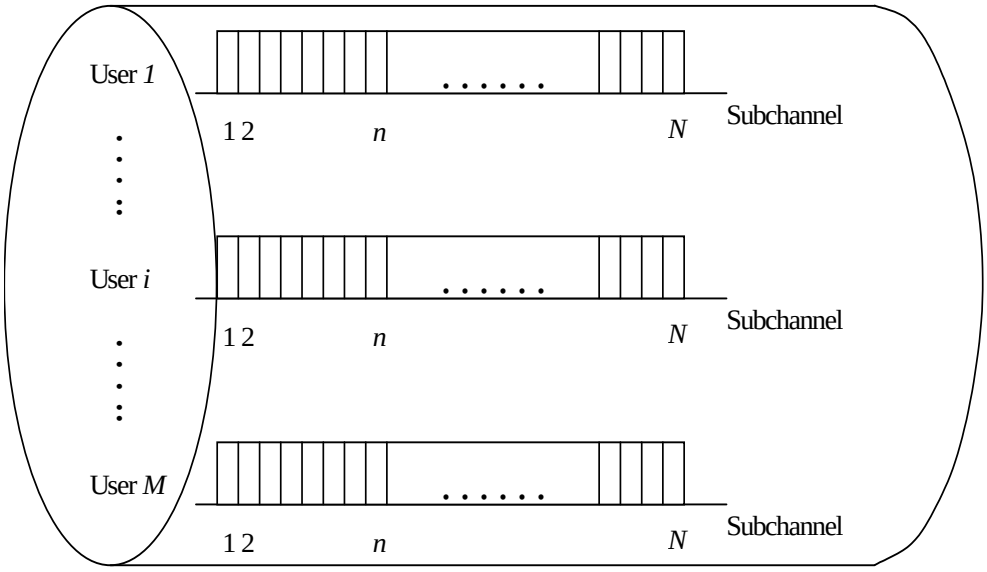


Fig. 4. Multi-User Multi-Channel Configuration

4.2 Problem definition for multi-user bit loading

Like single user bit-loading problem, the problem for multi-user bit loading is an optimization problem. The purpose of the optimization is to find the bit allocation schemes $\mathbf{b}(n)$ for all M users on each subchannel n (n = 1…N). The objective is to maximize the aggregate data rate of all users

$$\text{maximize } R_{total} = \sum_{i=1}^{M} R_i = \frac{1}{T_{symbol}} \sum_{i=1}^{M} \sum_{n=1}^{N} b_i(n) \tag{20}$$

with the constraints of power and bit limits, such as:

$$0 \le P_i(n) \le P_{mask}(n) \tag{21}$$

$$\sum_{n=1}^{N} P_i(n) \le P_{budget}(i) \tag{22}$$

$$b_{\min} \le b_i(n) \le b_{\max}, \text{ and } b_i(n) \text{ is integer} \tag{23}$$

This is called "rate-adaptive loading" (Starr et al., 1999). In some cases, what we care about is not to get the maximum data rate, but to minimize the power consumption with a fixed data rate. The second problem is called "margin-adaptive loading". Formulated as in Equation (24):

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{M} P_i = \sum_{i=1}^{M} \sum_{n=1}^{N} P_i(n) \\ &\text{subject to} \quad b_i = b_{target} \qquad (i = 1 \cdots M) \end{aligned} \tag{24}$$

These two kinds of problems are equivalent in that algorithms designed for one problem could be applied similarly to another one. In this thesis, we concentrate on the rate-adaptive problem, that is, to maximize the total data rate. As a consequence of maximizing total data rate over all users, the user with better channel condition, which means it has smaller cost to transmit additional bits, will have more chance to get bits assigned to it until it meets the target data rate or exceeds the power budget. The user with worse channel condition is sacrificed in order to gain the maximum total data rate.

However, in most real networks, users in the same cable are of equal priority. They pay the service provider the same fee to get broadband Internet access. Their service quality is supposed to be as equal as possible. Fairness should be considered in the design of bit-loading algorithm. Thus, the multi-user bit loading becomes a multi-objective problem. On one hand, we want maximum total data rate (or equivalently, the minimum power consumption with given target data rate). On the other hand, we want to minimize the difference of data rate among users. Therefore, we defined the second objective as to minimize the variance of data rate among users.

$$\text{minimize } Var(R_i) = \sum_{i=1}^{M} \left(R_i - \overline{R} \right)^2 = \sum_{i=1}^{M} \left(R_i - \frac{1}{M} R_{total} \right)^2 \tag{25}$$

It is clear that the two objectives contradict each other. One direct effect of minimizing the total data rate difference is that the best-condition user cannot obtain its highest data rate as

it can in single user algorithm. They cannot be achieved at the same time. So there must be a tradeoff between them.

4.3 Current multi-user bit-loading algorithms

Distributed iterative water-filling algorithm employs the single-user water-filling method to allocate power iteratively, user by user, until all users and subchannels are filled (Yu et al., 2002). The algorithm runs in two embedded loops. The outer loop looks for the optimal power constraint on each user by increasing or decreasing the power budget for the user, then the inner loop is called to calculate the power allocation under the given power constraint. If the result of inner loop gives data rate lower than target rate, total power will increase, and vice versa. The inner loop employs iterative water-filling method to get optimal power allocation for all the users. The problem of this algorithm is that if the target rate is not appropriate, this algorithm cannot converge. The question then switches to how to obtain the set of achievable target rates for each user. In the coordination of level 1 of DSM, a central agent with knowledge of all channel and interference transfer function exists and is able to calculate the achievable target rates. So, this algorithm is not totally autonomous, some kind of central control is required.

Iterative constant power (ICP) transmission, a slightly variation of iterative water-filling (IW) algorithm is proposed in (Yu & Cioffi, 2001). Both algorithms have the similar two-stage structure. The difference lies in the inner loop: only constant value or zero value of power is allowed in ICP, while continuous power value is used in IW. Both of these two algorithms are suboptimal, but easy to deploy because there is no coordination among users.

The optimal algorithm for discrete multi-user bit loading is a natural extension of single-user greedy algorithm. In the extended greedy algorithm, a matrix of cost is calculated. The elements in the matrix represent the power increment to transmit additional bits for each subchannel and each user. Then, the subchannel in a specific user with minimum cost is found, and additional bits are assigned to it. The process continues until all the power has been allocated. This algorithm is illustrated in (Lee et al., 2002).

A drawback of the multi-user greedy bit loading is the computation complexity. For a single iteration of the algorithm, only one subchannel on one user who has the minimum cost is selected to get additional bits. In each iteration step, the most time consuming calculation is to solve the linear equations to get power allocated to subchannels with specified bits allocation in order to updated the cost matrix. The number of subchannels in a DSL system is usually large, for example, in ADSL there are 223 subchannels for downstream (ANSI Std. T1.417, 2001). If the average number of bits assigned to a subchannel is 10, and there are 50 users in a cable, the total number of iterations that is required to allocate all bits is above 10^5.

4.4 Formulation of the problem

Before introduce the efficient greedy bit loading with fairness, we first formulate the bit-loading problem for multi-user DMT systems with the objectives of maximizing aggregate data rate (Equation (20)) and minimizing the data rate variance among users (Equation (25)). By extending the single user bit loading to multi-user case, the noise that appears at the

receiver is the summation of AWGN and crosstalk from all other users in the same cable (Fig. 3),

$$N_i(n) = \sigma_i^2(n) + \sum_{\substack{j=1 \\ j \neq i}}^{M} P_j(n) \left| H_{ji}(n) \right|^2 \tag{26}$$

Substitute Equation (26) into Equation (6) to replace the variable N_n, we get the multi-user bit loading expression as

$$b_i(n) = \log_2 \left(1 + \frac{P_i(n) \left| H_{ii}(n) \right|^2}{\Gamma \left(\sigma_i^2(n) + \sum_{\substack{j=1 \\ j \neq i}}^{M} P_j(n) \left| H_{ji}(n) \right|^2 \right)} \right) \tag{27}$$

The aggregate data rate of user i is the summation of bits transmitted over all subchannels divided by symbol period,

$$R_i = \frac{1}{T_{symbol}} \sum_{n=1}^{N} b_i(n) \tag{28}$$

T_{symbol} in Equation (28) is constant, so maximizing Equation (20) is equivalent to maximizing the aggregate bit number over all sub-channels and over all users.

$$b_{total} = \sum_{i=1}^{M} \sum_{n=1}^{N} b_i(n) \tag{29}$$

Let us start from the first maximization objective in Equation (29). Constraints (21) and (23) can be used as checking criteria during each step of the optimization. Substitute Equation (27) into Equation (29), we get the first objective function as

$$F_1\left[\mathbf{P}(n)\right] = \sum_{i=1}^{M} \sum_{n=1}^{N} \log_2 \left(1 + \frac{P_i(n) \left| H_{ii}(n) \right|^2}{\Gamma \left(\sigma_i^2(n) + \sum_{\substack{j=1 \\ j \neq i}}^{M} P_j(n) \left| H_{ji}(n) \right|^2 \right)} \right) \tag{30}$$

The constraint is Equation (22). Use the Lagrange multipliers method to solve this problem. Construct the Lagrangian function as

$$L = F_1\left[\mathbf{P}(n)\right] + \sum_{i=1}^{M} \lambda_i \left(\sum_{n=1}^{N} P_i(n) - P_{budget}(i) \right) \tag{31}$$

Make derivatives of L to $P_i(n)$, and let it equal to zero to find the optimal point of L.

$$\frac{\partial L}{\partial P_i(n)} = \frac{1}{\ln 2}\cdot\frac{\left|H_{ii}(n)\right|^2 \Big/ \Gamma\left(\sigma_i^2 + \sum_{\substack{j=1\\ j\neq i}}^{M} P_j(n)\left|H_{ji}(n)\right|^2\right)}{1+P_i(n)\left|H_{ii}(n)\right|^2 \Big/ \Gamma\left(\sigma_i^2 + \sum_{\substack{j=1\\ j\neq i}}^{M} P_j(n)\left|H_{ji}(n)\right|^2\right)}$$
$$+\frac{1}{\ln 2}\sum_{\substack{j=1\\ j\neq i}}^{M}\frac{-P_j(n)\left|H_{ii}(n)\right|^2\left|H_{ji}(n)\right|^2 \Big/ \Gamma\left(\sigma_j^2 + \sum_{\substack{m=1\\ m\neq j}}^{M} P_m(n)\left|H_{jm}(n)\right|^2\right)^2}{1+P_j(n)\left|H_{jj}(n)\right|^2 \Big/ \Gamma\left(\sigma_j^2 + \sum_{\substack{m=1\\ m\neq j}}^{M} P_m(n)\left|H_{jm}(n)\right|^2\right)} \tag{32}$$
$$+\lambda_i$$
$$=0$$

The first term is the contribution from user *i* itself. The second term is a function of $P_j(n)$ of all other (M-1) users, and shows the effect of crosstalk. Since this term has a high order in denominator and because the crosstalk is weak compare to the main signal, we can ignore the second term to make the equation tractable and get the approximated expression of $P_i(n)$.

$$\frac{1}{\ln 2}\cdot\frac{\left|H_{ii}(n)\right|^2 \Big/ \Gamma\left(\sigma_i^2 + \sum_{\substack{j=1\\ j\neq i}}^{M} P_j(n)\left|H_{ji}(n)\right|^2\right)}{1+P_i(n)\left|H_{ii}(n)\right|^2 \Big/ \Gamma\left(\sigma_i^2 + \sum_{\substack{j=1\\ j\neq i}}^{M} P_j(n)\left|H_{ji}(n)\right|^2\right)} + \lambda_i = 0 \tag{33}$$

Making a simple arrangement, $P_i(n)$ is expressed as,

$$P_i(n) = \frac{-1}{\lambda_i \ln 2} - \frac{\Gamma}{\left|H_{ii}(n)\right|^2}\left(\sigma_i^2 + \sum_{\substack{j=1\\ j\neq i}}^{M} P_j(n)\left|H_{ji}(n)\right|^2\right) \tag{34}$$

We denote

$$C_{i,\lambda} = \frac{-1}{\lambda_i \ln 2} \tag{35}$$

and

$$T_i(n) = \frac{\Gamma}{\left|H_{ii}(n)\right|^2}\left(\sigma_i^2 + \sum_{\substack{j=1\\ j\neq i}}^{M} P_j(n)\left|H_{ji}(n)\right|^2\right) \tag{36}$$

then $P_i(n)$ is expressed as

$$P_i(n) = C_{i,\lambda} - T_i(n) \tag{37}$$

Equation (37) has the form of "water filling" solution with the water level as $C_{i,\lambda}$, and the terrain that holds the poured power as $T_i(n)$. The constant $C_{i,\lambda}$ can be solved from the constraint (22). Substitute Equation (37) into Equation (22),

$$\sum_{n=1}^{N}\left[C_{i,\lambda}-\frac{\Gamma}{\left|H_{ii}(n)\right|^2}\left(\sigma_i^2+\sum_{\substack{j=1\\ j\neq i}}^{M}P_j(n)\left|H_{ji}(n)\right|^2\right)\right]=P_{budget}(i) \tag{38}$$

$$NC_{i,\lambda}-\sum_{n=1}^{N}\frac{\Gamma}{\left|H_{ii}(n)\right|^2}\left(\sigma_i^2+\sum_{\substack{j=1\\ j\neq i}}^{M}P_j(n)\left|H_{ji}(n)\right|^2\right)=P_{budget}(i) \tag{39}$$

Therefore, we get

$$C_{i,\lambda}=\frac{1}{N}\left[P_{budget}(i)+\sum_{k=1}^{N}\frac{\Gamma}{\left|H_{ii}(k)\right|^2}\left(\sigma_i^2+\sum_{\substack{j=1\\ j\neq i}}^{M}P_j(k)\left|H_{ji}(k)\right|^2\right)\right] \tag{40}$$

According to (Marler & Arora, 2003), the solution of a multi-objective optimization could be obtained by the "Lexicographic Method". The method solves the objective functions in the order of importance, and constructs new constraints for next objective function using the solution for previous objective. In this multi-user bit-loading problem, we first get the optimal solution $\mathbf{b}^*(n)$ of Equation (29) as described above, and then process the second objective by minimizing Equation (25) and subject to

$$b_{total}\left[\mathbf{b}(n)\right]\leq b_{total}\left[\mathbf{b}^*(n)\right] \tag{41}$$

It is obvious that this optimization is even more complex than the first objective optimization, because both objective function and constraint function include variables in high order terms and denominators. A practical implementation of this algorithm is required.

4.5 Greedy algorithm for multi-user nit loading

A practical method to obtain the optimal solution of Equation (29) is the extended greedy algorithm. The foundation of greedy algorithm is to calculate the power cost to transmit additional bits for each subchannel (*i*, *n*). First, we rearrange Equation (29) to remove the logarithm,

$$\Gamma\left(2^{b_i(k)}-1\right)=\frac{P_i(k)\left|H_{ii}(k)\right|^2}{\sigma_i^2+\sum_{\substack{j=1\\ j\neq i}}^{M}P_j(k)\left|H_{ji}(k)\right|^2} \tag{42}$$

Let,

$$f\left(b_i(n)\right)=\Gamma\left(2^{b_i(n)}-1\right) \tag{43}$$

Then we get,

$$P_i(n) - f\left(b_i(n)\right)\sum_{\substack{j=1\\ j\neq i}}^{M} P_j(n)\frac{\left|H_{ji}(n)\right|^2}{\left|H_{ii}(n)\right|^2} = f\left(b_i(n)\right)\frac{\sigma_i^2}{\left|H_{ii}(n)\right|^2} \tag{44}$$

The above equation can be expressed in matrix form as

$$AX = B \tag{45}$$

where

$$A = \begin{bmatrix} 1 & \cdots & \dfrac{-f\left(b_1(k)\right)\left|H_{i1}(k)\right|^2}{\left|H_{11}(k)\right|^2} & \cdots & \dfrac{-f\left(b_1(k)\right)\left|H_{M1}(k)\right|^2}{\left|H_{11}(k)\right|^2} \\ \vdots & 1 & \cdots & \cdots & \vdots \\ \dfrac{-f\left(b_i(k)\right)\left|H_{1i}(k)\right|^2}{\left|H_{ii}(k)\right|^2} & \cdots & 1 & \cdots & \dfrac{-f\left(b_i(k)\right)\left|H_{Mi}(k)\right|^2}{\left|H_{ii}(k)\right|^2} \\ \vdots & \cdots & \cdots & 1 & \vdots \\ \dfrac{-f\left(b_M(k)\right)\left|H_{1M}(k)\right|^2}{\left|H_{MM}(k)\right|^2} & \cdots & \dfrac{-f\left(b_M(k)\right)\left|H_{iM}(k)\right|^2}{\left|H_{MM}(k)\right|^2} & \cdots & 1 \end{bmatrix} \tag{46}$$

$$X = \begin{bmatrix} P_1(k) & \cdots & P_i(k) & \cdots & P_M(k) \end{bmatrix}^T \tag{47}$$

$$B = \begin{bmatrix} \dfrac{f\left(b_1(k)\right)\sigma_1^2}{\left|H_{11}(k)\right|^2} & \cdots & \dfrac{f\left(b_i(k)\right)\sigma_i^2}{\left|H_{ii}(k)\right|^2} & \cdots & \dfrac{f\left(b_M(k)\right)\sigma_M^2}{\left|H_{MM}(k)\right|^2} \end{bmatrix}^T \tag{48}$$

In Solving the above linear equation system, we obtain the power vector $\mathbf{P}(n)$ required to transmit bit allocation scheme $\mathbf{b}(n)$ on subchannel n for all M users. One thing worthy of notice is that the solution of $\mathbf{P}(n)$ vector may contain negative elements. This indicates that the corresponding users cannot afford to transmit the number of bits assigned to them on subchannel n.

In DMT, different subchannels are well separated. The crosstalk coupling between different users only appears in the subchannels with same frequency. So we assume there is no interference between subchannels. Equation (45) with different $n = 1, \ldots, N$ could be solved independently. Let $b_i'(k) = b_i(k) + \Delta b$ for i = 1 to M and n = 1 to N, do a calculation of Equation (45) again to obtain the new power allocation $\mathbf{P}'(n)$. Because of the crosstalk coupling, one element change in the coefficient matrix A in Equation (45) causes power of all the users on the same subchannel to change. The cost of adding Δb bits on subchannel (i, n) is the summation of power increment on all users on subchannel n.

$$\text{cost}(i,k) = \sum_{i=1}^{M}\left(P_i'(k) - P_i(k)\right) \tag{49}$$

This calculation needs to be done on every subchannel and every user. The final cost matrix is M rows by N columns, with each element at position (*i, n*) represents the cost of transmitting addition bits on subchannel *n* of user *i*. The position of the minimum cost determines the subchannel and user where additional bits will be transmitted.

Power budget need to be checked during the cost updating. Additional Δb bits added to subchannel *n* of user *i* may cause the user *i* to get more power on subchannel *n*, and at the same time, it causes all other users to increase their power on subchannel *n* in order to maintain their SNR to transmit already-assigned number of bits. Either the user *i* or other users may have the possibility to exceed their power budget. If this happens, it means that the adding of Δb bits to subchannel *n* of user *i* is not feasible.

4.6 Efficiency improvement to add bits to multiple users in parallel

As discussed in the Section 4, the large number of iterations for multi-user DMT makes the greedy bit-loading algorithm hard to deploy. We mitigate this problem by processing multiple users in parallel, so that the number of iterations could be reduced dramatically. As we know, crosstalk only interferes with users on the subchannels that are in same frequency. Different-frequency subchannels are independent with each other. In other words, we can rewrite Equation (45) by indicating the subchannel index *n* explicitly as

$$A(n)X(n) = B(n) \tag{50}$$

If, for example, two subchannels with different indices n_1 and n_2 get additional bits, no matter whether the subchannels are for the same user or for different users, two linear Equations of (50) have to be calculated. This means that adding bits to subchannels in the dimension of subchannel index requires the same number of iterations as the number of subchannels processed. However, if we add bits to subchannels in the dimension of user index with subchannel index fixed, it is possible to reduce the number of calculations of Equation (50). Let us say, for instance, within a specific subchannel *n*, we assign additional bits to two users i_1 and i_2, the resulting power scheme at subchannel *n* $\mathbf{P}'(n)$ could be calculated by solving a single equation of Equation (50).

The subchannel with minimum cost is identified by both subchannel index *n* and user index *i*. So, in the proposed algorithm, instead of adding bits to only one subchannel (*i*, *n*), we look for all users on subchannel *n*, which has cost very close to the minimum cost ($\text{cost}_{\min}$). The additional bits are added to subchannel *n* of all these users. Fig. (5) visualizes the idea.

The term "close" to the minimum cost is defined by a cost elastic coefficient δ_{cost}. On subchannel *n* where $\text{cost}_{\min}$ appears, if any user *i* satisfies the condition

$$\frac{\text{cost}(i,n) - \text{cost}_{\min}}{\text{cost}_{\min}} < \delta_{\text{cost}} \tag{51}$$

we add additional bits to it. The value of δ_{cost} shows the percentage degree of how much the cost on a given subchannel is greater than the minimum cost. It could have any value greater than zero, depending on the accuracy we want in the algorithm. The effect of this coefficient will be analyzed in next chapter. The simulation result shows that importing this cost elastic coefficient has nearly no negative impact on the final bit-loading scheme, but the number of iterations reduced greatly.

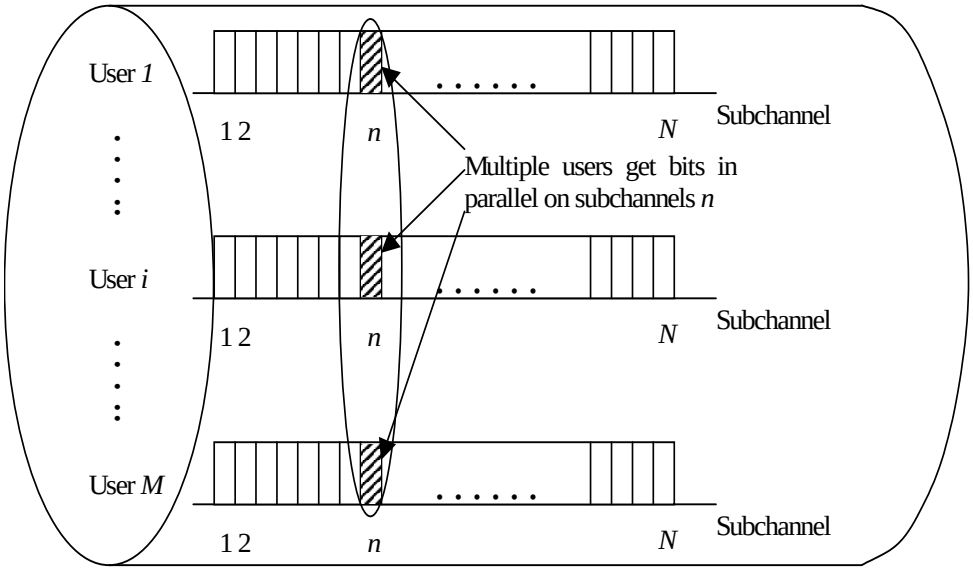


Fig. 5. Multiple Users Get Bits in Parallel

With the updated bit-loading scheme on subchannel n, we solve a single linear equation set in Equation (50) to obtain the new power allocation scheme. It reflects the changes of bit number for several users. To prepare for the next iteration, the nth column in cost matrix needs to be updated. As usual, we assume additional bits are added to each user at subchannel n, and calculate the cost according to Equation (49). The comparison of traditional algorithm and improved efficient algorithm is listed in Table 1.

Traditional Algorithm	Efficient Algorithm
Do Update cost matrix Find $cost_{min}$ at subchannel (i, n) Add bit to subchannel (i, n) Update nth column of A, B, X in (50) **While** all subchannels are filled or power budget are reached	**Do** Update cost matrix Find $cost_{min}$ at subchannel (i_0, n) Find all users (i_1,…i_k) that have cost close to $cost_{min}$ at subchannel n Add bit to users i_0, i_1,…i_k at subchannel n Update nth column of A, B, X in (50) **While** all subchannels are filled or power budget are reached

Table 1. Algorithm Comparison of Traditional and Efficient Multi-User Bit Loading

4.7 Fairness control

The objective of minimizing the data rate variance among M users in Equation (25) is equivalent to minimize the variance of total number of bits of M users. We obtain this objective by importing a fairness elastic coefficient δ_{fair} in the bit loading process. In each greedy bit loading iteration, we check the total bits number sum_bits(i) of the user that just

got the additional bits assigned to it, and keep the number in lock with the averaged total bit numbers of all the other users. The range of lock is adjustable by using the fairness control coefficient δ_{fair}.

$$\text{sum\_bits}(i) \le \delta_{fair} \frac{1}{M-1} \sum_{\substack{j=1, j\neq i \\ j\notin \text{Frozen}}}^{M} \text{sum\_bits}(j) \tag{52}$$

When the sum_bits(i) exceeds the average sum_bits of all other users, the user i is to be in "frozen" state temporarily. Users in frozen set will not participate in the further bits assignment until they are unfrozen. The value of fairness elastic coefficient δ_{fair} depends on the degree of fairness we want. δ_{fair} = 1 ensures the final sum_bits of all users have minimum variance. If we loose the elastic coefficient, the variance may increase, and the better-condition users can get higher data rate.

4.8 Algorithm Implementation

In this work, we designed our newly proposed algorithm as a flexible solution, so that it supports both parallel bit loading and fairness adjustment, or other kinds of combinations. For example, we could run the algorithm with only parallel bit loading, or only fairness adjustment, or neither of them, which becomes the traditional multi-user greedy bit loading. This flexibility makes it easier for us to run the algorithm in different modes, and observe the effects of our improvements.

Besides the switch variables that control the mode of algorithm, two flag variables (flag_user and flag_channel) are used to assist in the algorithm execution. The variable flag_user is a vector that indicates the status of M users. The user i is available in bit allocation process if flag_user(i) = 0; If the user i is set to "frozen" status because of the fairness adjustment, flag_user(i) has the value of 1. The user will relinquish its chance to get additional bits on all its subchannels, until the flag_user(i) is set to 0 again. Since multi-user bit loading involves two dimensions – users and subchannels, we defined a matrix flag_channel (M rows and N columns) to indicate whether a subchannel (i, n) is available to allocate more bits. flag_channel(i, n) = 0 indicates subchannel n of user i is available; flag_channel(i, n) = 1 indicates this subchannel is "full". There are two possible reasons that cause a subchannel to be full:

1. The number of bits assigned to the subchannel reaches the cap b_{max};
2. Adding additional bits causes the power consumed on user i to exceed the power budget.

The algorithm is described in the following list.

Initialization

1. $b_i(n) = O_{M, N}$; ($O_{M, N}$ is the M rows N columns zero matrix)
2. $P_i(n) = O_{M, N}$;
3. flag_user = $O_{1, M}$;
4. flag_channel = $O_{M, N}$;
5. sum_bits = $O_{1, M}$;
6. cost(i, n) = $O_{M, N}$;
7. Calculate the initial matrices A and B using (46) and (48);

8. Let additional bits to be assigned to each subchannel temporarily and calculate the new power vector $\mathbf{P}'(n)$ using Equation (45), then calculate the initial cost matrix using Equation (49);

Iteration

1. Find the subchannel (i, n) that has the minimum value in cost matrix.
2. If the minimum cost is greater than or equal to a predefined BIGNUMBER, stop the iteration; algorithm finishes because no subchannel can get additional bits without breaking the power budget.
3. If parallel bit loading is allowed, find all users that have costs close to the minimum cost to transmit addition bits on subchannel n.
4. Add Δb bits to subchannel n of all the users selected in step 3.
5. Update the matrices A and B for subchannel n, and calculate the new power vector $\mathbf{P}'(n)$.
6. Check if there are any elements in $\mathbf{P}'(n)$ that have negative value. If so, consider two cases: 1. multiple users got additional bits on subchannel n, then mark this subchannel to not allow parallel bit loading and return to step 1; 2. only one user got additional bits, set flag_channel(i, n) = "full", and set the $\text{cost}_i(n)$ to be BIGNUMBER, then return to step 1.
7. Check if the total power on any user exceeds the budget. If so, do the same processing as in step 6.
8. If fairness adjustment is allowed, check the fairness using Equation (52). If the bit sum of user i exceeds the elastic range, set flag_user = "frozen", and add a BIGNUMBER to the costs of all subchannels on this user.
9. For those users that already belong to the frozen set, check if their bit sum drop back into the elastic range. If so, unfreeze them, and subtract BIGNUMBER from subchannel costs to restore the original values.
10. Update part of cost matrix, which correspond to subchannel n of all users.
11. Go to step 1.

5. Simulation, results and analysis

We implemented the proposed multi-user bit-loading algorithm for DMT in MATLAB. In Section 5, we analyzed the performance of our algorithm and made comparisons for different variances. The simulation was applied on ADSL and straight loop with variant loop lengths. The models of loop and far-end crosstalk (FEXT) came from (ANSI Std. T1.417, 2001). We assume there are 50 subscriber loops in a cable, so the number of FEXT sources is 49. The loop lengths are generated as uniformly distributed random numbers within the range of 2 – 16 kft. They are sorted in ascendant order so that the performance of these loops would be displayed in descending order. Fig 6 shows the sorted loop lengths that were used in our simulation.

Only downstream signals were considered for simplicity in this simulation. The process of upstream signal is similar. As shown in Fig. 7, each receiver at the far end of the loop gets signal from both its own transmitter and FEXT noise from all other transmitters in the same cable. We assume all the 50 loops in the cable transmit ADSL signals; and no interference from other services exists.

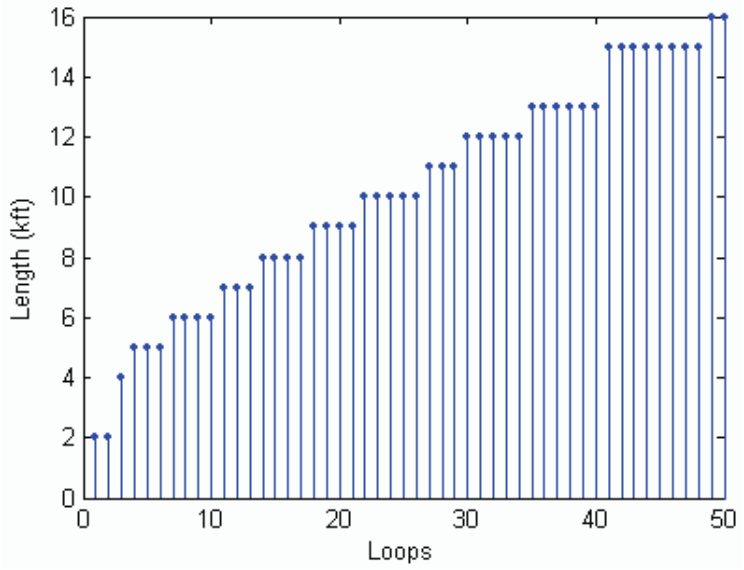


Fig. 6. Loop Lengths for the Simulation

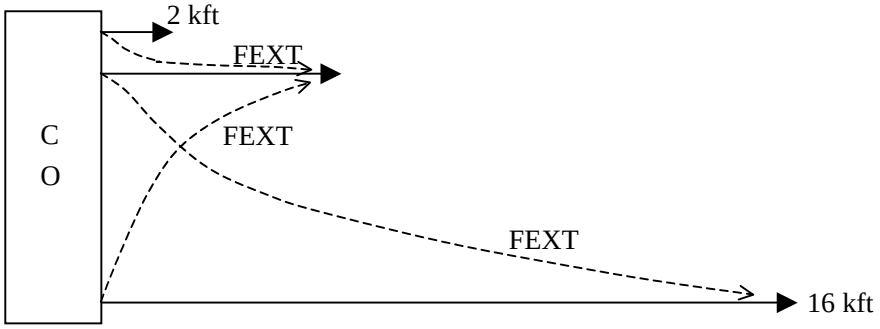


Fig. 7. FEXT Among Loops with Differnt Lengths

We assume there is no code gain and no noise margin, so the SNR gap Γ is chosen as 9.8 dB according to Equation (5). In the downstream of ADSL DMT, the subchannels start from 33 and end at 255 (ANSI Std. T1.417, 2001). Therefore, we are considering 255-33+1=223 subchannels in this simulation. The corresponding frequency range is 142 kHz – 1100 kHz with subchannel spacing as 4312.5 Hz. The power budget for each user is 20.4 dBm (ANSI Std. T1.417, 2001). For simplicity, we chose the incremental unit for bit addition as $\Delta b = 1$ bit.

5.1 Simulation mode

We have two improvements in the proposed algorithm: parallel bit allocation and fairness control. So we run the algorithm in all possible combinations. Table 2 shows the four operational modes of the algorithm. In traditional mode, neither of the two new features are applied, this is equivalent to the traditional multi-user DMT bit loading as in (Lee et al., 2002). We used this mode as reference to compare with our proposed features.

	Parallel Bit Allocation	Fairness
Traditional Mode	N	N
Parallel Only Mode	Y	N
Fairness Only Mode	N	Y
Both Mode	Y	Y

Table 2. Operational Mode of the Algorithm

5.2 Simulation result analysis

In the proposed algorithm, we introduced two elastic coefficients (δ_{cost} and δ_{fair}) for parallel bit loading and fairness control respectively. However, how the coefficients affect the performance of algorithm is not straightforward. This section shows the analysis of the simulation results to demonstrate that by using cost elastic coefficient, the computational load decreased significantly; and by using fairness elastic coefficient, we have a way to control the data rate fairness among users in the same cable.

Traditional operational mode is used as comparison reference. In the following analysis, the value of –1 for δ_{cost} or δ_{fair} indicates that the algorithm does not utilize the proposed parallel bit loading or fairness control. The first data points in the following figures that have control value of –1 are used as reference points.

5.3 Effectiveness of parallel bit loading

The purpose of parallel bit loading is to reduce the computational load in the traditional multi-user DMT bit loading. In this section, we ran the algorithm with no fairness control so that it has comparability with traditional operational mode in terms of parallel bit loading. From Fig. 8 we see that when there are 50 users in a cable, the number of iterations required

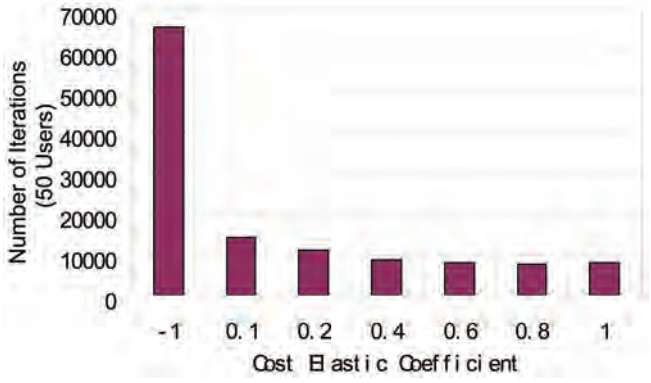


Fig. 8. Number of Iterations vs. Cost Elastic Coefficient (50 Users)

to allocate all bits to users and subchannels is 66016, if no parallel bit-loading technique is used (δ_{cost} = -1). If δ_{cost} is set to 0.1, the iterations number dropped immediately to only 22% of the previous value. The number even reduced to 12% if the δ_{cost} is set to be greater than 0.6. This shows that parallel bit loading has a great improvement on the computational load for multi-user DMT bit loading. Fig. 9. shows the similar effect with the comparison for both 50 users and 15 users. We see that the more users in a cable, the more significant improvement could be achieved for number of iterations. Furthermore, we noticed that as

long as parallel bit loading is used, the iteration numbers is not very sensitive to the value of δ_{cost}, especially when $\delta_{cost} \geq 0.4$. Set δ_{cost} to 1 is a good choice.

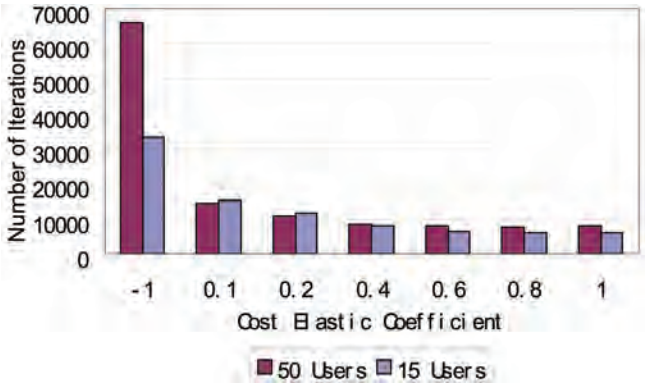


Fig. 9. Number of Iterations vs. Cost Elastic Coefficient (50 Users and 15 Users)

Reducing the iterations number is not the only goal of the algorithm. To make the algorithm meaningful, we must guarantee that the final bit-loading result has no or only little loss comparing with the traditional algorithm. We run the algorithm to generate bit-loading schemes for 50 users. The number of bits on each user over all subchannels determines the data rate of the user. The average value (mean) of the bits numbers for 50 users specifies the performance of bit-loading algorithm.

$$\bar{b} = \frac{1}{50}\sum_{i=1}^{50}\sum_{n=1}^{N} b_i(n) \tag{53}$$

Table 3 and Fig. 10. show the mean bit numbers when different δ_{cost} values were used. It is clear from the figure that the mean bits numbers are constant for all δ_{cost} values that are less than or equal to 1. As cost elastic coefficient increases to greater than 1, the mean bit number decreases. Therefore, selecting δ_{cost} to be 1 does not result in loss of any accuracy of the algorithm.

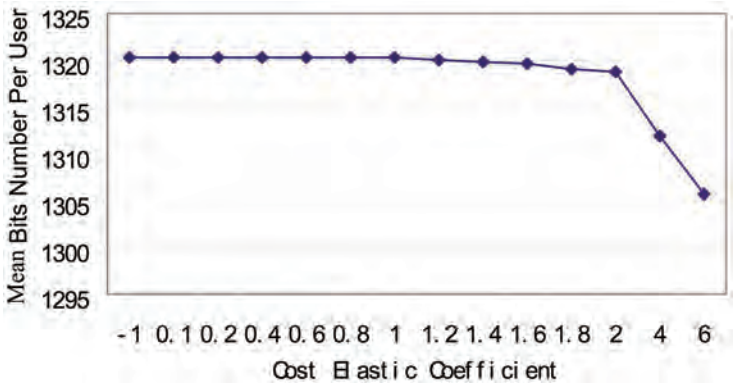


Fig. 10. Mean Bits Number Per User vs. Cost Elastic Coefficient

Fig. 11 demonstrates that many users get bits assigned to them simultaneously in a single iteration (marked by *). In this case, bits were assigned to 32 users in the iteration 3000. In other words, one iteration in the new efficient algorithm is equivalent to 32 iterations in traditional algorithm. That is a remarkable improvement.

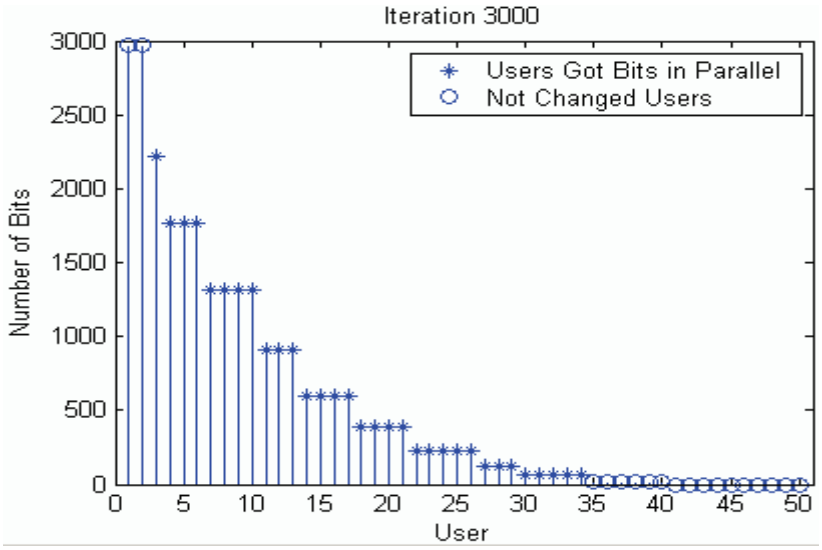


Fig. 11. Multiple Users Get Bits in a Single Iteration

5.4 Effectiveness of fairness control

By introducing the fairness control coefficient δ_{fair}, we have the ability to constraint the variance of bit numbers over all users. As stated before, the loop lengths of 50 users have been sorted in ascendant order. That is, user 1 has the shortest loop length, and user 50 has the longest loop length. Therefore, if no fairness control is applied, the total number of bits assigned to each user is in the descendant order. The dashed curve in Fig. 13 is in this case. When we set the δ_{fair} to be 1, the bits number of each user is limited to have minimum variance. Actually, in Fig. 12, it is a straight line (the solid curve). Two more curves are showed in Fig. 12, which represent the case for δ_{fair} equals to 1.2 and 1.4 respectively. Fig. 13 shows the mean and standard deviation of bits number per user in the same figure when different δ_{fair} were used. We see that these two curves have similar shapes. That is, when the standard deviation reduced, the mean also reduced. The benefit of fairness is compensated by the loss of average data rate. Fig. 14 shows how the fairness control coefficient affects the number of iterations. When δ_{fair} equals to 1, the number of iterations is minimum, but at the same time, the number of bits assigned to each user is also minimum.

Fig. 12 also shows that the bits numbers of short-loop users have very limited improvement when fairness control is applied. So the bits numbers for short-loop users have to be dropped greatly to obtain the small variance. This indicates that although the application of fairness control makes the standard deviation of bits number (data rate) small, we pay the cost to sacrifice the performance for short-loop users. Therefore, the fairness control may not be strongly desired.

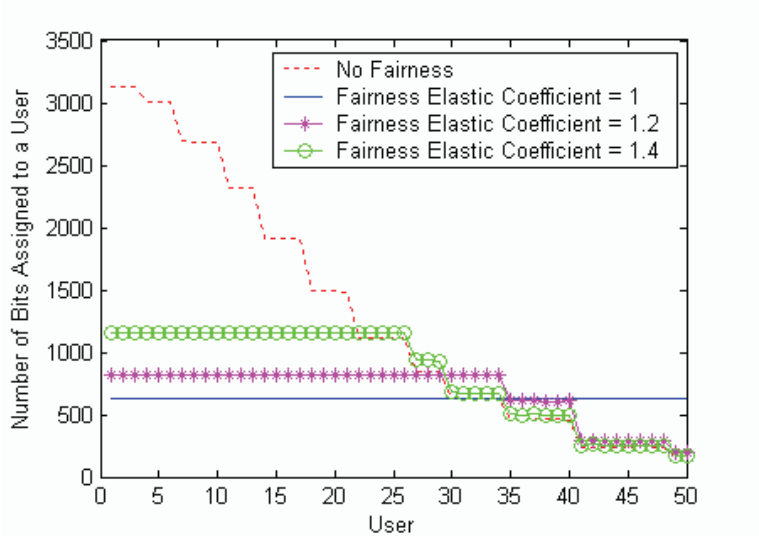


Fig. 12. Bits Number of Users

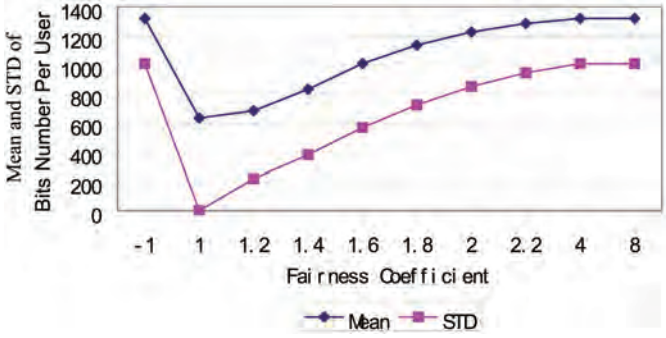


Fig. 13. Mean and Standard Deviation of Bits Number vs. Fairness Coefficient

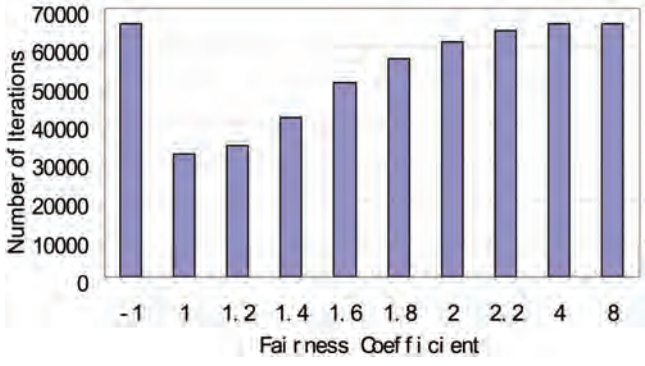


Fig. 14 Number of Iterations vs. Fairness Coefficient

5.5 Combination of parallel bit loading and fairness control

When the parallel bit loading and fairness control are applied at the same time, the combined effectiveness of them needs to be identified. This section analyzed the simulation results and illustrated them in 3-D plots. The algorithm was run in full combination of δ_{cost} and δ_{fair}. The major performance measures are recorded, such as number of iterations, mean and standard deviation of bits number per user. Tables 3, and 4 are some of the simulation results.

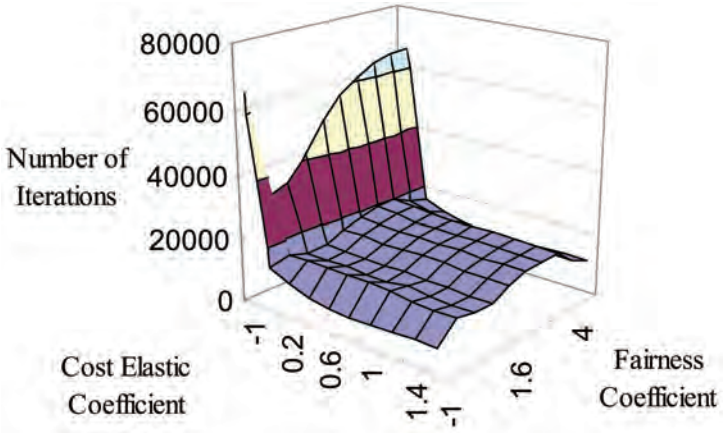


Fig. 15. Number of Iterations vs. Cost Elastic Coefficient and Fairness Coefficient

Fig. 15 indicates that the number of iterations has a strong correlation with cost elastic coefficient. As soon as the parallel bit loading is applied (δ_{cost} has any value other than -1), the number of iterations dropped significantly. Compared with the significance of parallel bit loading, the fairness coefficient has relatively little contribution to reduce the iteration number.

δ_{fair} \ δ_{cost}	-1	0.1	0.2	0.4	0.6	0.8	1
-1	1320.3	1320.3	1320.3	1320.3	1320.3	1320.3	1320.3
1	632.96	632.96	632.96	632.96	632.96	632.96	629.98
1.2	683.4	703.48	703.46	705.1	701.92	696.02	719.48
1.4	831.48	831.52	831.46	830.78	830.1	829.96	829.38
1.6	1009.22	1009.22	1009.24	1008.76	1009.2	1008.82	1009.36
1.8	1136.78	1137.18	1137.16	1137.7	1138.2	1138.26	1138.38
2	1226.76	1227.2	1226.98	1228.22	1228.9	1229.32	1229.4
2.2	1285.82	1286	1286.02	1287.18	1287.8	1287.74	1287.8
4	1320.3	1320.3	1320.3	1320.3	1320.3	1320.2	1320.2
8	1320.3	1320.3	1320.3	1320.3	1320.3	1320.3	1320.3

Table 3. Mean Bits Number Per User Under Different Coefficients Values

δ_{cost} / δ_{fair}	-1	0.1	0.2	0.4	0.6	0.8	1
-1	1008.1713	1008.1713	1008.1713	1008.1713	1008.1713	1008.1713	1008.1713
1	0.19795	0.19795	0.19795	0.19795	0.19795	0.19795	0.14142
1.2	218.5638	223.4621	223.5784	224.1786	223.0507	221.6587	229.2879
1.4	380.3019	380.3123	380.3548	379.924	379.6629	379.6214	379.1612
1.6	567.4984	567.5644	567.6492	567.3555	568.2036	567.9887	568.7456
1.8	723.7036	724.4047	724.506	725.7295	726.8624	727.1101	727.4138
2	850.2531	850.9939	850.6976	852.8352	854.0387	854.7548	854.9612
2.2	944.991	945.3436	945.3933	947.5979	948.9535	948.8161	948.8768
4	1008.1713	1008.1713	1008.1713	1008.1713	1008.1713	1007.935	1007.9803
8	1008.1713	1008.1713	1008.1713	1008.1713	1008.1713	1008.1713	1008.1713

Table 4. Standard Deviation of Bits Number Per User Under Different Coefficients Values

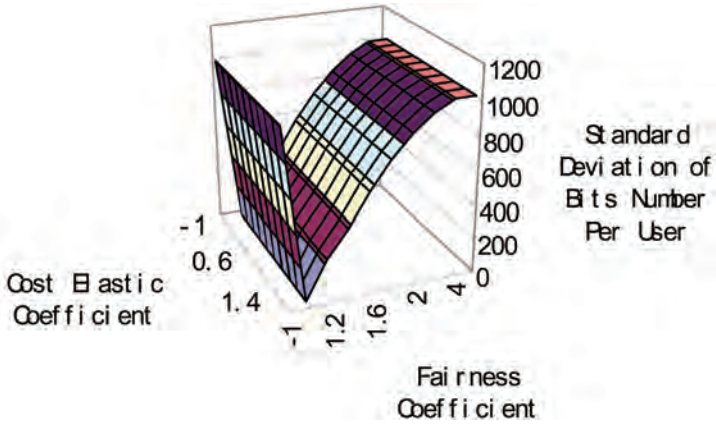


Fig. 16. Standard Deviation of Bits Number vs. Cost Elastic Coefficient and Fairness Coefficient

In the plot of Table 3 which is not shown here, it shows how the mean bits number per user is affected by cost elastic coefficient and fairness coefficient. The surface is flat along the cost elastic coefficient direction, which means the selection of δ_{cost} has almost no influence on the mean bits number. The only contribution to affect of the mean bits number is the value of δ_{fair}. Fig.16 shows how cost elastic coefficient and fairness coefficient affect the standard deviation of bits number per user. The isolated effects of parallel bit loading and fairness control gave us the flexibility to apply them separately or in combination according to the requirement.

6. Summary of conclusions

Traditional bit-loading algorithms for DMT system were designed for a single user. As more and more users turn to DSL services, crosstalk inside the cable makes the traditional single-user bit-loading algorithms unable to reach the optimal solution. Multi-user bit-loading algorithms were designed to obtain the global optimization among all users in a cable. The problem of multi-user bit loading was computational complexity. Because the number of subchannels for all users is very large, the calculation of power allocation needs to done many times.

This research work studied several bit-loading algorithms for both single user and multiple user cases. Then an improved parallel bit-loading algorithm for multi-user DMT was proposed. The new algorithm was based on multi-user greedy bit loading. In a single iteration, the bits were allocated to multiple users on the subchannels with same frequency. Thus, the number of iterations to allocate bits to all users decreased significantly. The adjustable cost elastic coefficient defined the range of power cost. In the bit-loading iteration, all subchannels for different users that have the power cost within the range have the chance to get additional bits assigned to them. Furthermore, this research work studied the possibility to improve the fairness among users in the same cable. We introduced a fairness coefficient during bit-loading iterations so that the variance of total number of bits for all users could be controlled.

The analysis of simulation results showed that the effectiveness of applying parallel bit loading is significant. If the cost elastic coefficient has the value of 1, it means that all subchannels who has power cost less than twice of the minimum cost could get additional bits assigned to them. For 50 users case and δ_{cost}=1, the number of iterations to load all bits to all users reduced to only 12% of the number if no parallel bit loading is used. Another good characteristic of parallel bit loading is that it has very small effect on the final loading result. In the same 50-user case, when δ_{cost} is less than 1, the loading result is exactly the same as no parallel bit loading. This means that we reduced the computational complexity without losing any accuracy of the result. The simulations also showed that the fairness in the algorithm control could limit the variance of total bit numbers among all users. The cost of fairness is that the better-condition loops (for example, shorter loop length) have to reduce their data rate, because the worse-condition loops have little improvement in their data rate.

7. Recommendations for future work

The parallel multi-user bit-load algorithm proposed in this research work reduced the number of iterations to allocated bits and power to all users. However, the computational load was still quite big even under the improved algorithm. Further improvements are required. For example, the idea of bit removal algorithm in (Sonalker & Shively, 1998) could be applied in the multi-user bit loading. In long-length loops, many high-frequency subchannels have no chance to get bits assigned to them. The better way is to turn off these subchannels before the algorithm starts. The computational load could be reduced even more when those "bad" subchannels are eliminated from the bit-loading process.

It is implied in the multi-user bit loading that a central process unit must exist in the network. The central unit possesses the channel information of all users, such as channel transfer function and crosstalk transfer function. The bit-loading algorithm runs on the central unit and the loading scheme is then distributed to all transmitters in the same cable. Therefore, the channel and crosstalk estimation need to be added into the multi-user bit-loading system (Cioffi et al., 2001) and (Zeng et al., 2001).

8. Reference

Akujuobi, C. M. & Shen, J. (2006). A New Parallel Greedy Bit-Loading Algorithm With Fairness for Multi-Users in a DMT System, IEEE Transactions on Communications, Vol. 54, No. 8, August 2006

ANSI Std. T1.417 (2001). Spectrum Management for Loop Transmission Systems, *American National Standard Institute*

Campello, J. (1998). Optimal Discrete Bit Loading for Multicarrier Modulation Systems, *Proceedings of IEEE International Symposium on Information Theory*, pp. 193, 1998

Campello, J. (1999). Practical Bit Loading for DMT, *IEEE International Conference on Communications*, Vol. 2, pp 801-805, June 1999

Cioffi, J. M.; Aldana, C.; Ekbal, A. ; Fang, J.; Ginis, G.; Lee, J.; Salvekar, A.; Yu, W.; and Zeng, C. (2001). Channel Identification with Dynamic Spectrum Management, *T1E1.4 Spectrum Management II Project*, 2001-147, May 2001

Cioffi, J. M. (2001). Proposed Scope and Mission for Dynamic Spectrum Management, *T1E1.4 Spectrum Management Project*, 2001-188R4, November 2001

Cioffi, J. M.; Yu, W.; Chung, S. T.; and Ginis, G. (2001). Iterative Water-filling for DSM, *T1E1.4 Spectrum Management Project*, 2001-200R5, November 2001

Chow, P. S.; Cioffi, J. M.; Bingham, J. A. C. (1995). A Practical Discrete Multitone Transceiver Loading Algorithm for Data Transmission over Spectrally Shaped Channels, *IEEE Transactions on Communications*, Vol. 43, no. 234, pp. 773-775, February/March/April 1995

Chow, P. S.; Cioffi, J. M. (1995). Method and Apparatus for Adaptive, Variable Bandwidth High-Speed Data Transmission of a Multicarrier Signal Over Digital Subscriber Lines, *U.S. Patent* 5,479,447 (December 1995)

Cover, T. M. & Thomas, J. A. (1991). Elements of Information Theory, *John Wiley & Sons*, 1991

Gallager, R. G. (1968). Information Theory and Reliable Communication, *John Wiley*, New York, 1968

Hughes-Hartogs, D. (1987-1989). Ensemble Modem Structure for Imperfect Transmission Media, *U.S. Patent* 4,679,227 (July 1987), 4,731,816 (March 1988), and 4,883,706 (May 1989)

Lee, J.; Sonalkar, R. V. and Cioffi, J. M. (2002). Multi-user discrete bit-loading for DMT-based DSL Systems, *IEEE Global Telecommunications Conference*, Vol. 2, pp. 1259-1263, November 2002

Leke, A. and Cioffi, J. M. (1997). A Maximum Rate Loading Algorithm for Discrete Multitone Modulation Systems, *IEEE Global Telecommunications Conference*, Vol. 3, pp. 1514-1518, November 1997

Marler, R. T. and Arora, J. S. (2003). Review of Multi-Objective Optimization Concepts and Algorithms for Engineering, University of Iowa Optimal Design Laboratory, *Technical Report Number ODL-01.03*, 2003

Shannon, C. E. (1948). A Mathematical Theory of Communication, *The Bell System Technical Journal,* Vol. 27, pp.379-423, 623-656, 1948

Sonalker, R. V. & Shively, R. R. (1998). An Efficient Bit-Loading Algorithm for DM Applications, *IEEE Global Telecommunications Conference*, Vol. 5, pp 8-12, November 1998

Starr, T.; Cioffi, J. M.; Silverman, P. J. (1999). Understanding Digital Subscriber Line Technology, *Prentice Hall*, New York.

Tu, J. C.; and Cioffi, J. M. (1990). A Loading Algorithm for the Concatenation of Coset Codes with Multichannel Modulation Methods, *IEEE Global Telecommunications Conference*, pp. 1183-1187, December 1990

Valenti, C. (2002). NEXT and FEXT Models for Twisted-Pair North American Loop Plant, *IEEE Selected Areas in Communications Journal*, Vol. 20, No. 5, June 2002, pp. 893-900

Yu, W.; Cioffi, J. M.; (2001). On Constant Power Water-filling, *Proc. IEEE International Conference on Communications,* Vol. 6, pp. 1665–1669, June 2001

Yu, W.; Ginis, G.; Cioffi, J. M. (2002). Distributed Multiuser Power Control for Digital Subscriber Lines, *IEEE Journal of Selected Areas in Communications*, Vol. 20, No. 5, pp. 1105-1115, June 2002

Zeng, C.; Aldana, C.; Salvekar, A. and Cioffi, J. M. (2001). Crosstalk Identification in xDSL Systems, *IEEE Journal of Selected Areas in Communications*, , Vol. 19, No. 8, pp. 1488-1496, August 2001

8

Energy Efficient Greedy Approach for Sensor Networks

Razia Haider and Dr. Muhammad Younus Javed
National University of Science and Technology
Pakistan

1. Introduction

Applications of sensor networks ranging from environmental/military monitoring to industrial asset management. The development of sensor networks was originally motivated by military applications. However, now a days sensor networks are used in many civilian application areas, including environment and habitat monitoring, healthcare applications, home automation, and traffic control (Lewis,2004). It is mentioned above that the main purpose of deployment of sensor networks is to provide feed back and monitoring of environmental variables in areas, which are intractable to humans. The design of energy efficient routing algorithms is an important issue in sensor networks with such deployments.

In wireless ad-hoc and sensor networks geographic routing is a key paradigm that is quite commonly adopted for information delivery, where the location information of the nodes are available (either a-priori or through a self-configuring localization mechanism). The implication of geographic routing protocols is efficient in sensor networks for several reasons. Firstly, nodes need to know only the location information of their direct neighbors in order to forward packets and therefore the state stored is minimum. Secondly, since discovery floods and state propagation are not required beyond a single hop hence, such protocols conserve energy and bandwidth.

Due to energy constraints in these networks geographic routing in sensor networks has been a challenging issue for researchers. The design of routing strategy may also effect by deployment methodology. Sensors may be deployed randomly or deterministically based on the application in the sensor networks field. These random deployments might result in irregular topologies which in turn affect the routing strategy. Sensors perform both data sending and data routing. Inter-sensor communication is usually short ranged. The nodes in the network cooperate in forwarding other nodes' packets from source to destination. Hence, certain amount of energy of each node is spent in forwarding the messages of other nodes. Lots of work has been done in this respect but still energy depletion of sensor nodes is a big challenge in sensor networks.

The aim of this research is to present such geographic algorithm for sensor networks which will be simple, easy to implement and efficient in terms of energy consumptions. In this research various geographic routing protocols for sensor networks have been studied with their applications to have better understanding of these protocols, This research will explore the paradigm of routing in sensor networks in terms of energy efficiency.

The chapter comprises of 6 parts, description of each part is given as follow. Part 1 gives the introduction to sensor networks and objective of the research. Part 2 gives the description of sensor networks and different issues related to it. At the end some of the application areas of sensor networks have been discussed. Part 3 contains the details study of some routing protocols for sensor networks and explores their potential limitations. Part 4 presents the motivation of this research and defines the problem definition. Later it describes the proposed solution design and implementation in detail. Part 5 captures the detail of different test results and provides their analysis. Finally, conclusion and future work are discussed in Part 6.

2. Sensor networks and their applications

A number of new applications that benefit a large number of fields have come into existence with the emergence of sensor networks. A key concern in wireless sensor networks is energy efficiency. In a sensor network the nodes did not charged once their energy is drained so the lifetime of the network depends critically on energy conservation mechanism (Chan et al., 2005; Melodia et al., 2004; Wu et al., 2004).
With deployments of sensor networks in mission critical applications, they gained importance and provide for immense potential for research in this area. Two challenging issues are identified in this realm. First, being the reduction in consumption of power by these sensors to increase their lifetime. Second, being the design of routing strategies for communication in the network. In this part a brief description of sensor networks, challenges faced by them and some of the important application of sensor networks has been discussed.

2.1 Sensor nodes

A device that is capable of observing and recording a phenomenon is known us sensor. This is termed as *sensing*. Sensors are used in various applications such as in rescue operations, seismic sensors are used to detect survivors caught in landslides and earthquakes With the advancements in technology it is easy to manufacture low cost and high performance sensors but only with limited resources which include energy supply and communication bandwidth.
Sensor nodes can be imagined as small computers, extremely basic in terms of their interfaces and their components. They usually consist of processing unit with limited computational power and limited memory, sensors (including specific conditioning circuitry), a communication device (usually radio transceivers or alternatively optical), and a power source usually in the form of a battery (Han et al., 2006). A sensor node usually consists of four sub-systems. Computing subsystem, communication subsystem, sensing subsystem and power supply subsystem.

2.2 Wireless sensor networks

Sensor networking is an emerging technology that has a wide range of potential applications including environment monitoring, smart spaces, medical systems and robotic exploration. A wireless sensor network, WSN, is an ad-hoc network with many sensors deployed in an area for a specific reason. A sensor network consists of possibly several hundred sensor nodes, deployed in close proximity to the phenomenon that they are

designed to observe. The position of sensor nodes within a sensor network need not be pre-determined (Zeng et al., 2006). Sensor networks must have the robustness to work in extreme environmental conditions with scarce or zero interference from humans. This also means that they should be able to overcome frequent node failures. Thus, network topological changes become common. Sensor networks must conserve energy since they are limited in energy, usually the battery as the sole supply of energy. Sensor nodes may also have limited mobility, which allow them to adjust to topology changes.

2.3 Challenges

The unique features of sensor networks pose challenging requirements to the design of the underlying algorithms and protocols. Several ongoing research projects in academia as well as in industry aim at designing protocols that satisfy these requirements for sensor networks. In spite of the diverse applications, sensor networks pose a number of unique technical challenges due to the following factors (Akyildiz et al., 2002).

The sensor nodes are not connected to any energy source. There is only a finite source of energy, which must be optimally used for processing and communication. An interesting fact is that communication dominates processing in energy consumption. Thus, in order to make optimal use of energy, communication should be minimized as much as possible. Environmental stress on sensor nodes cause frequent node failures leading to connectivity changes. These require frequent reconfiguration of the network and re-computation of routing paths. The high probability of node failures in sensor networks requires that the cost of sensor nodes is minimal. This will enable redundancy of sensor nodes to account for node failures. In some cases, sensor nodes have the ability to move, although their mobility is restricted in range to a few meters at the most. Mobility of sensor nodes raises the possibility that nodes might go out of range and new nodes might come within the range of communication. The routing protocols for sensor networks must take these changes into account when determining routing paths. Thus, unlike traditional networks, where the focus is on maximizing channel throughput or minimizing node deployment, the major consideration in a sensor network is to extend the system lifetime as well as the system robustness.

A number of solutions propose to one or more of the above problems. The survey focuses on the suggested solutions is energy efficiency which is a dominant consideration no matter what the problem is. This is because sensor nodes only have a small and finite source of energy. Many solutions, both hardware and software related, have been proposed to optimize energy usage. Traditional routing schemes are no longer useful since energy considerations demand that only essential minimal routing be done.

2.5 Important applications of sensor networks

Wireless sensor networks have significant impact upon the efficiency of military and civil applications such as environment monitoring, target surveillance, industrial process observation, tactical systems, etc. A number of applications have been discussed, (Barrenechea et al., 2004; Barett et al., 2003;Braginsky et al., 2002;intanagonwiwat et al., 2000; Krishnamachari et al., 2002; Servetto & Barrenechea, 2002; Ye et al., 2002). There are different potential applications of sensor networks in many areas due to their different communication model. A number of applications are in military where sensors are widely used in applications such as surveillance, communication from intractable areas to base-

stations. Since these are inexpensive and deployed in large numbers, loss of some of these sensors would not affect the purpose for which they were deployed. In distributed surveillance highly mobile sensor networks make it possible to transmit huge amounts of data at low powers. Structure monitoring systems detect, localize, and estimate the extent of damage. Civil engineering structures can be tested for soundness using sensors. Sensors also used to monitor pollution and toxic level. These sensors collect data from industrial areas and areas where toxic spills occur.

3. Literature review

Like mobile ad-hoc networks, sensor networks also involve multi-hop communications. Many routing algorithms have been proposed for mobile networks. Yet, these algorithms are not applicable to sensor networks due to several factors (Akyildiz et al., 2002). Some of these factors are as the size of the sensor network is usually larger than that of ad-hoc networks. High density of sensor nodes are deployed in sensor networks as compared to mobile hosts. Sensor nodes have energy constraints and are highly susceptible to failures. In addition, they are generally static compared to mobile network. Sensor nodes use reverse multi-cast communication while ad-hoc networks use peer to peer communication. These nodes have several constraints with respect to power, memory, CPU processing which prohibits them from handling high data rate. Hence, sensors have low data rate than that of mobile hosts.
All these factors distinguish sensor networks from mobile networks, and make most of the routing protocols of mobile networks inapplicable to sensor networks. Hence, new routing algorithms are investigated for sensor networks.

3.1 Routing mechanism in sensor networks

Generally data centric routing is used in sensor networks (Barrenechea et al., 2004). Unlike the mobile ad hoc networks, in sensor networks sensor nodes are most likely to be stationary for the entire period of their lifetime. Even though the sensor nodes are fixed, the topology of the network can change. During periods of low activity, nodes may go to inactive sleep state, to conserve energy. When some nodes run out of battery power and die, new nodes may be added to the network. Although all nodes are initially equipped with equal energy, some nodes may experience higher activity as result of region they are located in.
As mentioned before, conventional routing protocols have several limitations when being used in sensor networks due to the energy constrained nature of these networks. These protocols essentially follow the flooding technique in which a node stores the data item it receives and then sends copies of the data item to all its neighbors. There are two main deficiencies to this approach implosion and resource management..
In implosion if a node is a common neighbor to nodes holding the same data item, then it will get multiple copies of the same data item. Therefore, the protocol wastes resources sending the data item and receiving it. In conventional flooding, nodes are not resource-aware. They continue with their activities regardless of the energy available to them at a given time. So there is need of resource management in flooding.
Due to such differences, many new algorithms have been proposed for the problem of routing data in sensor networks. These routing mechanisms have considered the characteristics of sensor nodes along with the application and architecture requirements.

Almost all of the routing protocols can be classified as data-centric, hierarchical or location-based although there are few distinct ones based on network flow or QoS awareness. Data-centric protocols are query-based and depend on the naming of desired data, which helps in eliminating many redundant transmissions. Hierarchical protocols aim at clustering the nodes so that cluster heads can do some aggregation and reduction of data in order to save energy. Location-based protocols utilize the position information to relay the data to the desired regions rather than the whole network. The last category includes routing approaches that are based on general network-flow modeling and protocols that strive for meeting some QoS requirements along with the routing function.

3.2 Data centric protocols

It is not feasible to assign global identifiers to each node due to the sheer number of nodes deployed in many applications of sensor networks. Therefore, data is usually transmitted from every sensor node within the deployment region with significant redundancy. Since this is very inefficient in terms of energy consumption, routing protocols that will be able to select a set of sensor nodes and utilize data aggregation during the relaying of data have been considered. This consideration has led to data centric routing, which is different from traditional address-based routing where routes are created between addressable nodes managed in the network layer of the communication stack. In data-centric routing, the sink sends queries to certain regions and waits for data from the sensors located in the selected regions. Since data is being requested through queries, attribute based naming is necessary to specify the properties of data. The two classical mechanisms flooding and gossiping which are used to relay data in sensor networks without the need for any routing algorithms and topology maintenance (Hedetniemi & Liestman, 1998). In flooding, each sensor receiving a data packet broadcasts it to all of its neighbors and this process continues until the packet arrives at the destination or the maximum number of hops for the packet is reached. On the other hand, gossiping is a slightly enhanced version of flooding where the receiving node sends the packet to a randomly selected neighbor, which picks another random neighbor to forward the packet to and so on.

SPIN (Sensor Protocols for Information via Negotiation) is the first data-centric protocol, which considers data negotiation between nodes in order to eliminate redundant data and save energy (Heinzelman et al., 1999). Later, Directed Diffusion has been developed and has become a breakthrough in data-centric routing (Intanagonwiwat et al., 2000). Directed Diffusion is an important milestone in the data-centric routing research of sensor networks (Estrin et al., 1999). Then, many other protocols have been proposed either based on Directed Diffusion or following a similar concept (Shah & Rabaey, 2002; Schurgers & Srivastava, 2001).

3.3 Hierarchical protocols

Scalability is one of the major design attributes of sensor networks similar to other communication networks. To allow the system to handle with additional load and to be able to cover a large area of interest without degrading the service, networking clustering has been purposed in some routing approaches.

The main objective of hierarchical routing is to efficiently maintain the energy consumption of sensor nodes by involving them in multi-hop communication within a particular cluster and by performing data aggregation and fusion in order to decrease the number of

transmitted messages to the sink. Cluster formation is typically based on the energy reserve of sensors and sensor's proximity to the cluster head. LEACH (Low Energy Adaptive Clustering Hierarchy) is one of the first hierarchical routing approaches for sensors networks (Heinzelman et al., 2000). The idea proposed in LEACH has been an inspiration for many hierarchical routing protocols (Manjeshwar & Agrawal, 2001; Lindsey & Raghavendra, 2002; Lindsey et al., 2001; Manjeshwar & Agrawal, 2002), although some protocols have been independently developed (Subramanian & Katz; 2000; Younis et al., 2002; Younis et al., 2003).

3.4 Location-based protocols

Location- based protocols are most commonly used in sensor networks as most of the routing protocols for sensor networks require location information for sensor nodes. In most cases location information is needed in order to calculate the distance between two particular nodes so that energy consumption can be estimated. Since, there is no addressing scheme for sensor networks like IP-addresses and they are spatially deployed on a region, location information can be utilized in routing data in an energy efficient way. Some of the protocols discussed here are designed primarily for mobile ad hoc networks and consider the mobility of nodes during the design (Xu et al., 2001; Rodoplu & Ming, 1999; Li & Halpern, 2001). However, they are also well applicable to sensor networks where there is less or no mobility. It is worth noting that there are other location-based protocols designed for wireless ad hoc networks, such as Cartesian and trajectory-based routing. However, many of these protocols are not applicable to sensor networks since they are not energy aware. In order to stay with the theme of the research, the scope has limited to the coverage of only energy-aware location based protocols.

3.4.1 Geographic and energy aware rotuing

Yu et al. have suggested the use of geographic information while disseminating queries to appropriate regions since data queries often includes geographic attributes (Yu et al., 2001). The protocol, namely Geographic and Energy Aware Routing (GEAR), uses Energy Aware and geographically informed neighbor selection heuristics to route a packet towards the target region. The idea is to restrict the number of interests in Directed Diffusion by only considering a certain region rather than sending the interests to the whole network. GEAR compliments Directed Diffusion in this way and thus conserves more energy.

3.4.2 Geographic adaptive fidelity

Geographic Adaptive Fidelity (GAF) is an energy-aware location-based routing algorithm designed primarily for mobile ad hoc networks, but may be applicable to sensor networks as well (Xu et al., 2001). GAF conserves energy by turning off unnecessary nodes in the network without affecting the level of routing fidelity.

4. Design and implementation

In sensor networks, building efficient and scalable protocols is a very challenging task due to the limited resources and the high scale and dynamics. In this realm, geographic protocols [Xu et al., 2001; Yu et al., 2001) take advantage of the location information of nodes, are very valuable for sensor networks. The state required to be maintained is minimum and their overhead is low in addition to their fast response to dynamics.

4.1 Geographic routing

Basic geographic protocol at the network layer has been examined for geographic routing based on greedy mechanisms. Geographic routing provides a way to deliver a packet to a destination location, based only on local information and without the need for any extra infrastructure. It makes geographic routing the main basic component for geographic protocols. With the existence of location information, geographic routing provides the most efficient and natural way to route packets comparable to other routing protocols.

Geographic routing protocols require only local information and thus are very efficient in wireless networks. First, nodes need to know only the location information of their direct neighbors in order to forward packets and hence the state stored is minimum. Second, such protocols conserve energy and bandwidth since discovery floods and state propagation are not required beyond a single hop.

It is based on assumption that the node knows the geographical position of the destination node. This approach to routing involves relaying the message to one of its neighbors that is geographically closest to the destination node of all neighbors, and is geographically closer to the destination. This approach attempts to find a short path to the destination, in terms of either distance or number of hops. It is based on the geographical distances between the nodes.

A node that requires sending a message acquires the address of the destination. After preparing the message, it calculates the distance from self to the destination. Next, it calculates distance from each of its neighbors to the destination. The greedy approach always tries to shorten the distance to be traveled to the destination to the maximum possible extent. Therefore, the node considers only those neighbors that are closer to the destination than itself. The sending node then chooses the node closest to the destination and relays the message onto the neighbor.

A node receiving a message may either be the final destination, or it may be one of the intermediate nodes on the route to the destination. If the node is an intermitted hop to the message being relayed, the node will calculate the next hop of the message in the manner described above.

A sample topology is shown in Figure 1. Nodes A and B are the sender and receiver respectively. Node A sends the message to node Y as it is the closest of its neighbors to the destination node B. On receiving the message, Y calculates its closest neighbor and forwards message to it. This process will continue until the massage reached to the final destination B. The dotted arrows show the shortest path followed by the node.

The basic geographic routing doest not use any data structures stored locally on a node apart from the neighbor table. Thus, no information is stored locally. The sending component doest not differentiate between the source of the message and an intermediate node on its route. The receiving component needs to handle to two different types of messages; one that says that the node is the destination, and the other that specifies the node to be an intermediate node for relaying the message. Both messages are handled in exactly the same way, without any form of distinction.

A typical sensor network consisting of sensor nodes scattered in a sensing field in the vicinity of the phenomenon to be observed is shown in Figure 2 (http://www.acm.org/crossroads/xrds9-4/sensornetworks.html). The nodes are connected to a larger network like the Internet via a gateway so that users or applications can access the information that is sent from the sensor nodes. The dotted circle shows the area where

sensor nodes are scattered to sense the specific task and then route the sensed processed data to the gateway. The main focus is on this dotted area and this research has proposed an Energy efficient greedy scheme for inter-sensor nodes communication where information relay between these sensor nodes. Proposes algorithm will provide simple and effcient path to nodes for forwarding their messages which will furthure conserve total energy of the entire network.

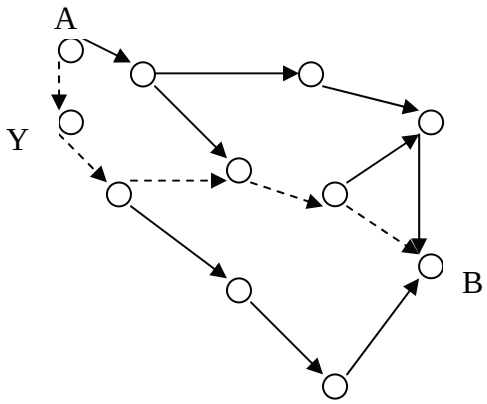


Fig. 1. Sample Route for Basic Geographic Routing

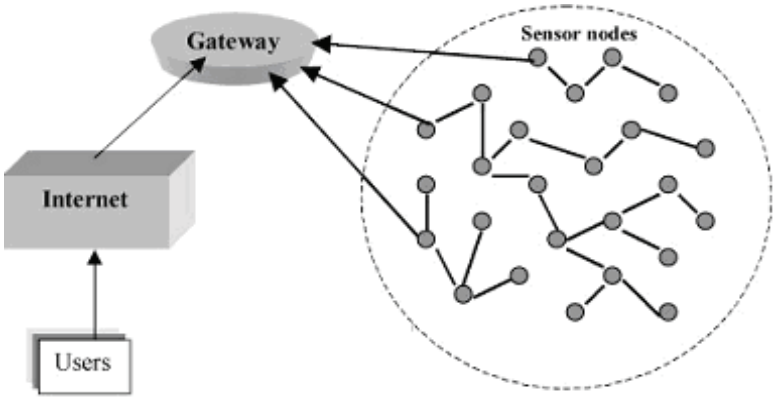


Fig. 2. Sensor Nodes Connected in a Network.

4.2 Routing scheme for inter-sensor nodes communication

Energy consumption is the most important factor to determine the life of a sensor network because usually sensor nodes are driven by battery and have very low energy resources. This makes energy optimization more complicated in sensor networks because it involves not only reduction of energy consumption but also prolonging the life of the network as much as possible. This can be done by having energy awareness in every aspect of design and operation. This ensures that energy awareness is also incorporated into groups of communicating sensor nodes and the entire network and not only in the individual nodes.

4.2.1 Weak node problem

The main component of geographic routing is usually a greedy forwarding mechanism whereby each node forwards a packet to the neighbor that is closest to the destination. However, while assuming highly dense sensor deployment and reasonably accurate localization several recent experimental studies on sensor networks have shown that node energy can be highly unreliable and this must be explicitly taken into account when considering higher layer protocol (Li & Halpern, 2001). The existence of such unreliable nodes exposes a key weakness in greedy forwarding. At each step in greedy forwarding, the neighbors those are closest to the destination may not have sufficient energy to transmit the messages. These weak nodes would result in a high rate of packet drops, resulting in drastic reduction of delivery rate or increased energy wastage if retransmissions are employed.

In sensor networks ,sensor nodes use their energy in forwarding messages in network but at some point when node deplete its all energy it fails to transmit the further messages resulting in loss of data (formation of holes). In this research work, the geographic routing thorough the greedy forwarding (Karp & Kung, 2000) has been considered for implementation. Usually, in the greedy forwarding the closest neighbor node will be heavily utilized in routing and forwarding messages, while the other nodes are less utilized. Due to this uneven load distribution it results in heavily loaded nodes to discharge faster when compared to others. This causes few over-utilized nodes which fail and result in formation of holes in network, resulting in increase number of failed/dropped messages in the network. Energy efficient routing scheme should be investigated and developed such that it loads balances the network and prevents the formation of holes. In this research, the above mentioned problems faced by greedy forwarding approach will be taken care of in sensor networks.

4.2.2 Energy efficeint greedy scehem: basic principle

The concept of neighbor classification based on node energy level and their distances has been used in Energy Efficient Greedy Scheme (EEGS) has been used to cater of the weak node problem. Some neighbors may be more favorable to choose than the others, not only based on distance, but also based on energy characteristics. It suggests that a neighbor selection scheme should avoid the weak nodes. If the geographic forwarding scheme purely based on greedy forwarding attempts to minimize the number of hops by maximizing the geographic distance covered at each hop, it is likely to incur significant energy expenditure due to retransmission on the weak nodes. On the other hand, if the forwarding mechanism attempts to maximize per hop reliability by forwarding only to close neighbors with good nodes, it may cover only small geographic distance at each hop. It would also result in greater energy expenditure due to the need for more transmission hops for each packet to reach the destination. So in both cases energy is not being conserved to increase the lifetime of the network. Therefore, the strategy used in the proposed Energy Efficient Greedy Scheme (EEGS) first calculates the average distance of all the neighbors of transmitting node and checks their energy levels. Finally, it selects the neighbor which is alive (i.e. having energy level above than the set threshold) and having the maximum energy plus whose distance is equal to or less than the calculated average distance among its entire neighbors. Hence, the proposed scheme uses Energy Efficient routing to select the neighbor that has sufficient energy level and is closest to the destination for forwarding the query.

4.2.3 Assumptions for EEGS

Some basic assumptions have been considered for the implementation of EEGS in this research work. Sensor nodes are static in the network (i.e. Once the node has learned its location, its co-ordinates do not change). The central location database has been managed by a central entity which enables each of the nodes to discover its position. In the real scenario, each node would learn its location by some kind of GPS system so the above assumptions can be made without the loss of generality. The irregular random topology for sensor networks has been considered. Single destination scenario is taken into the account. There are infinite-size buffers at each node to support the incoming and outgoing message packets. Hence, buffer overflows and queuing analysis are not the part of this research. In the proposed system the fixed size of packets are used. So the packet sizes will not be considered during the analysis.

4.3 General mechnism of purposed syetem

There are four basic modules have been implemented in the proposed system i.e. network generator, route generator, routing algorithm and router. The platform of Visual Studio 6.0 and OMNET++ network simulator has been used for implementation of purposed system.

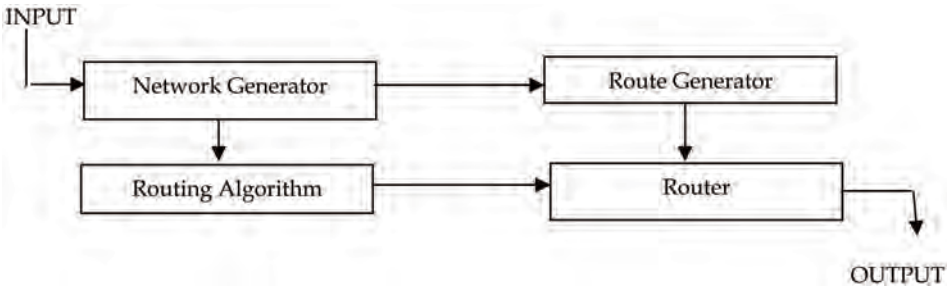


Fig. 3. Block Diagram of the System

The brief description of each module is as follow.

(a) Network Generator: This module generates the network. It has two type of parameters i.e. data rate and number of nodes. It includes a sub module network node which defines structure of single network node. Sub module network node has three types of parameters i.e. address of the node, then number of stations which is equal to number of nodes in this case and data rate. Two types of gates are defines for each node i.e. in gate and out gate. Input and output interfaces of modules are through these gates; messages are sent out through output gates and arrive through input gates, (b) Route Generator: In this module the source and destination are specified for the data sending and receiving. Each node randomly sent the massages to the destination node, (c) Routing Algorithm: This module takes location information of nodes in the network and set their weights. Then it chooses the next hop on the basis of EEGS, (d) Router: This module routes the packet to other nodes and generates the next hop destination map which comes from routing algorithm and then route/forward the received packet to the selected neighbor nodes. This map gives the complete information to each node about its own location and location of its neighbor nodes with their energy levels which are being updates after every transmission. Finally, router module gives the output in the form of packets delivered successfully, packet dropped, remaining energy level and status of the node.

4.4 Energy model

For the implementation purpose, a simple energy model has been used. Each node starts with the same initial energy and forwards a packet by consuming same energy. Initially, all nodes have energy level equal to 1 joule (Yu et al., 2001). Each node depletes energy in transmitting one packet which is equal to 0.1mjoule.

5. Simulations and results

Different simulations results are presented with different number of nodes in order to check performance of the proposed algorithm. Location of nodes has been taken randomly in each network. Performance of two algorithms (i.e. Greedy & EEGS) has been compared in terms of successful delivery rate and number of nodes.

5.1 Implementation with OMNET++

OMNET++ is an object-oriented modular discrete event simulator. The name itself stands for Objective Modular Network Testbed in C++. The simulator can be used for traffic modeling of telecommunication networks, protocol modeling, modeling queuing networks, modeling multiprocessors and other distributed hardware systems, validating hardware architectures, evaluating performance aspects of complex software systems, modeling any other system where the discrete event approach is suitable.

5.2 Simulation model

In simulation model, the numbers of nodes chosen ranged from 16 to 100 sensor nodes. Random topology has been considered in this implementation. The immobile sensor network has been considered, so every sensor node is static. Initially, each node has same energy level as specified in energy model. Any node having energy less than or equal to set threshold will be considered as dead. One node is located as the destination node for all nodes i.e. one node is declared as target node for all data receiving as mentioned in assumptions that one destination scenario has been considered. The packet size is of 562 bytes. Total simulation time is set to 500 seconds and each scenario is repeated ten times with different randomly deployed nodes. As mentioned above discrete even driven simulator OMNET++ has been used in this research for implementation purpose. It simulates routing packets among different nodes in the network.

Figure 4 shows the sample network with 30 nodes. Nodes start sending packets randomly to destination node by choosing the neighbor nodes on the basis of EEGS approach. Initially each node has energy of 1J. EEGS approach calculates the average distance of all neighbor nodes of the sending node and checks their energy levels. Then, it selects the neighbor which is alive (i.e. having energy level above than the set threshold) and having the maximum energy plus whose distance is equal to or less than the calculated average distance among its entire neighbors. This process will continue until the packet reaches to the destination node.

5.3 Evaluation metrics and measurement criteria

There are four performance metrics have been defined in order to measure performance of the proposed algorithm. These metrics includes number of packets delivered successfully, number of packets dropped, number of nodes alive and number of nodes dead.

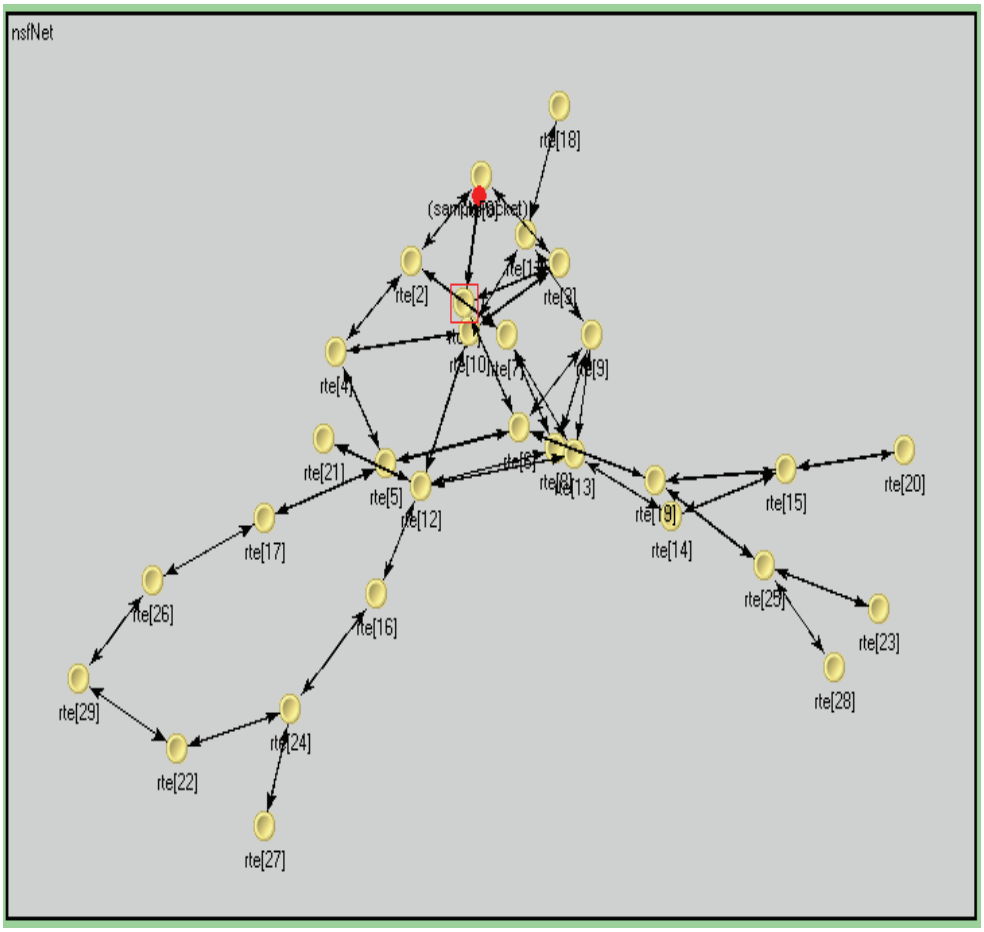


Fig. 4. Sample Network with the Network Size of 30 Nodes.

In Table 1 the results of simulation for network size of 30 nodes has been shown. Each node starts with the energy equal to 1J. These simulation results clearly show that EEGS approach provides better packet delivery rate as compared to Greedy algorithm. In these results it is also worth noticing that EEGS approach is more reliable by having more number of nodes alive, thus it results in longer life of the network as compared to the Greedy algorithm.
Total Packets Generated: 15,000

	Packets Successfully Delivered	Packets Dropped	Nodes Dead	Nodes Alive
Greedy	10500	4500	2	28
EEGS	14536	464	1	29

Table 1. The Simulation Results with Network Size of 30 Nodes

It is evident from Table 2 that the proposed EEGS approach provides better data delivery rate than the Greedy algorithm. The successful packet delivery of EEGS is 90% while Greedy algorithm has 72% on average. The main focus is on varying size of network by keeping other parameters constant. The main aim is to design an algorithm that can scale to thousands of nodes in future sensor networks, therefore the research has been focused on how the algorithm scales and perform better with networks of different sizes. It has been observed that the difference of amount of packets delivered successfully is getting larger as the number of nodes increases. It means that EEGS improves the performance much more as the number of source nodes increases. Also EEGS approach is more reliable in terms of energy consumption as it has less number of nodes dead as compared to Greedy algorithm. Hence, it provides longer the life to the sensor network as compared to the Greedy algorithm.

Number of Nodes		Total Number of Packets Generated by both Schemes	Packets Successfully Delivered	Packets Dropped	Nodes Dead	Nodes Alive
20	Greedy EEGS	10,000	9251 10000	749 0	1 0	19 20
35	Greedy EEGS	17,500	15834 17369	1666 131	1 1	34 34
45	Greedy EEGS	22,500	18877 21599	3623 901	3 1	42 44
55	Greedy EEGS	27,500	21253 23979	6247 3521	5 2	50 53
65	Greedy EEGS	32,500	31019 32500	1481 0	1 0	64 65
75	Greedy EEGS	37,500	25917 26421	11583 11079	3 2	72 73
85	Greedy EEGS	42,500	27325 38468	15175 4032	3 1	82 84
95	Greedy EEGS	47,500	31019 35872	16481 11628	5 3	90 92
100	Greedy EEGS	50,000	32237 40445	17763 4555	5 2	95 98

Table 2. The Complete Simulation Results

5.4 Results and performance comparsion

In geographic routing greedy communication model has been used as the basic comparison model. Greedy algorithm is purely geographic based and does not consider the energy consumption of the nodes. As per the minimum criteria, proposed communication scheme should be having greater successful packet delivery than Greedy algorithm and should have less number of dead nodes.

The results presented in the Table 2 are shown in form of graphs in order to have the clear comparison between the EEGS and Greedy algorithm, which shows that proposed EEGS approach has performance clearly better than Greedy algorithm.
In Figure 5 a comparison has been shown between the total numbers of packets that are successfully delivered in both algorithms. It is clear from the graph that the proposed EEGS approach has much higher successful delivery rate than the Greedy algorithm.

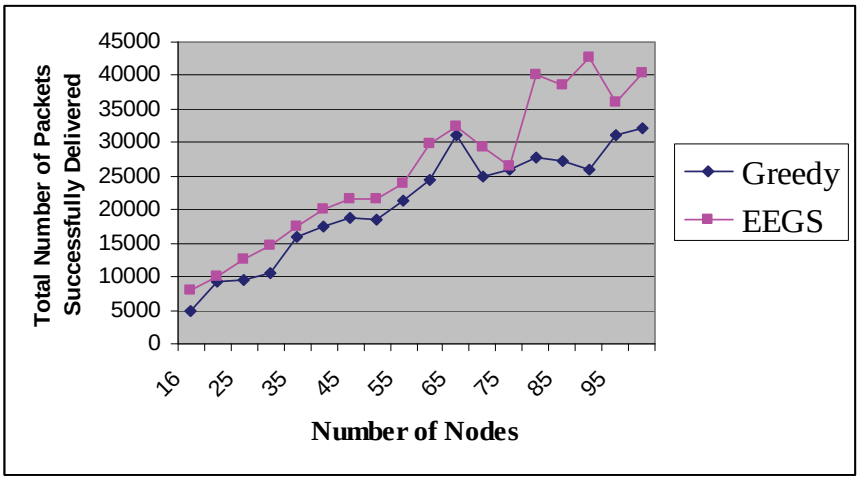


Fig. 5. Successful Packet Delivery in EEGS and Greedy Algorithm.

It is also indicated in Figure 6 that the packet drop rate is very less in EEGS approach as compared to the Greedy algorithm. Hence, EEGS approach conserves more energy and more efficient then Greedy algorithm.

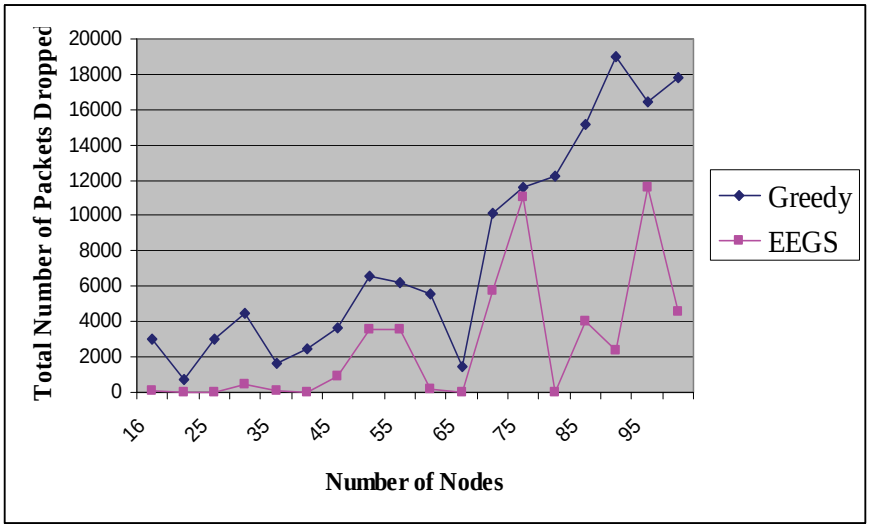


Fig. 6. Packets Dropped in EEGS and Greedy Algorithm.

From Figure 7 it is observed that EEGS approach is more reliable and results in longer life of the network as the total number of nodes which are alive are greater as compared to Greedy algorithm.

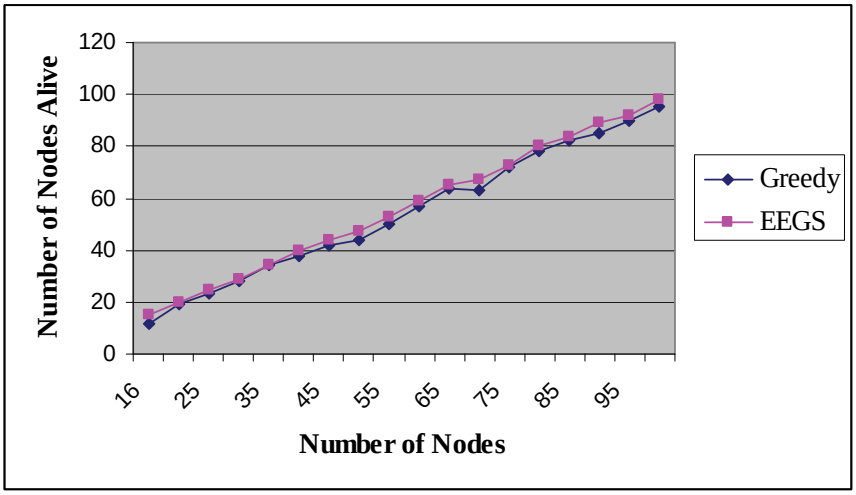


Fig. 7. Nodes Alive in EEGS and Greedy Algorithm.

Figure 8 clearly shows that Greedy algorithm has greater number of dead nodes as compared to EEGS approach. Hence, network lifetime is greater for EEGS than the Greedy algorithm.

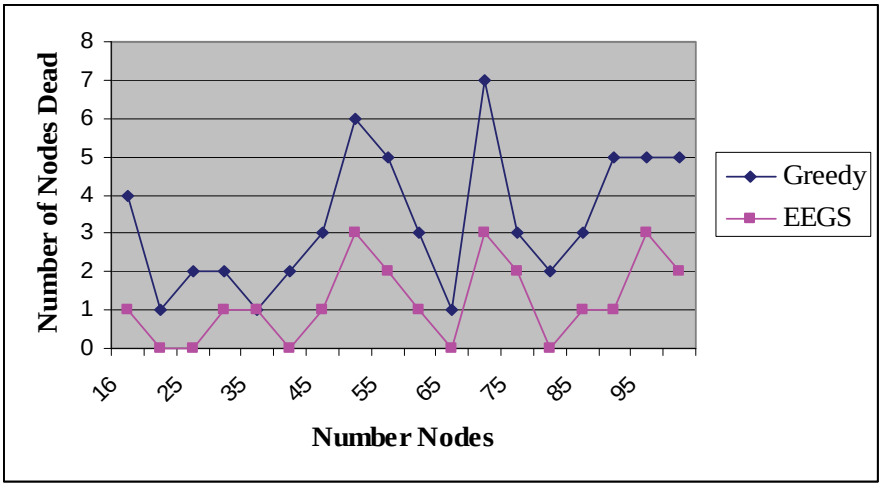


Fig. 8. Dead Nodes in EEGS and Greedy Algorithm.

5.5 Comparison between greedy algorithm & EEGS and analysis

In Greedy algorithm, packets are marked by their originator with their destinations' locations. As a result, a forwarding node can make a locally optimal, greedy choice in

choosing a packet's next hop closest to the destination. Forwarding in this regime follows successively closer geographic hops until the destination is reached. On the other hand, EEGS has the forwarding rule based on location as well as energy levels of nodes. In energy aware algorithm, each node knows its neighbor positions and their energy levels. The transmitting node writes the geographic position of the destination node into the packet header and forwards it to the neighbor which has the distance equal to or less than the average distance of all neighbors of that transmitting node plus having the maximum energy level among all its neighbors.
The geographical position provides the direction when the data is relayed in that direction until being reached to the gateway. The nodes add the geographical position of the gateway in the packet header and forward it to the neighbor who resides geographically closest to the destination and has the maximum energy (from the neighbors which has distance equal to or less than the average distance of neighbors).
During the packet transmission, each node chooses the next hop based on the routing policy used. The procedure repeats until the packet reaches the destination node. The packet transmission energy required between the two nodes is calculated by the energy model specified in 4.4.
The Greedy algorithm and EEGS approach have been run by using different number of sensor nodes with the same energy per node initially. It has been noted that EEGS approach as compared to the Greedy algorithm gives better results in terms of successful packet delivery and less number of dead nodes. Hence, proposed EEGS approach is more reliable as compared to Greedy algorithm and results in longer lifetime of the network.

6. Conclusion and future work

6.1 Conclusion

A sensor network is a promising technology for applications ranging from environmental/military monitoring to industrial asset management. Due to sensor networks communication model, these networks have potential applications in many areas. Sensor networks, similar to mobile ad-hoc networks involve multi-hop communications. There have been many routing algorithms proposed for mobile networks. Yet, these algorithms are not applicable to sensor networks due to several factors. Due to these factors sensor networks are distinguished from mobile networks, and make most of the routing protocols of mobile networks inapplicable to sensor networks. Hence, new routing algorithms are investigated for sensor networks. Almost all of the routing protocols can be classified as data-centric, hierarchical or location-based although there are few distinct ones based on network flow or QoS awareness. Geographic routing in sensor networks has been a challenging issue for researchers considering the energy constraints in these networks. The nodes in the network cooperate in forwarding other nodes' packets from source to destination. Hence, certain amount of energy of each node is spent in forwarding the messages of other nodes. Lots of work has been done in this respect but still energy depletion of sensor nodes is a big challenge in sensor networks. Sensor nodes use their energy in forwarding messages in network but at some point when node deplete its all energy it fails to transmit the further messages resulting in loss of data (formation of holes). In this research work, the geographic routing thorough the greedy forwarding has been

considered for implementation. . In greedy forwarding uneven load distribution results in heavily loaded nodes to discharge faster when compared to others. This causes few over-utilized nodes which fail and result in formation of holes in network, resulting in increase of failed messages in the network. So there was a need of such energy efficient routing strategy that should be balance the load of the network and prevents the formation of holes. Therefore this research work has investigated an Energy Efficient Greedy Scheme (EEGS) for geographic routing in sensor networks. The Greedy algorithm and EEGS approach have been implemented and simulation results have been obtained. From these results it has been shown that proposed EEGS approach performs better and efficiently than the Greedy routing. The simulations based upon the different number of nodes by employing these two algorithms considering different parameters (i.e. the successful packet delivery and number of nodes alive as the performance criterion). Therefore, performance of EEGS approach is much better than the Greedy algorithm in the defined parameters. Consequently, it can be concluded that EEGS can efficiently and effectively extend the network lifetime by increasing the successful data delivery rate.

6.2 Future work

The emerging field of sensor networks has lot of potential for research. In this research work, has considered fixed sizes of packets by using very simple energy model for energy computation purposes. This work can be extended by considering the variable length of packets and the changing distance of transmitting node from its neighbors. For this purpose, there is a need of such an energy model that can calculate the energy consumption of nodes based on their sizes and distances.

7. References

Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y. & Cayirci, E. (2002). "Wireless sensor networks: a survey," *Computer Networks*, 38(4):393-422, March 2002

Barett, C. L.; Eidenbenz, S. J.; Kroc, L.; Marathe, M. & Smith, J. P.(2003). "Parametric Probabilistic Sensor Network Routing," *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pp. 122–131, San Diego, California, September 2003

Barrenechea, G.; Beferull-Lozano, B. & Vetterli, M.(2004). "Lattice Sensor Networks: Capacity Limits, Optimal Routing and Robustness to Failures," *Proceedings of the third international symposium on Information processing in sensor networks,* pp. 186–195, Berkeley, California, April 2004

Braginsky, D. & Estrin, D.(2002). "Rumor Routing Algorithm for Sensor Networks," *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications,* pp.22–31, Atlanta, Georgia, 2002

Chan, K.S.; Nik, H. P. & Fekri, F. (2005). "Analysis of hierarchical Algorithms for Wireless Sensor Network Routing Protocols", *in the Proceedings of IEEE Communications Society /WCNC,* 2005.

Estrin, D. et al. (1999). "Next century challenges: Scalable Coordination in Sensor Networks," *in the Proceedings of the 5th annual ACM/IEEE international conference on Mobile Computing and Networking (MobiCom'99),* Seattle, WA, August 1999

Han, U. P.; Park, S. E.; Kim, S. N. & Chung, Y. J. (2006). "An Enhanced Cluster Based Routing Algorithm for Wireless Sensor Networks", *in the Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications and Conference on Real Time Computing systems ad Applications, PDPTA 2006*, pp.758-763, Lasvegas, Nevada, USA, June 2006

Hedetniemi, S. & Liestman, A. "A survey of gossiping and broadcasting in communication networks," *Networks*, Vol. 18, No. 4, pp. 319-349, 1988

Heinzelman, W.; Kulik, J. & Balakrishnan, H. (1999). "Adaptive protocols for information dissemination in wireless sensor networks," *in the Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99),* Seattle, WA, August 1999

Heinzelman, W.; Chandrakasan, A. & Balakrishnan, H. (2000). "Energy-efficient communication protocol for wireless sensor networks," *in the Proceeding of the Hawaii International Conference System Sciences,* Hawaii, January 2000

http://www.acm.org/crossroads/xrds9-4/sensornetworks.html

Intanagonwiwat, C.; Govindan, R. & Estrin, D. (2000). "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Proceedings of the 6th annual international conference on Mobile computing and networking,* pp. 56–67, Boston, Massachusetts, August 2000

Karp, B. & Kung, H. T. (2000). "GPSR: Greedy perimeter stateless routing for wireless sensor networks," *in the Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '00),* pp.243-254, Boston, MA, August 2000

Krishnamachari, B.; Estrin, D. & Wicker, S. (2002). "Modeling Data-Centric Routing in Wireless Sensor Networks," *Proceedings of the 2002 IEEE INFOCOM,* New York, NY, June 2002

Lewis, F. L. (2004) "Wireless Sensor Networks", *in the Proceedings of Smart Environment Technologies, Protocols, and Applications,* 2004(to appear), New York

Li, L & Halpern, J. Y. (2001). "Minimum energy mobile wireless networks revisited," *in the Proceedings of IEEE International Conference on Communications (ICC'01),* Helsinki, Finland, June 2001

Lindsey, S.; Raghavendra, C. S. & Sivalingam, K, (2001). "Data Gathering in Sensor Networks using the Energy*Delay Metric", *in the Proceedings of the IPDPS Workshop on Issues in Wireless Networks and Mobile Computing,* San Francisco, CA, April 2001

Lindsey, S. & Raghavendra, C. S. (2002). "PEGASIS: Power Efficient GAthering in Sensor Information Systems," *in the Proceedings of the IEEE Aerospace Conference*, Big Sky, Montana, March 2002

Manjeshwar, A. & Agrawal, D. P. (2001). "TEEN : A Protocol for Enhanced Efficiency in Wireless Sensor Networks," *in the Proceedings of the 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing,* San Francisco, CA, April 2001

Manjeshwar, A. & Agrawal, D. P. (2002). "APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks," *in the Proceedings of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile computing,* Ft. Lauderdale, FL, April 2002

Melodia, T.; Pompili, D. & Akyildiz, F. Ian, (2004). "Optimal Local Topology Knowledge for Energy geographic Routing in Sensor networks", *in the proceedings of IEEE INFOCOM*, Hong Kong S.A.R., PRC, March 2004

Rodoplu, V. & Ming, T. H. (1999). "Minimum energy mobile wireless networks," *IEEE Journal of Selected Areas in Communications*, Vol. 17, No. 8, pp. 1333-1344, 1999

Schurgers, C. & Srivastava, M. B. (2001). "Energy efficient routing in wireless sensor networks," *in the MILCOM Proceedings on Communications for Network-Centric Operations: Creating the Information Force*, McLean, VA, 2001

Servetto, S. D. & Barrenechea, G. (2002). "Constrained Random Walks on Random Graphs: Routing Algorithms for Large Scale Wireless Sensor Networks," *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 12–21, Atlanta, Georgia, September 2002

Shah, R. & Rabaey, J. (2002). "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks", *in the Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC),* Orlando, FL, March 2002

Subramanian, L. & Katz, R. H. (2000). "An Architecture for Building Self ConFigureurable Systems," *in the Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing,* Boston, MA, August 2000.

Wu, S.; K. Sel¸ K. & Candan, C. (2004). "GPER: Geographic Power Efficient Routing in Sensor Networks", *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04),* pp.161-172, Berlin, Germany, October 2004.

Xu, Y.; Heidemann, J. & Estrin, D. (2001). "Geography-informed energy conservation for ad hoc routing," *in the Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'01),* Rome, Italy, July 2001.

Ye, F.; Luo, H.; Cheng, J.; Lu, S. & Zhang, L. (2002). "A Two-Tier Data Dissemination Model for Large-scale Wireless Sensor Networks," *Proceedings of the 8th annual international conference on Mobile computing and networking*, pp. 148–159, Atlanta, Georgia, September 2002.

Younis, M.; Youssef, M. & Arisha, K. (2002). "Energy-Aware Routing in Cluster-Based Sensor Networks", *in the Proceedings of the 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS2002)*, Fort Worth, TX, October 2002.

Younis, M.; Munshi, P. & Al-Shaer, E. (2003) "Architecture for Efficient Monitoring and Management of Sensor Networks," *in the Proceedings of the IFIP/IEEE Workshop on End-to-End Monitoring Techniques and Services (E2EMON'03),* Belfast, Northern Ireland, September 2003 (to appear).

Yu, Y.; Estrin, D. & Govindan, R. "Geographical and Energy-Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks," *UCLA Computer Science Department Technical Report,* UCLA-CSD TR-01-0023, May 2001

Zeng, K.; Ren, K.; Lou, W. & Moran, P. J. (2006). " Energy – Aware Geographic Routing in Lossy Wireless Sensor Networks with Environmental Energy Supply", *The Third International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks,* Waterloo, ON, Canada, ACM, August, 2006

9

Enhancing Greedy Policy Techniques for Complex Cost-Sensitive Problems

Camelia Vidrighin Bratu and Rodica Potolea
Technical University of Cluj-Napoca
Romania

1. Introduction

One of the most prominent domains of application for machine learning techniques is data mining, which focuses on the discovery of novel, structured and potentially useful patterns in data. Each machine learning algorithm makes assumptions regarding the underlying problems it needs to solve. These assumptions and the search strategy employed constitute the bias of the learning algorithm. This bias restricts the area of successful application of the algorithm. Also, recent research in machine learning has revealed the necessity to consider more complex evaluation measures for learning algorithms, in some problem domains. One such novel measure is the cost. There are various types of costs involved in inductive concept learning, but, for the domains we focus on, the most important are test costs and misclassification costs.

This chapter presents ProICET (Vidrighin et al, 2007), a hybrid system for solving complex cost-sensitive problems. We focus both on the theoretical principles of the approach, as well as highlight some implementation aspects and comparative evaluations on benchmark data (using other prominent learning algorithms).

The remainder of the chapter is structured as follows: section 2 reviews the search problem, and a few fundamental strategies, and provides basic definitions for data mining and decision trees. Section 3 presents the cost-sensitive problem, together with a brief survey of the most prominent cost-sensitive learners present in literature. Section 4 presents the theory behind ProICET, followed by the enhancements considered and an overview of the implementation. Section 5 presents the experimental work performed in order to validate the approach. The chapter summary is presented in the final section.

2. Basic techniques for search and knowledge extraction

Search is a universal problem-solving mechanism is various domains, including daily life. It also represents one of the main applications in computer science in general, and in artificial intelligence in particular (Korf, 1999). The problem can be formalized as finding a path (in the state space) from the root node to the goal node. In many cases, a particular path is requested, namely the one which obeys some optimization criterion. The main task here comes from the difficulty of finding the right search strategy for the particular problem to solve. In evaluating them several criteria are considered, such as completeness (the ability of the strategy to find the solution in case there is one), time/space complexity (the amount of

time/memory the search strategy needs to find the solution), optimality (the ability of the strategy to find the best solution, according to the optimization criterion).
The next section presents two important search strategies, with their derivatives. Moreover, their performance criteria are discussed and compared. An *uninformed search strategy* (sometimes called blind search) performs in the absence of knowledge about the number of steps or the path cost from the current state to the goal. The most prominent approaches in this category are breadth first search and depth first search. As opposed to uninformed methods, the *informed search strategy* employs problem-specific knowledge. The best first search strategy from this category is reviewed, and one of its simplest, yet effective versions, greedy search. The common pattern in all strategies is the expansion of the current node (i.e. considering its successors as candidates for finding the path to goal), while the particularity consists in the order in which the neighbors are evaluated for expansion.

2.1 Fundamental search strategies

In the *breadth first search* strategy the root node is expanded first. In the second step, all nodes generated by it are expanded, in the third step, their successors, and so on. This means that at every step the expansion process occurs for nodes which are at the same distance from the root, and every expanded node in a step is on the boundary of the covered/uncovered region of the search space. Breadth first search considers a systematic approach, by exhaustively searching the entire state space without considering the goal until it finds it. Due to the fact that the whole space is covered, the strategy is *complete* (i.e. on a finite space, the solution is found, in case there is one). Moreover, the strategy is *optimal*. The drawback is the large complexity, both in time and space: $O(b^d)$, where b represents the branching factor (i.e. number of descendents of a node) and d the depth of the space. Breadth first search can be implemented using a general search strategy with a FIFO queue for the states (Russell & Norvig, 1995).
Uniform cost search comes as a flavor of breadth first search. Assuming a cost function g(n) is considered, breadth first search is modified by expanding the lowest cost node (min g(n)) on the boundary. The default distance to the root, used by the breadth first search is replaced by some specific cost function g(n) (i.e. for breadth first search, g(n)=depth(n) by default). Thus, the systematic approach of covering the space is relaxed to reach the optimal solution faster. Dijkstra's algorithm is a uniform cost search algorithm.
The *depth first search strategy* has a similar approach, but instead of expanding nodes on the boundary, it always expands one node at the deepest level. In case the search reaches a dead end, where no expansion is possible, a node on a shallower level is considered. This way, the "horizontal" approach of covering the states space is replaced by a "vertical" one. Depth first search can be implemented by a general search strategy if a stack is used to keep the states. That is, the FIFO policy is replaced by a LIFO. The major advantage of this strategy is reduced space requirement: $O(bm)$, where m is the maximum depth. The time complexity remains in the exponential domain: $O(b^m)$. The drawback is that the method is *neither complete, nor optimal*. This is the reason why it should be avoided for spaces with large or infinite max depths.
By imposing an upper limit to the maximum depth of a path these pitfalls can be avoided. This modified strategy is implemented by *depth-limited search*. In this situation, the strategy becomes complete, if the depth of the solution is smaller than the threshold imposed, yet it is still not optimal.

The *bidirectional search* simultaneously searches both from the root (forward) and the goal (backward) and stops when the two meet in the middle. It has the advantage of being *optimal* and *complete* at the same time. Moreover, it reduces the time complexity to $O(b^{d/2})$, with the cost of increasing the space complexity to $O(b^{d/2})$. The disadvantage is that the backward search from the goal is not applicable to all problems. This requires expanding the predecessor node, rather than the successor. When the operators are reversible, computing the predecessor is not an issue. However, for some problems, calculating predecessors is very difficult. Moreover, the goal state may not be unique, meaning that the backward strategy should be started from several nodes. Finally, there is no unique way to perform the search in the two halves: the strategy is strongly dependent on the problem. Other issues, such as checking the appearance of a node on the other half, have to be considered for the bidirectional approach as well.

If some knowledge is added to the queuing function, which determines the node to expand next, the chances to find (completeness) the optimal (optimality) solution faster (time complexity) increases and we deal with an informed search strategy. The knowledge usually refers to some performance function, as a measure of the desirability of expanding a node.

Best first search strategy expands the node for which the performance function is estimated to be the best. Emphasis on estimation is important: expansion applies to the most promising node, rather than to be the one which surely leads to the best solution. Thus, the strategy doesn't necessarily deliver the optimal solution (but the one which appears to be the best according to the performance criterion). If the evaluation was precise, and we could expand the best node, it would not be a search strategy at all, but a straight path from the root to the goal. Selecting the best candidate for expansion is done using a priority queue.

Greedy (best first) search is one of the simplest strategies in this category. The knowledge added here is the estimated cost of the cheapest path from the current node to the goal. As mentioned for the generic case, this cost cannot be determined exactly; the function that estimates the cost is called a *heuristic function*, *h(n)*. The greedy strategy takes the best local decision, with no evaluation of further effort. The method resembles depth first search, as it follows a single path in the attempt to reach the goal, backing up in case a dead end is found. Because of the similarities, it has the same drawbacks with depth first search: it is *not optimal*, and *incomplete*. The time complexity is still exponential: $O(b^m)$. Even worse, due to the fact that the strategy memorizes all nodes, the space complexity is also $O(b^m)$. Although the optimality of the solution is not guaranteed, it usually finds a good solution (close to the optimal). Moreover, if some problem-specific knowledge is added, it can obtain the optimal solution. Both Prim's and Kruskal's algorithms for finding the minimum spanning tree are greedy search algorithms. Because it minimizes the estimated cost to the goal, h(n), greedy search decreases the search costs as well, by cutting search branches. This makes the strategy efficient, although not optimal.

One trap greedy strategy falls in is estimating the performance function from the current node to the goal, without taking into account the component of the function from the root to the current node (which can actually be calculated exactly, not just estimated). This cost is the selection criterion for uniform cost search ($g(n)$). Thus, the choice can be based on a fusion of the two criteria. That is, the summation of h and g is considered as performance function: $f(n)=g(n)+h(n)$. Such a strategy (similar to the branch and bound technique), of minimizing the total path cost defines the *A* search*. $f(n)$ represents the estimated cost on the cheapest path from the start node to the goal, and it incorporates $g(n)$ as an exact measure of

the path from the start node to the current node (as for *uniform cost search*), and *h(n)* as the estimation of the remainder path to the goal (as for *greedy search*). By finding a restriction that never overestimates the cost to reach the goal for *h*, the method is both *complete and optimal* (Russell & Norvig, 1995). Such a restriction is an *admissible heuristic*, which is optimistic by nature, by always underestimating the cost of solving the problem. Since *h* is admissible, the effect transfers to *f* as well (since *f*=*g*+*h*), and it underestimates the actual cost as well. *A* search* is a best first search using *f* as the evaluation function and an admissible *h* function.

2.2 Enhanced search strategies

Iterative improvement techniques are efficient practical approaches for boosting search strategies. They can be divided into two classes. The *hill-climbing strategy* makes changes to improve the current state. The algorithm does not maintain a search tree. Rather, it moves in the direction of increasing value within a loop. Although simple by nature, and efficient in practice, it suffers the drawback of becoming trapped in local optima. *Simulated annealing* represents the other class of iterative improvement strategies. It simply allows escaping a local optimum, by taking some steps to break out. It is an effective strategy for a good approximation of the global optimum in a large search space.

In *combinatorial search,* the goal is to find the best possible solution out of the feasible ones. There are two main approaches here. In *lazy evaluation* the computation is delayed until it is really needed, in contrast to *look-ahead* where, before making a decision, a few input steps are evaluated, in order to avoid backtracking at later stages. Both methods try to save both time and space in their evaluation.

Another distinctive technique is employed by *genetic algorithms*, which are essentially stochastic search methods, inspired from the principles of natural selection in biology. They employ a population of competing solutions—evolved over time—to converge to an optimal solution. Effectively, the solution space is searched in parallel, which helps in avoiding local optima, and provides straightforward parallelization possibilities. The search is an iterative process where each successive generation undergoes selection in the presence of variation-inducing operators such as mutation and recombination (crossover). A fitness function is used to evaluate individuals, and reproductive success varies with fitness.

Straightforward parallelization and the possibility of applying them in ill-defined problems make genetic algorithms attractive.

2.3 Data mining

Traditionally, *data mining* refers to the activity of extracting new, meaningful and potentially useful information from data. The term has recently expanded to the entire knowledge discovery process, encompassing several pre-/post- and processing steps. The learning step is central in any data mining process. It consists of presenting a dataset – the *training set* – to a learning algorithm, so that it learns the model "hidden" in the data. A dataset consists of a set of *instances*, each instance having a set of *predictive attributes* and a *target attribute*, the class. The aim is to predict the value of the class using the values of the predictive attributes and the model learned by the induction algorithm. In order to assess the generalization ability of the learned model (i.e. its quality), usually a *test set* is employed, consisting of instances that have not been "seen" by the model during the learning phase. Such a problem is known as a classification problem (if the class attribute is discrete), or a regression

problem (if the class is continuous). Another data mining task is clustering, which identifies similar characteristics and groups cases with similar characteristics together. In this case the class attribute is not present.

2.4 Decision trees

One of the most prominent techniques used for classification (and regression) problems in data mining are *decision trees*. They are tree structures, where each interior node corresponds to a decision attribute; an arc from a node to a child represents a possible value of that attribute. A leaf represents the value of the class attribute, given the values of the attributes present on the path from the root to that leaf. Decision tree algorithms apply a greedy search heuristic and construct the model in a top-down, recursive manner ("divide and conquer"). At every step, the algorithm considers the partition of the training set with respect to the "best" attribute (which becomes the decision attribute for that node). The selection of the "best" attribute is made according to some splitting measure. After an appropriate split has been selected, the training set is divided among the branches going out of that node into smaller subsets. The process continues until no split is considered good enough or a stopping criterion is satisfied.

The decision on which attribute to choose at a given step is based on measures provided by the information theory, namely on the entropy. It measures the uncertainty associated with a random variable. The most common attribute selection criterion is the expected reduction in entropy due to splitting on that attribute – the information gain.

While being rather simple and easy to understand, decision trees are also very robust with respect to the data quantity. Also, they require little data preparation, being able to handle both numerical and categorical data, as well as missing data. Furthermore, it is possible to validate the model using statistical tests, such as to determine its reliability.

3. Cost-sensitive learning

Traditionally, learning techniques are concerned with error minimization, i.e. reducing the number of misclassifications. However, in many real-world problems, such as fraud detection, loan assessment, oil-slick detection or medical diagnosis, the gravity of different types of classification errors is highly unbalanced.

For example, in credit assessment, given a customer loan application, the goal is to predict whether the bank should approve the loan, or not. In this situation, false positives are much more dangerous than false negatives. This means that an incorrect prediction that the credit should be approved, when the debtor is not actually capable of sustaining it, is far more damaging than the reverse situation. Another domain where different errors bear different significance and consequences is medical diagnosis (classifying an ill patient as healthy is by far riskier than the reverse situation).

In domains like these, the measure of total *cost* is introduced to determine the performance of learning algorithms. Total cost minimization is at least as important as minimizing the number of misclassification errors. This strategy is employed by cost-sensitive learning, a category of learning schemes which consider different approaches to achieve minimal costs.

As presented in (Turney, 2000), there are several types of costs involved in inductive concept learning, the most important being the *misclassification costs* and the *test costs*. These are also the focus of most cost-sensitive algorithms. Misclassification costs try to capture the

unbalance in different misclassifications. They are modeled through the use of a cost matrix $(C_{ij})_{nxn}$, where C_{ij} is the cost of misclassifying an instance of class j as being of class i. Test costs quantify the "price" of an attribute, without being restricted to its economical value. For example, in the medical domain, a test cost could represent a combination between the costs of the equipments involved in the investigation, the time spent to gather the results, the impact on the patient (psychical or physical - pain), a.s.o. Test costs are specified as attribute – value pairs.
In most real-world problems, setting the true costs is a difficult issue. If, in the case of test costs, the decision is made easier by the possibility of considering the different dimensions (time, monetary, pain, emotional implications, e.t.c.), when it comes to determining the misclassification costs we come across a more serious issue: we have to put a price on human life. Perhaps an appropriate approach here would be to experiment with several close proportions for the errors' unbalance.

3.1 Cost-sensitive algorithms

Most cost-sensitive classifiers focus on minimizing the misclassification costs. There exist, however, several algorithms which tackle test costs. Significantly less work has been done in aggregating the two cost components. This section reviews some of the most prominent cost-sensitive approaches in literature: stratification, MetaCost (Domingos, 1999) and AdaCost (Fan et. al., 2000) as misclassification cost-sensitive approaches, and Eg2 (Nunez, 1988), IDX (Norton, 1989) and CS-ID3 (Tan & Schlimmer, 1989, 1990) which consider test costs.

3.1.1 Stratification

Stratification is one of the earliest and simplest techniques for minimizing misclassification costs. It is a sampling procedure, which modifies the distribution of instances in the training set, such that the classes with a higher misclassification cost are better represented. Stratification can be achieved either through undersampling, or oversampling. While being a very simple and intuitive technique for considering the unbalance of different types of errors, the modification of the set distribution induces drawbacks, since it may bias the learning process towards distorted models. Also, each alternative has its own drawbacks: undersampling reduces the data available for learning, while oversampling increases the training time. However, the most serious limitation of this method comes from the fact that it restricts the dimension or the form of the cost matrix. For problems with more than two classes, or when the cost is dependent on the predicted class ($C_{ij} \neq C_{kj}$, where $k \neq i$), the cost matrix may become too complicated, such that proportions for each class cannot be established (Domingos, 1999).

3.1.2 MetaCost and AdaCost

More complex approaches usually involve meta-learning, and can be applied to a variety of base classifiers. The most representative in this category are MetaCost (Domingos, 1999) and AdaCost (Fan et. al., 2000).
MetaCost, introduced by Pedro Domingos, is a method for converting error-based classifiers into cost-sensitive approaches. It employs the *Bayes minimal conditional risk* principle to perform class re-labeling on the training instances. In order to determine the Bayes optimal prediction for each training example, i.e. the class which minimizes the conditional risk, an ensemble of classifiers is initially trained and employed to estimate the class probability for

each instance. After that, the risk for each class is computed, using the cost matrix settings. Each instance is then re-labeled with the class having the lowest risk. After obtaining the modified dataset, any error-based classifier will also minimize the cost while seeking to minimize zero-one loss (the error).
AdaCost is the misclassification cost-sensitive variant of the AdaBoost.M1 algorithm. Being a boosting-based approach, AdaCost employs an ensemble method, which builds a new model at each phase. Weights are assigned to each instance, and they are modified after each boosting phase, using the cost of misclassifications in the weight-update mechanism. Initially, high weights are assigned to costly instances, as opposed to AdaBoost.M1, where uniform weights are assigned to the training instances for the first boosting phase. In the empirical evaluations performed, AdaCost yielded a consistent and significant reduction in misclassification costs over AdaBoost.M1.

3.1.3 Eg2, CS-ID3 and IDX

The category of algorithms which focus on minimizing test costs is largely based on decision trees. Eg2, CS-ID3 or IDX are basically decision trees which employ a modified attribute selection criterion such as to embed the cost of the attribute in the selection decision. *Eg2*'s criterion is detailed in the section regarding the algorithm ICET.
IDX (Norton, 1989) uses a look-ahead strategy, by looking n tests ahead, where n is a parameter that may be set by the user. Its attribute selection criterion is:

$$\frac{\Delta I_i}{C_i} \tag{1}$$

where ΔI_i represents the information gain of attribute i, and C_i is its cost.
CS-ID3 (Tan & Schlimmer, 1989, 1990) uses a lazy evaluation strategy, by only constructing the part of the decision tree that classifies the current case. Its attribute selection heuristic is:

$$\frac{(\Delta I)^2}{C_i} \tag{2}$$

4. ICET – Inexpensive Classification with Expensive Tests

Introduced by Peter D. Turney as a solution to cost-sensitive problems, *ICET* (*Inexpensive Classification with Expensive Tests*) is a hybrid technique, which combines a *greedy search heuristic* (decision tree) with a *genetic algorithm*. Its distinctive feature is that it considers both test and misclassification costs, as opposed to the other cost-sensitive algorithms, which fail to consider both types of costs. Since it models real-world settings, where both the attributes and the different classification errors bear separate prices, the approach is more successful in true-life.
The technique combines two different components, on two levels:

- On the bottom level, a test cost-sensitive decision tree performs a greedy search in the space of decision trees
- On the top level, the evolutionary component performs a genetic search through a space of biases; these are used to control the preference for certain types of decision trees in the bottom layer

The components used in the initial version of ICET are: Eg2 (Nunez, 1988) for the decision tree component and GENESIS (Grefenstette, 1986) for the genetic component. *Eg2* has been implemented as a modified component of Quinlan's C4.5 (Quinlan, 1993), using ICF (Information Cost Function) as attribute selection function. For the ith attribute, ICF may be defined as follows:

$$ICF_i = \frac{2^{\Delta I_i} - 1}{(C_i + 1)^w}, \text{ where } 0 \leq w \leq 1 \quad (3)$$

This means that the attribute selection criterion is no longer based solely on the attribute's contribution to obtaining a pure split, but also on its cost, C_i. Also, the Information Cost Function contains parameter w, which adjusts the strength of the bias towards lower cost attributes. Thus, when $w = 0$, the cost of the attribute is ignored, and selection by ICF is equivalent to selection by the information gain function. On the other hand, when $w = 1$, ICF is strongly biased by the cost component.

The algorithm flow: the algorithm starts by the genetic component evolving a population of randomly generated individuals (an individual corresponds to a decision tree). Each individual in the initial population is then evaluated by measuring its fitness. Standard mutation and crossover operators are applied to the trees population and, after a fixed number of iterations, the fittest individual is returned (Fig. 1).

Each individual is represented as a bit string of $n + 2$ numbers, encoded in Gray. The first n numbers represent the bias parameters ("alleged" test costs in the ICF function). The last two stand for the algorithm's parameters *CF* and w; the first controls the level of pruning (as defined for C4.5), while w is needed by ICF.

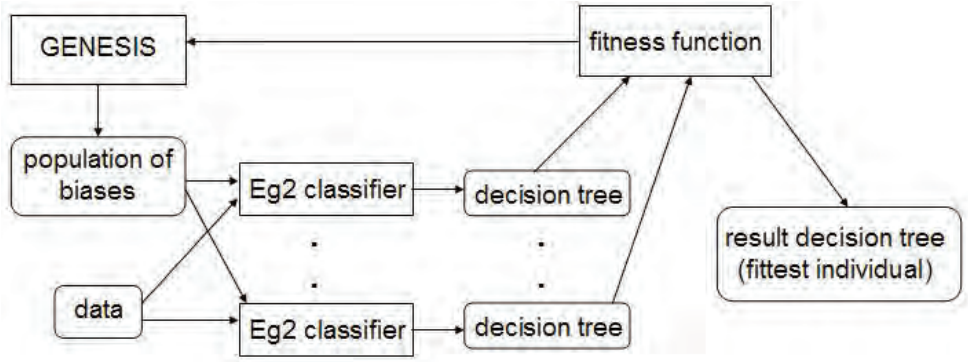


Fig. 1. The ICET technique

An important remark is that, unlike Eg2, ICET does not minimize test costs directly. Instead, it uses ICF for the codification of the individuals in the population. The n costs, C_i, are not true costs, but *bias parameters*. They provide enough variation to prevent the decision tree learner from getting trapped in a local optimum, by overrating/underrating the cost of certain tests based on past trials' performance. However, it is possible to use true costs, when generating the initial population, which has been shown to lead to some increase in performance.

Each trial on an individual consists in training and evaluating a decision tree on a given dataset, using the biases in the individual to set the attribute costs, *CF* and *w*. This is done by splitting the available dataset into two subsets: sub-training and sub-testing dataset. Since the split is random, there may be that two identical individuals will yield different outcomes (since the form of a decision tree is strongly related to the distribution in the training set – different training sets produce different trees).
In ICET, the *fitness function* for an individual is computed as the *average cost* of classification of the corresponding tree (obtained by randomly dividing the training set in two subsets, the first used for the actual tree induction and the second for error estimation). The average cost of classification is obtained by normalizing the total costs (obtained by summing the test and misclassification costs) to the test set size. Test costs are specified as attribute - cost value pairs. The classification costs are defined by a cost matrix $(C_{ij})_{nxn}$, where Cij - the cost of misclassifying an instance of class j as being of class i. If the same attribute is tested twice along the path (numeric attribute), the second time its cost is 0.
The particularity presented by ICET, of allowing the test costs (encoded inside a genetic individual) to vary freely in the search domain, and then applying the fitness evaluation to guide the individuals towards an optimal solution, increases the variability in the heuristic component. Moreover, *w* and *CF* – two key features which influence the future form of a decision tree – are also encoded in the individual, providing even more possibility of variation in the decision trees search space. Theoretically, this variability is desirable, especially for greedy algorithms such as decision tree learners – that yield unique structures for a fixed training set.

4.1 ProICET – improving the basic algorithm

Although ICET has a strong theoretical background, some enhancements can be considered, in order to boost its performance in real-world settings. Most of the changes affect the genetic component, but the training process is slightly different as well. This section also presents other implementation details, and briefly reviews the two tools employed for the current implementation.

4.1.1 Enhancements

First, and most importantly, the *single population technique* is employed as replacement strategy (instead of the multiple populations). In this technique the population is sorted according to the fitness of its elements. At each step two individuals are generated and their fitness is evaluated. According to their score, they are added to the same population their parent elements came from. Then, the individuals with the lowest fitness values are eliminated, so that the size of the population remains the same.
The single population technique has the advantage of directly implementing *elitism*: the best individuals of the current generation can survive unchanged in the next generation. Another prominent feature is the use of *ranking* in the fitness function estimation. The individuals in the population are ordered according to their fitness value, after which probabilities of selection are distributed evenly, according to their rank in the ordered population. Ranking is a very effective mechanism for avoiding the premature convergence of the population, which can occur if the initial pool has some individuals which dominate, having a significantly better fitness than the others.

Some amendments have been considered in the training process as well. Thus, the percentage of the training examples used when evaluating the fitness score of an individual in the population is now 70% of the original training set, as opposed to 50% (in the initial implementation).
The number of evaluation steps has also been increased. Due to the fact that a new generation is evolved using single population, the final result yielded by the procedure is the best individual over the entire run, which makes the decision on when to stop the evolution less critical. More than that, experiments show that usually the best individual does not change significantly after 800 steps: in more than 90% of the cases the algorithm converges before the 800th iteration, while in the rest of the cases the variations after this point are small (less than 3.5%). Therefore, the number of steps in our implementation is 1000.

4.1.2 Implementation overview

The current implementation of the improved ICET technique (*ProICET*) has been done starting from an existing implementation of revision 8 of the C4.5 algorithm, present in Weka (Witten, 2005) and a general genetic tool, GGAT (GGAT, 2002), developed at Brunel University.
Weka (*Waikato Environment for Knowledge Analysis*) is a data mining tool developed at the University of Waikato, New Zealand. It is distributed under the GPL (Gnu Public License) and it includes a wide variety of state-of-the-art algorithms and data processing tools, providing extensive support for the entire process of experimental data mining (input filtering, statistical evaluation of learning schemes, data visualization, preprocessing tools). The command-line interface it provides was particularly useful when invoking the modified decision tree learner for computing the fitness function in the genetic algorithm part of the application.

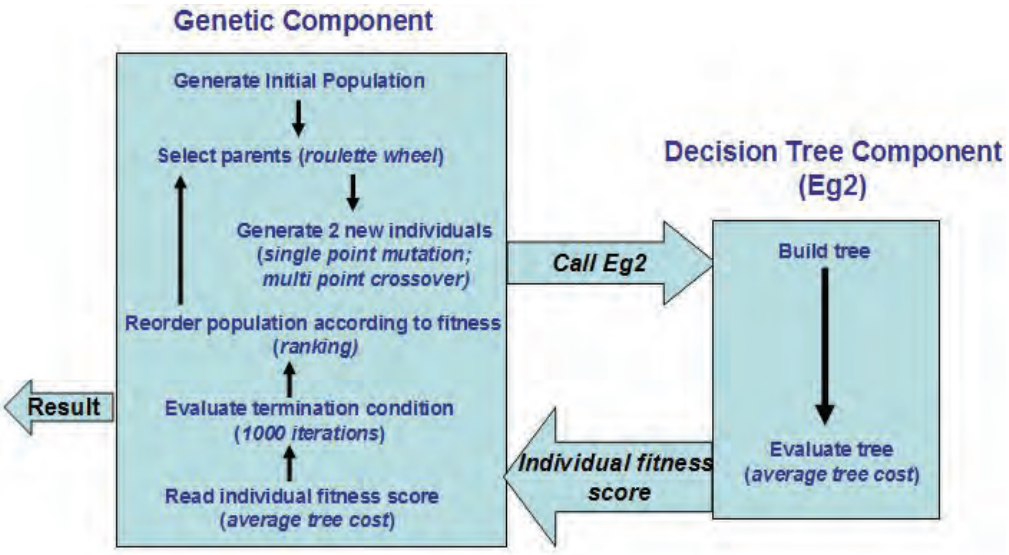


Fig. 2. ProICET main flow

GGAT is a generic GA library, developed at the Brunel University, London. It implements most genetic algorithm mechanisms. Of particular interest are the single population technique and the ranking mechanisms.
In order to obtain the Eg2 attribute selection criterion, as presented in equation (3), the information gain function of J4.8 algorithm was modified, similarly to the implementation presented in (Turney, 1995).
ProICET has been implemented within the framework provided by GGAT. For each individual, the $n + 2$ chromosomes are defined (n being the number of attributes in the data set, while the other two correspond to parameters w and CF); each chromosome is represented as a 14 bits binary string, encoded in Gray. The population size is 50 individuals. The *roulette wheel* technique is used for parent selection; as recombination techniques, we have employed *single point random mutation* with mutation rate 0.2, and *multipoint crossover*, with 4 randomly selected crossover points.
Since the technique involves a large heuristic component, the evaluation procedure assumes averaging the costs over 10 runs. Each run uses a pair of randomly generated training-testing sets, in the proportion 70% - 30%; the same proportion is used when separating the training set into a component used for training and one for evaluating each individual (in the fitness function).

5. Experimental work

A significant problem related to the original ICET technique is rooted in the fact that costs are learned indirectly, through the fitness function. Rare examples are relatively more difficult to be learned by the algorithm. This fact was also observed in (Turney, 1995), where, when analyzing complex cost matrices for a two-class problem, it is noted that: *it is easier to avoid false positive diagnosis [...] than it is to avoid false negative diagnosis [...]. This is unfortunate, since false negative diagnosis usually carry a heavier penalty, in real life.*
Turney, too, attributes this phenomenon to the distribution of positive and negative examples in the training set. In this context, our aim is to modify the fitness measure as to eliminate such undesirable asymmetries.
Last, but not least, previous ICET papers focus almost entirely on test costs and lack a comprehensive analysis of the misclassification costs component. Therefore we attempt to fill this gap by providing a comparative analysis with some of the classic cost-sensitive techniques, such as MetaCost and Eg2, and prominent error-reduction based classifiers, such as J4.8 and AdaBoost.M1.

5.1 Symmetry through stratification

As we have mentioned before, it is believed that the asymmetry in the evaluated costs for two-class problems, as the proportion of false positives and false negatives misclassification costs varies, is owed to the small number of negative examples in most datasets. If the assumption is true, the problem could be eliminated by altering the distribution of the training set, either by oversampling, or by undersampling. This hypothesis was tested by performing an evaluation of the ProICET results on the Wisconsin breast cancer dataset. This particular problem was selected as being one of the largest two-class datasets presented in the literature.

For the stratified dataset, the negative class is increased to the size of the positive class, by repeating examples in the initial set, selected at random, with a uniform distribution. Oversampling is preferred, despite of an increase in computation time, due to the fact that the alternate solution involves some information loss. Undersampling could be selected in the case of extremely large databases, for practical reasons. In that situation, oversampling is no longer feasible, as the time required for the learning phase on the extended training set becomes prohibitive.
The misclassification cost matrix used for this analysis has the form:

$$C = 100 \cdot \begin{pmatrix} 0 & p \\ 1-p & 0 \end{pmatrix}, \tag{4}$$

where p is varied with a 0.05 increment.
The results of the experiment are presented in Fig. 3. We observe a small decrease in misclassification costs for the stratified case throughout the parameter space. This reduction is visible especially at the margins, when costs become more unbalanced. Particularly in the left side, we notice a significant reduction in the total cost for expensive rare examples, which was the actual goal of the procedure.

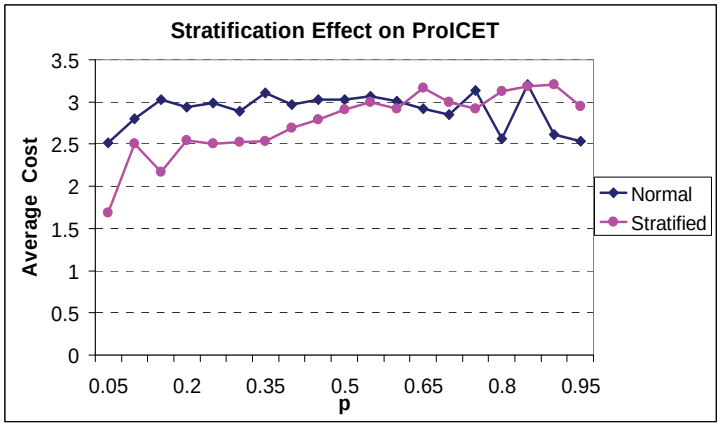


Fig. 3. ProICET average costs for the breast cancer dataset

Starting from the assumption that the stratification technique may be applicable to other cost-sensitive classifiers, we have repeated the procedure on the Weka implementation of MetaCost, using J4.8 as base classifier. J4.8 was also considered in the analysis, as baseline estimate.
The results for the second set of tests are presented in Fig. 4. We observe that MetaCost yields significant costs, as the cost matrix drifts from the balanced case, a characteristic which has been described previously. Another important observation is related to the fact that the cost characteristic in the case of J4.8 is almost horizontal. This could give an explanation of the way stratification affects the general ProICET behavior, by making it insensitive to the particular form of the cost matrix. Most importantly, we notice a general reduction in the average costs, especially at the margins of the domain considered. We

conclude that our stratification technique could be also used for improving the cost characteristic of MetaCost.

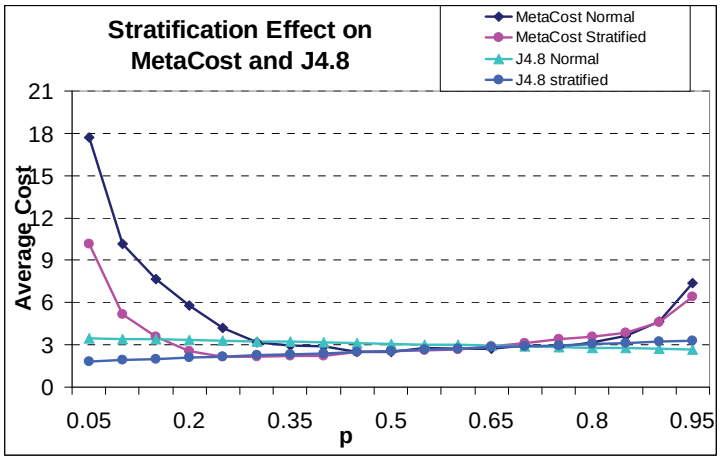


Fig. 4. Improved average cost for the stratified Wisconsin dataset

5.2 Comparing misclassification costs

The procedure employed when comparing misclassification costs is similar to that described in the previous section. Again, the Wisconsin dataset was used, and misclassification costs were averaged on 10 randomly generated training/test sets. For all the tests described in this section, the test costs are not considered in the evaluation, in order to isolate the misclassification component and eliminate any bias.

As illustrated by Fig. 5, MetaCost yields the poorest results. ProICET performs slightly better than J4.8, while the smallest costs are obtained for AdaBoost, using J4.8 as base classifier. The improved performance is related to the different approaches taken when searching for the solution. If ProICET uses heuristic search, AdaBoost implements a procedure that is guaranteed to converge to minimum training error, while the ensemble voting reduces the risk of overfitting. However, the approach cannot take into account test costs, which should make it perform worse on problems involving both types of costs.

5.3 Total cost analysis

When estimating the performance of the various algorithms presented, we have considered four problems from the UCI repository. All datasets involve medical problems: Bupa liver disorders, thyroid, Pima Indian diabetes and heart disease Cleveland. For the Bupa dataset, we have used the same modified set as in (Turney, 1995). Also, the test costs estimates are taken from the previously mentioned study. As mentioned before, the misclassification costs values are more difficult to estimate, due to the fact that they measure the risks of misdiagnosis, which do not have a clear monetary equivalent. These values are set empirically, assigning higher penalty for undiagnosed disease and keeping the order of magnitude as to balance the two cost components (the actual values are displayed in tables 1, 2, 3 and 4).

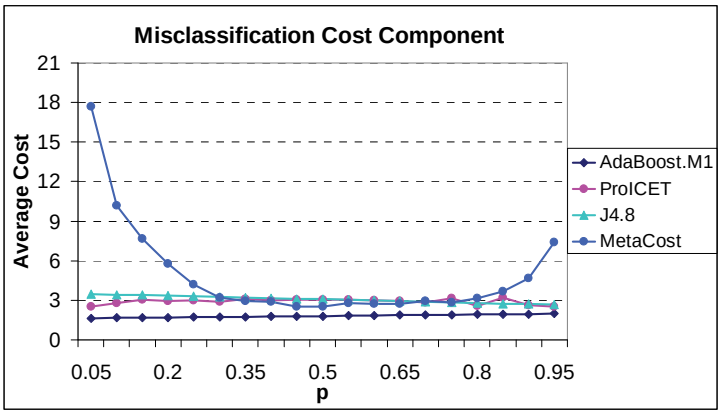


Fig. 5. A comparison of average misclassification costs on the Wisconsin dataset

Class	less than 3	more than
less than 3	0	5
more than 3	15	0

Table 1. Misclassification cost matrix for Bupa liver disorder dataset

Class	3	2	1
3	0	5	7
2	12	0	5
1	20	12	0

Table 2. Misclassification cost matrix for the Thyroid dataset

Class	less than 3	more than
less than 3	0	7
more than 3	20	0

Table 3. Misclassification cost matrix for the Pima dataset

Class	0	1	2	3	4
0	0	10	20	30	40
1	50	0	10	20	30
2	100	50	0	10	20
3	150	100	50	0	10
4	200	150	100	50	0

Table 4. Misclassification cost matrix for the Cleveland heart disease dataset

As anticipated, ProICET significantly outperforms all other algorithms, being the only one built for optimizing total costs (Fig. 6-9). ProICET performs quite well on the heart disease dataset (Fig. 6), where the initial implementation obtained poorer results. This improvement is probably owed to the alterations made to the genetic algorithm, which increase the population variability and extend the ProICET heuristic search.

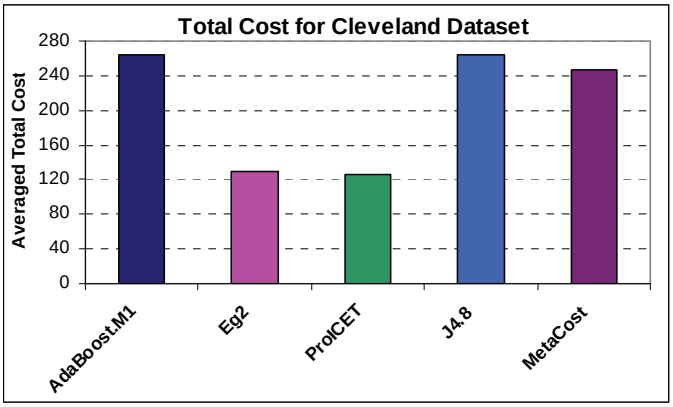


Fig. 6. Average total costs of the considered algorithms on the Cleveland dataset

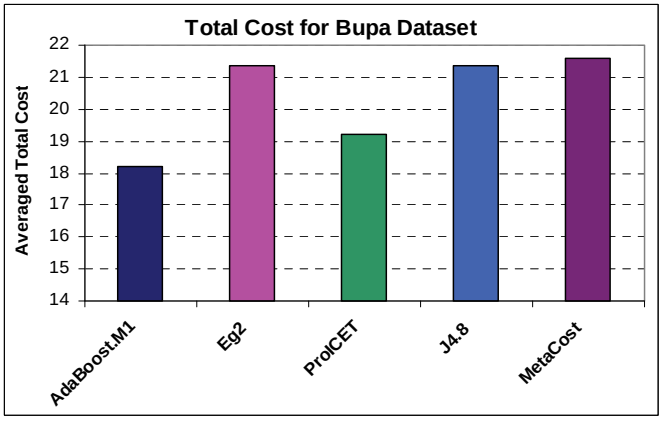


Fig. 7. Average total costs of the considered algorithms on the Bupa dataset

On the Bupa dataset (Fig. 7), AdaBoost.M1 slightly outperforms ProICET, but this is more an exception, since on the other datasets, AdaBoost.M1 yields poorer results. Moreover, the cost reduction performed by ProICET relative to the other methods, on this dataset, is very significant.

The cost reduction is relatively small in the Thyroid dataset (Fig. 8), compared to the others, but is quite large for the other cases, supporting the conclusion that ProICET is the best approach for problems involving complex costs.

6. Chapter summary

This chapter presents the successful combination of two search strategies, greedy search (in the form of decision trees) and genetic search, into a hybrid approach. The aim is to achieve increased performance over existing classification algorithms in complex cost problems, usually encountered when mining real-world data, such as in medical diagnosis or credit assessment.

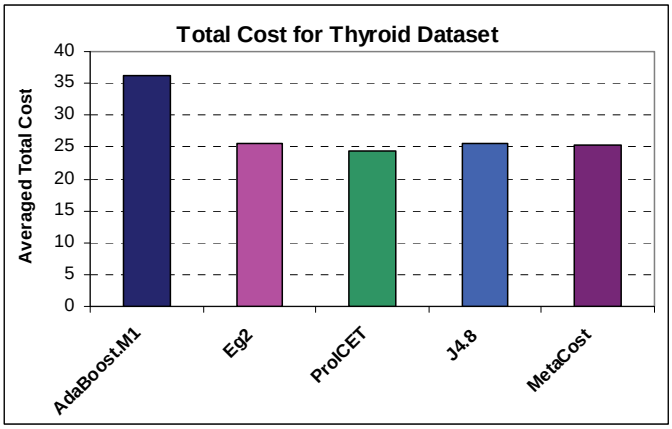


Fig. 8. Average total costs of the considered algorithms on the Thyroid dataset

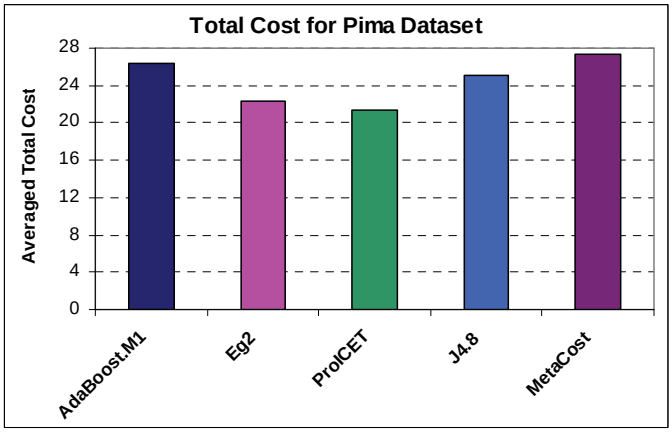


Fig. 9. Average total costs of the considered algorithms on the Pima dataset

Any machine learning algorithm is based on a certain search strategy, which imposes a bias on the technique. There are many search methods available, each with advantages and disadvantages. The distinctive features of each search strategy restrict its applicability to certain problem domains, depending on which issues (dimensionality, speed, optimality, etc.) are of importance. The dimension of the search space in most real-world problems renders the application of complete search methods prohibitive. Sometimes we have to trade optimality for speed. Fortunately, greedy search strategies, although do not ensure optimality, usually provide a sufficiently good solution, close to the optimal one. Although they have an exponential complexity in theory, since they do not explore the entire search space, they have a very good behaviour in practice, in speed terms. This makes them suitable even for complex problems. Their major drawback comes from the fact that they can get caught at local optima. Since the complexity of the search space is too large, such that the problem is intractable for other techniques, in most real problems this is an accepted disadvantage. Greedy search strategies are employed in many machine learning algorithms.

One of the most prominent classification techniques which employ such a strategy are decision trees.

The main advantages of decision trees are: an easy to understand output model, robustness with respect to the data quantity, little data preparation, ability to handle both numerical and categorical data, as well as missing data. Therefore, decision trees have become one of the most widely employed classification techniques in data mining, for problems where error minimization is the target of the learning process.

However, many real-world problems require more complex measures for evaluating the quality of the learned model. This is due to the unbalance between different types of classification errors, or the effort of acquiring the values of predictive attributes. A special category of machine learning algorithms focuses on this task – cost-sensitive learning. Most existing techniques in this class focus on just one type of cost, either the misclassification, or the test cost. Stratification is perhaps the earliest misclassification cost-sensitive approach (a sampling technique rather than an algorithm). It has been followed by developments in the direction of altering decision trees, such as to make their attribute selection criterion sensitive to test costs (in the early 90's). Later, new misclassification cost-sensitive approaches emerged, the best known being MetaCost or AdaCost. More recent techniques consider both types of cost, the most prominent being ICET.

Initially introduced by Peter D. Turney, ICET is a cost-sensitive technique, which avoids the pitfalls of simple greedy induction (employed by decision trees) through evolutionary mechanisms (genetic algorithms). Starting from its strong theoretical basis, we have enhanced the basic technique in a new system, ProICET. The alterations made in the genetic component have proven beneficial, since ProICET performs better than other cost-sensitive algorithms, even on problems for which the initial implementation yielded poorer results.

7. References

Baezas-Yates, R., Poblete, V. P. (1999). Searching, In: *Algorithms and Theory of Computation Handbook*, Edited by Mikhail J. Atallah, Purdue University, CRC Press

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining,* pp. 155-164, 1-58113-143-7, San Diego, CA, USA

Fan, W.; Stolfo, S.; Zhang, J. & Chan, P. (2000). AdaCost: Misclassification cost-sensitive boosting. *Proceedings of the 16th International Conference on Machine Learning*, pp. 97–105, Morgan Kaufmann, San Francisco, CA

Freund, Y. & Schapire, R. (1997). A decision-theoretic generalization of on- line learning and an application to boosting. *Journal of Computer and System Sciences*, Volume 55, Number 1, August 1997 , pp. 119-139

General Genetic Algorithm Tool (2002), GGAT, http://www.karnig.co.uk/ga/content.html, last accessed on July 2008

Grefenstette, J.J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16, 122-128

Korf, R. E. (1999). Artificial Intelligence Search Algorithms, In: *Algorithms and Theory of Computation Handbook*, Edited by Mikhail J. Atallah, Purdue University, CRC Press

Norton, S.W. (1989). Generating better decision trees. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, IJCAI-89,* pp. 800-805. Detroit, Michigan.

Núñez, M. (1988). Economic induction: A case study. *Proceedings of the Third European Working Session on Learning, EWSL-88*, pp. 139-145, California, Morgan Kaufmann.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, ISBN:1-55860-238-0, San Francisco, CA, USA

Quinlan, J. (1996) Boosting first-order learning. *Proceedings of the 7th International Workshop on Algorithmic Learning Theory*, 1160:143–155

Russell, S., Norvig, P. (1995) *Artificial Intelligence: A Modern Approach*, Prentice Hall

Tan, M., & Schlimmer, J. (1989). Cost-sensitive concept learning of sensor use in approach and recognition. *Proceedings of the Sixth International Workshop on Machine Learning, ML-89,* pp. 392-395. Ithaca, New York

Tan, M., & Schlimmer, J. (1990). CSL: A cost-sensitive learning system for sensing and grasping objects. *IEEE International Conference on Robotics and Automation*. Cincinnati,Ohio

Turney, P. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, Volume 2, pp. 369–409

Turney, P. (2000). Types of cost in inductive concept learning. *Proceedings of the Workshop on Cost-Sensitive Learning, 7th International Conference on Machine Learning,* pp. 15-21

Vidrighin, B. C., Savin, C. & Potolea, R. (2007). A Hybrid Algorithm for Medical Diagnosis. *Proceedings of Region 8 EUROCON 2007*, Warsaw, pp. 668-673

Vidrighin, C., Potolea, R., Giurgiu, I. & Cuibus, M. (2007). ProICET: Case Study on Prostate Cancer Data. *Proceedings of the 12th International Symposium of Health Information Management Research,* 18-20 July 2007, Sheffield, pp. 237-244

Witten I. & Frank, E. (2005) *Data Mining: Practical machine learning tools and techniques,* 2nd *ed.* Morgan Kaufmann, 0-12-088407-0

10

Greedy Algorithm: Exploring Potential of Link Adaptation Technique in Wideband Wireless Communication Systems

Mingyu Zhou, Lihua Li, Yi Wang and Ping Zhang
Beijing University of Posts and Telecommunications, Beijing China

1. Introduction

As the development of multimedia communication and instantaneous high data rate communication, great challenge appears for reliable and effective transmission, especially in wireless communication systems. Due to the fact that the frequency resources are decreasing, frequency efficiency obtains the most attention in the area, which motivates the research on link adaptation technique. Link adaptation can adjusts the transmission parameters according to the changing environments [1-2]. The adjustable link parameters includes the transmit power, the modulation style, etc. All these parameters are adjusted to achieve:

1. Satisfactory Quality of Service (QoS). This helps guarantee the reliable transmission. It requires that the bit error rate (BER) should be lower than a target.
2. Extra high frequency efficiency. This brings high data rate. It can be described with throughput (in bit/s/Hz).

In conventional systems with link adaptation, water-filling algorithm is adopted to obtain the average optimization for both QoS and frequency efficiency [3]. But the transmit power may vary a lot on different time, which brings high requirement for the implementation and causes such algorithm not applicable in practical systems.

Recently, wideband transmission with orthogonal frequency division multiplexing (OFDM) technique is being widely accepted, which divides the frequency band into small sub-carriers [4]. Hence, link adaptation for such system relates to adaptation in both time and frequency domain. The optimization problem becomes how to adjust the transmit power and modulation style for all sub-carriers, so as to achieve the maximum throughput, subject to the constraint of instantaneous transmit power and BER requirement. The transmit power and modulation style on every sub-carrier may impact the overall performance, which brings much complexity for the problem [5].

In order to provide a good solution, we resort to Greedy algorithm [6]. The main idea for the algorithm is to achieve global optimization with local optimization. It consists of many allocation courses (adjusting modulation style equals to bit allocation). In each course, the algorithm reasonably allocates the least power to support reliable transmission for one additional bit. Such allocation course is terminated when transmit power is allocated. After allocation, the power on each sub-carrier can match the modulation style to provide reliable

transmission, the total transmit power is not higher than the constraint, and the throughput can be maximized with reasonable allocation.
We investigate the performance of Greedy algorithm with aid of Matlab. With the simulation result, we can observe that:
1. The transmit power is constraint as required with the algorithm;
2. The algorithm can satisfy the BER requirement;
3. It brings great improvement for the throughput.

Hence we conclude that Greedy Algorithm can bring satisfactory QoS and high frequency efficiency. In order to interpret it in great detail, we will gradually exhibit the potential of Greedy algorithm for link adaptation. The chapter will conclude the following sections:
Section 1: As an introduction, this section describes the problem of link adaptation in wireless communication systems, especially in OFDM systems.
Section 2: As the basis of the following sections, Section 2 gives out the great detail for the theory of link adaptation technique, and presents the problem of the technique in OFDM systems.
Section 3: Greedy Algorithm is employed to solve the problem of Section 2 for normal OFDM systems. And the theory of Greedy Algorithm is provided in the section. We provide comprehensive simulation results in the section to prove the algorithm can well solve the problem.
Section 4: Greedy Algorithm is further applied in a multi-user OFDM system, so as to bring additional great fairness among the transmissions for all users. Simulation results are provided for analysis.
Section 5: OFDM relaying system is considered. And we adopt Greedy Algorithm to bring the optimal allocation for transmit power and bits in all nodes in the system, so as to solve the more complex problem for the multi-hop transmission. We also present the simulation result for the section.
Section 6: As a conclusion, we summarize the benefit from Greedy Algorithm to link adaptation in wireless communication systems. Significant research topics and future work are presented.

2. Link Adaptation (LA) in OFDM systems

In OFDM systems, system bandwidth is divided into many fractions, named sub-carriers. Information is transmitted simultaneously from all these sub-carriers, and because different sub-carriers occupy different frequency, the information can be recovered in the receiver. The block diagram for such systems is shown in Fig. 1.
As far as multiple streams on all these sub-carriers are concerned, the problem came out about how to allocate the transmit power and bits on all the sub-carriers, so as to bring highest throughput with constraint of QoS, or BER requirement. Due to the fact that there exists channel fading and that the impact with different sub-carrier varies because of the multi-path fading, the allocation should be different for different sub-carriers. The system can be described with the following equation.

$$R_n = H_n \sqrt{P_n} S_n + N_n \tag{1}$$

where S_n denotes the modulated signal on the n-th sub-carrier, which carries b_n bits with normalized power; P_n denotes the transmit power for the sub-carrier; H_n denotes the channel

fading for the sub-carrier; N_n denotes the additive white Gaussian noise (AWGN) with variance of σ^2; and R_n denotes the received signal on the sub-carrier.

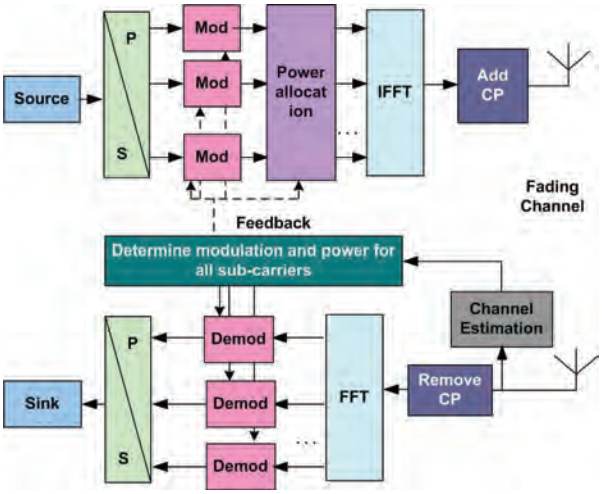


Fig. 1. Block diagram for OFDM with link adaptation

In that case, the received signal-to-noise ratio (SNR) can be calculated as

$$SNR_n = |H_n|^2 P_n / \sigma^2 \tag{2}$$

In order to satisfy the requirement of BER, the received signal should satisfy that SNR_n is larger than a certain threshold, or T_v for the v-th modulation. In the chapter, we assume that the target BER is 10^{-3}. Hence, the constraint of BER can be described as

$$SNR_n > T_v \tag{3}$$

As for the transmit power, it is required that the total transmit power should be constraint to a certain value P. That is to say

$$\frac{1}{N}\sum_{n=1}^{N} P_n \le P \tag{4}$$

As a conclusion, the optimization problem is how to determine b_n and P_n to maximize throughput, i.e.

$$\arg\max_{b_n, P_n} \sum_{n=1}^{N} b_n \tag{5}$$

subject to equations (3) and (4).

3. Application of greedy algorithm in OFDM systems

The Greedy algorithm can be applied in solving the problem of (5). For the research in the section, we assume the parameters for the candidate modulation as shown in Table 1, where the thresholds are obtained through simulation.

In order to obtain the maximum throughput across all these N sub-carriers, Greedy algorithm can be taken advantage of. The problem can be seen as a problem with global optimization, and Greedy algorithm can help achieve the global optimization with a lot of local optimization. The theory of Greedy algorithm can be understood from an example shown in Fig. 2.

v	Modulation	Number of bits $b(v)$	$T(v)$
0	No transmission	0	0
1	QPSK	2	9.78dB
2	16QAM	4	16.52dB
3	64QAM	6	22.52dB

Table 1. Candidate modulation and parameters

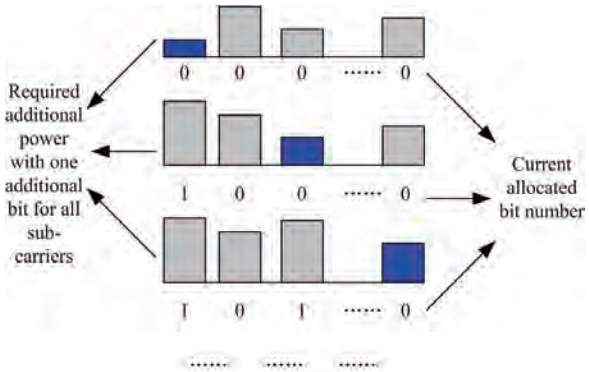


Fig. 2. Theory of the application of Greedy algorithm in OFDM systems

In the initialization step, all the sub-carrier is allocated with 0 bit. And the required additional power with one additional bit for all sub-carriers can be calculated. The local optimization is to allocate one bit to the sub-carrier with the least required power. Hence, as shown in Fig. 2, the 1st sub-carrier is allocated with 1 bit. And the required additional power with one additional bit for it is updated. In the second allocation, the 3rd sub-carrier obtains the least required additional power. Hence it is allocated with 1 bit. The processes continue until all sub-carriers are allocated with the maximum bits or the power is not enough to support one further bit. When the processes end, the global optimization is achieved and the current allocation is the optimal allocation with the maximum throughput. Due to the candidate modulations in Table 1, the incremental bit number is 2 in the research.

Fig. 3 shows the BER performance with Greedy algorithm, and the performance with fixed modulation of QPSK, 16QAM and 64QAM is shown for comparison, where *SNR* in the x-axis denotes the average SNR in the link. From the figure, the fixed modulation schemes have bad performance. Only when *SNR* is as high as 30dB can the BER achieve 10^{-3}. When Greedy algorithm is adopted, the BER performance can achieve the target BER with all *SNR* cases. Hence, it can be concluded that Greedy algorithm can satisfy the requirement of BER very well. Fig. 4 gives out the throughput performance with Greedy algorithm. As *SNR* rises larger, the throughput can achieve higher, with the maximum of 6 bits/symbol which denotes that all sub-carriers adopt 64QAM in this case. According to Greedy algorithm, the throughput is maximized.

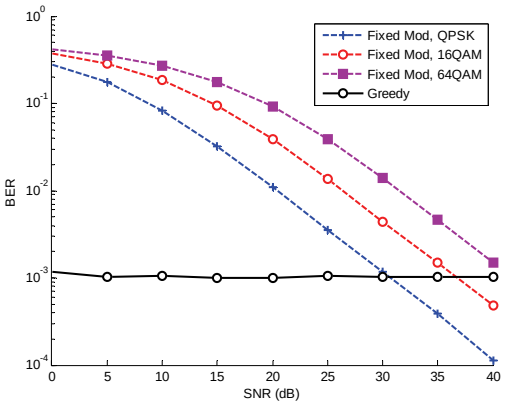


Fig. 3. BER performance with fixed modulation and Greedy algorithm

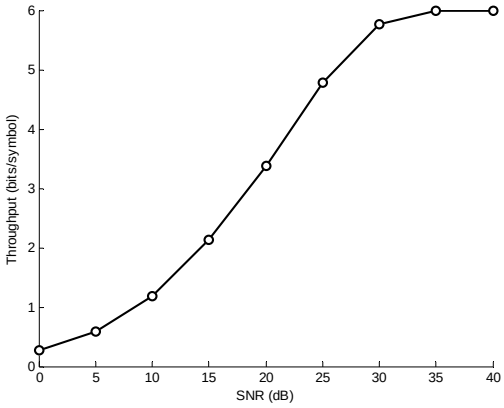


Fig. 4. Throughput performance with Greedy algorithm

4. Application of greedy algorithm in multi-user OFDM systems

When multiple users are concerned, the problem becomes more complex. The block diagram for a typical multi-user adaptive OFDM system is shown in Fig. 5. Downlink transmission is taken for research in the section. It is assumed that channel state information (CSI) regarding to all users is available for the base station (BS). In the transmitter on BS, resource allocation is carried out according to CSIs regarding to all the users, so as to determine the allocated sub-carriers for each user, as well as the transmit power for sub-carriers and loaded bits on them. Different loaded bits correspond to different modulation. All users' bits are modulated accordingly, after which inverse discrete Fourier transform (IDFT) is carried out and cyclic prefix (CP) is added to form OFDM symbols to transmit. In the receiver on each mobile terminal (MT), symbols in frequency domain are obtained after removing CP and DFT. Relevant demodulation is carried out for all sub-carriers, and source bits for each user are recovered finally after demodulation.

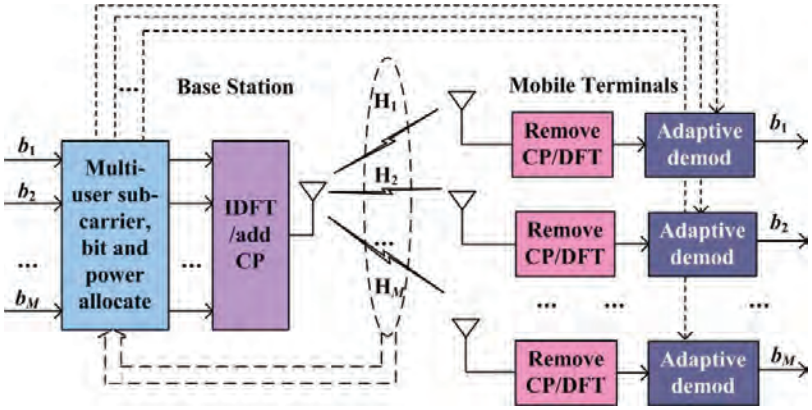


Fig. 5. A typical multi-user adaptive OFDM downlink

Considering a multi-user adaptive OFDM system with M users and N sub-carriers, the multi-user adaptive system model can be described as:

$$R_{m,n} = H_{m,n}\sqrt{P_{m,n}}S_{m,n} + N_{m,n} \tag{6}$$

where $S_{m,n}$ denotes power-normalized modulated symbol on the n-th sub-carrier for the m-th user; it contains $b_{m,n}$ source bits. In order to eliminate interference among the users, each sub-carrier can be allocated to only one user in the system, i.e. $if\ b_{m,n} > 0,\ then\ b_{m',n} = 0(\forall m' \neq m)$. $P_{m,n}$ denotes allocated power for $S_{m,n}$. $H_{m,n}$ denotes channel transfer function for $S_{m,n}$. $P_{m,n}$ and $b_{m,n}$ is determined according to $H_{m,n}$ by the "multi-user sub-carrier, bit and power allocation" block as shown in Fig. 5. $N_{m,n}$ denotes the additive white Gaussian noise (AWGN) with variance σ^2. $R_{m,n}$ is the received symbol in the receiver on the m-th MT.

4.1 Multi-user resource allocation problem

From (6), the received SNR can be calculated as

$$\gamma_{m,n} = |H_{m,n}|^2 P_{m,n} / \sigma^2 \tag{7}$$

As for conventional multi-user OFDM systems, each user is allocated with N/M sub-carriers with constant power fixedly. The throughput is very low since it is very likely that many users are allocated with sub-carriers with poor CSI. And the fairness performance is also poor. Adaptive resource allocation can take advantage of CSIs for all users to improve system performance through reasonable allocation of sub-carriers, bits and power. The multi-user adaptive problem for an OFDM system can be described as maximizing overall throughput while satisfying requirement of fairness, subject to power restriction and QoS requirement. Consequently, the problem can be described in the following way.

$$\max \sum_{n=1}^{N}\sum_{m=1}^{M} b(v_{m,n}) \tag{8}$$

subject to

$$\begin{aligned} &\max_{m\neq m'} \left| \sum_{n=1}^{N} [b(v_{m,n}) - b(v_{m',n})] \right| \leq b_0 && \text{(a)} \\ &P_{m,n} \geq T(v_{m,n})\sigma^2 / |H_{m,n}|^2 && \text{(b)} \\ &\sum_{m=1}^{M}\sum_{n=1}^{N} P_{m,n} \leq P_T && \text{(c)} \end{aligned} \tag{9}$$

where $v_{m,n}$ denotes the index of a candidate modulation ($v_{m,n}$ =1, 2, …, V); P_T denotes the total transmit power; $T(v_{m,n})$ indicates the least required SNR when adopting $v_{m,n}$-th modulation (one modulated symbol containing $b(v_{m,n})$ bits) to ensure QoS guaranteeing (BER lower than a certain value), i.e. Pm,n is determined to satisfy $\gamma_{m,n} \geq T(v_{m,n})$, so as to transmit $b(v_{m,n})$ bits with the target BER requirement; b_0 is the upper limit for maximum difference of allocated bits numbers among all users. Accordingly, (9a) is set to guarantee fairness requirement among all users, (9b) is used to satisfy requirement of QoS, and (9c) is to ensure the transmit power restriction.

In this section, 4 candidate modulations in Table 1 are considered. b_0 is set to be 0. And V is 4. BER performance versus SNR of received symbol can be calculated through

$$\varepsilon^b(\gamma) = 2\left(1 - 1/\sqrt{2^b}\right) Q\left(\sqrt{3/(2^b - 1)\gamma}\right) \tag{10}$$

where $Q(x) = 1/\sqrt{2\pi}\int_x^{\infty} e^{-u^2/2} du$, and b=2, 4, 6 correspond to QPSK, 16QAM and 64QAM, respectively. Parameters for the candidate modulations and least required SNR for BER lower than 10^{-3} are summarized in Table 1.

4.2 Multi-user adaptive resource allocation methods

The optimal joint problem of sub-carrier, bit and power allocation is a NP-hard combinatorial problem. It is quite difficult to determine how many and which sub-carriers should be assigned to every user subject to many restrictions as shown in (9). Some existing typical methods are introduced in [7-11]. They perform well in some aspects, but for services with tight fairness requirement, these solutions cannot provide nice performance.

A. Existing methods

According to [7-11], there have been many methods to allocate resources including sub-carriers, bits and power to multiple users. The fixed sub-carrier allocation method allocates the same number of sub-carriers to each user fixedly, and then adopts the optimal Greedy algorithm [6] to carry out bit-loading and power allocation on all sub-carriers [9]. This merit of the method is quite simple, and fairness can be guaranteed in a certain degree since each user is allocated with the same number of sub-carriers. But since it is very likely that many users are allocated with sub-carriers with poor CSI, the throughput is low; and because CSIs of different users vary much, fairness performance is also poor.

Typically, [11] provides another solution. In each allocation, a user is assigned with one sub-carrier with the best CSI from the remaining un-allocated sub-carriers. For allocations with odd indices, the order to allocate sub-carriers is from the 1st user to the M-th user. For

allocations with even indices, the order is from the M-th user to the 1st user. Allocation continues until all sub-carriers are assigned. The optimal Greedy algorithm is then adopted to accomplish bit-loading and power allocation. The ordered allocation method avoids that some user's sub-carriers have much better CSI than other users, and fairness can be improved much over the fixed method.

The fixed allocation method and ordered allocation method both allocate resources in two steps, and fairness is fine in some degree by allocation equal number of sub-carriers to each user. But because CSI of different users may vary greatly, allocating the same number of sub-carriers to all users may not provide enough fairness for some services with tight fairness requirement. Therefore, a multi-user sub-carrier, bit and power allocation method is introduced in the following, which can bring to much better fairness than the existing methods, while achieving high throughput and satisfying QoS requirement.

B. Proposed method

In the proposed method, sub-carriers are allocated equally to all users firstly as initialization. This step realizes coarse sub-carrier allocation. Second, bits and power are loaded on the sub-carriers following the optimal Greedy algorithm for all users. Coarse resource allocation is fulfilled in the two steps, and they can benefit much for the overall throughput. In order to improve fairness as required, the next step is added. In the 3rd step, sub-carriers, bits and power are adjusted among all users. This step can be seen as fine adjustment for the resources. The three steps can be described as follows.

1. Coarse sub-carrier allocation

This step is to allocate sub-carriers to all users with high throughput and coarse fairness among all users. The numbers of allocated sub-carriers are the same for all users. In every allocation, if the CSI relating the m-th user and the n-th sub-carrier is the best, the n-th sub-carrier will be allocated to the m-th user. The allocation for one user will be terminated when the user has been allocated with N/M sub-carriers. We make the assumptions: $\Theta=\Theta\backslash\{n\}$ denotes removing the n-th sub-carrier from the set Θ, and ϕ denotes the null set; $c(n)$ denotes the index of user who is allocated with the n-th sub-carrier; N_m denotes the number of allocated sub-carriers for the m-th user. The processes for this step are as follows.

$\Theta = \{1, 2, ..., N\}; \Psi = \{1, 2, ..., M\}$

$c(n) = 0, \forall n \in \Theta; N_m = 0, \forall m \in \Psi$

while $\Psi \neq \phi$

{ $[m, n] = \arg\max_{m \in \Psi, n \in \Theta} |H_{m,n}|$

$c(n) = m$

$\Theta = \Theta \backslash \{n\}$

$N_m = N_m + 1$

if $N_m = N / M$, **then** $\Psi = \Psi \backslash \{m\}$

}

2. Coarse bit loading and power allocation

The optimal Greedy algorithm is adopted in this step to realize bit loading and power allocation on all the sub-carriers. We assume that: v_n denotes the index of the modulation adopted on the n-th sub-carrier; P_0 denotes the remaining power; P_n indicates power allocated on the n-th sub-carrier, and W_n denotes the required additional power in order to adopt a next higher order modulation. Then the step can be described as follows.

$\Theta = \{1, 2, ..., N\}; \Psi = \{1, 2, ..., M\}$

$\underline{H}_n = H_{c(n),n}, \forall n \in \Theta$

$v_m = 0, \forall m \in \Psi; P_n = 0, \forall n \in \Theta$

$P_0 = P_T$

$W_n = T(1)\sigma^2 / |\underline{H}_n|^2$

while $P_0 \geq \min_{n \in \Theta} W_n$

{ $n = \arg\min_{n \in \Theta} W_n$

$P_0 = P_0 - W_n$

$P_n = P_n + W_n$

$v_n = v_n + 1$

if $v_n = V$, **then** $W_n = \infty$

else $W_n = [T(v_n + 1) - T(v_n)]\sigma^2 / |\underline{H}_n|^2$

}

3. Fine sub-carriers, bits and power allocation

After Step 2, the numbers of allocated sub-carriers for all users are the same, and the most bits are loaded to the sub-carriers through Greedy algorithm. However, due to the fact that CSI for all users varies much, the numbers of loaded bits may vary much for all users. This causes big difference in allocated bits for all users. Hence, Step 3 is used to make modification to the allocated sub-carriers and bits: reallocate sub-carriers of the user with the most number of loaded bits to the user with the least number of loaded bits. In each iteration, we assume the U^{max}-th and the U^{min}-th user obtain the most and the least number of loaded bits for current allocation scheme, respectively. This step reallocates one sub-carrier of the U^{max} -th user to the U^{min} -th user to balance the allocated bits among the users, so as to guarantee the fairness; and in order not to bring down the throughput greatly, the sub-carrier with the worst CSI for the U^{max} -th user is reallocated to the U^{min} -th user. After the re-allocation of sub-carriers, Greedy algorithm is adopted again for bit-loading on the new sub-carrier allocation scheme. The iterations continue until the difference among the loaded bits of all users is low enough. Since this step reallocates the sub-carriers for balance among the loaded bits for all users, the overall throughput may be reduced, but fairness can be guaranteed better. We assume that β_m denotes the loaded bits for the m-th user; S^{max} and S^{min} denotes sub-carriers set which contains sub-carriers allocated to the U^{max} -th user and U^{min} -th user, respectively; S0 denotes the sub-carrier with worst CSI for the U^{max} -th user. Step 3 can be realized as follows.

$$\beta_m = \sum_{\hat{n}\in\{n\,|c(n)=m\}} b(v_{\hat{n}});$$

$$U^{\max} = \arg\max_m(\beta_m); U^{\min} = \arg\min_m(\beta_m)$$

while $\beta_{U^{\max}} - \beta_{U^{\min}} > b_0$

{ $S^{\max} = \{n \mid c(n) = U^{\max}\}, S^{\min} = \{n \mid c(n) = U^{\min}\}$

$S_0 = \arg\min_{n\in S^{\max}}\{|H_{U^{\max},n}|\}$

$c(S_0) = U^{\min}$

$\underline{\Theta} = S_0 \bigcup S^{\min}$

$P_0 = P_0 + \sum_{n'\in\underline{\Theta}} P_{n'}; P_{n'} = 0$

$\underline{H}_{n'} = H_{U^{\min},n'}; W_{n'} = T(1)\sigma^2 / |\underline{H}_{n'}|^2, v_{n'} = 0, \quad \forall n' \in \underline{\Theta}$

while $P_0 \geq \min_{n'\in\underline{\Theta}} W_{n'}$

{ $n' = \arg\min_{n'\in\underline{\Theta}} W_{n'}$

$P_0 = P_0 - W_{n'}$

$P_{n'} = P_{n'} + W_{n'}$

$v_{n'} = v_{n'} + 1$

if $v_{n'} = V$, **then** $W_{n'} = \infty$

else $W_{n'} = [T(v_{n'}+1) - T(v_{n'})]\sigma^2 / |\underline{H}_{n'}|^2\}$

$\beta_m = \sum_{\hat{n}\in\{n|c(n)=m\}} b(v_{\hat{n}});$

$U^{\max} = \arg\max_m(\beta_m); U^{\min} = \arg\min_m(\beta_m)$

}

4.3 Simulation results

In order to give comprehensive evaluation to the proposed method, simulation is carried out and analyzed in the section. The adopted channel model is a typical urban (TU) multi-path channel composed of 6 taps with the maximum time delay of 5μs [12] and the maximum moving velocity is 5kmph. The carrier frequency is 2.5GHz. The system bandwidth is 10MHz, and 840 sub-carriers are available, i.e. *N*=840. Sub-carrier spacing is 11.16 kHz, and CP is set to be 5.6μs to eliminate inter-symbol interference. Candidate modulation schemes in Table 1 are employed for adaptive modulation (bit loading). The proposed method is compared with the fixed allocation method from [9] and ordered allocation method from [11]. And performance for BER, throughput and fairness is provided.

A. BER performance

BER performance reflects the ability to satisfy QoS requirement. As mentioned above, target BER of 10^{-3} is investigated in the section. So it is required that the resource allocation

methods should bring BER lower than 10^{-3}. BER performance with different methods is shown in Fig. 6. Cases for different user number (M= 2, 8, 20) are exhibited. It can be observed that BER is just lower than 10^{-3} for systems with all the resource allocation methods and for different M. Hence, all the methods, including the proposed method, can guarantee QoS requirement for such kind of service.

A. Overall throughput

In order to show the transmission ability of these methods, overall throughput is investigated as shown in Fig. 7. Here the overall throughput is defined as the total transmittable bits per symbol for the system, and can be calculated through

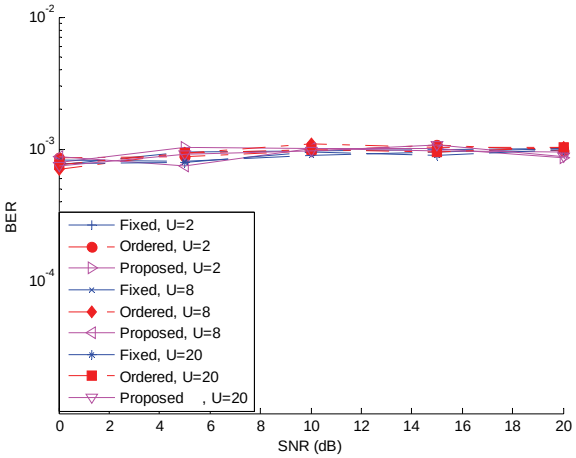


Fig. 6. BER v. s. *SNR* for systems with different methods

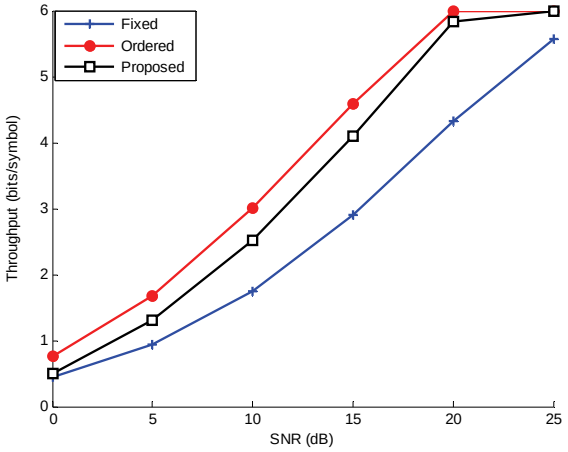


Fig. 7. Overall throughput v. s. *SNR* for M=8

$$\beta = 1/N \sum_{n=1}^{N} b(v_n)(1-\varepsilon) \tag{11}$$

where ε denotes BER. Performance with fixed method, ordered method and the proposed method when M=8 in the section is given out. As can be seen from Fig. 7, performances for the ordered method can obtain the highest overall throughput among all the methods, and throughput for the fixed sub-carrier allocation method is the worst. When *SNR* is low, the proposed method obtains lower throughput than that with the ordered method, but larger than the fixed method. When *SNR*=10dB, it obtains 0.8bits/symbol gain over the fixed method, about 46% improvement. When *SNR* is high enough, the proposed method can obtain the same overall throughput as the ordered method. When *SNR*=20dB, it obtains 1.3bits/symbol gain over the fixed method, and only 0.15bits/symbol lower than the ordered method. The throughput is lower as the cost for excellent fairness, as will be introduced below.

B. Fairness

Though there is little difference in the overall throughput for the proposed method compared with the ordered method, the fairness can be guaranteed very well.

Firstly, we investigate the allocated bit for all users. Fig. 8 shows an example to the allocated bits of all users for different methods when M=8 and *SNR*=25dB. It can be seen that, when the fixed method is adopted, allocated bits number varies much for all the 8 users, the difference between the bits number of the users with most allocated bits and the least allocated bits is as high as 34bits; when the ordered method is adopted, fairness is improved much, and the maximum difference among the allocated bits of all the users is 3 bits. When the proposed method is adopted, all users are allocated with the same number of bits; what's more, the number of bits allocated to the users are almost the same as the maximal value with the ordered method, and much larger than that with the fixed method.

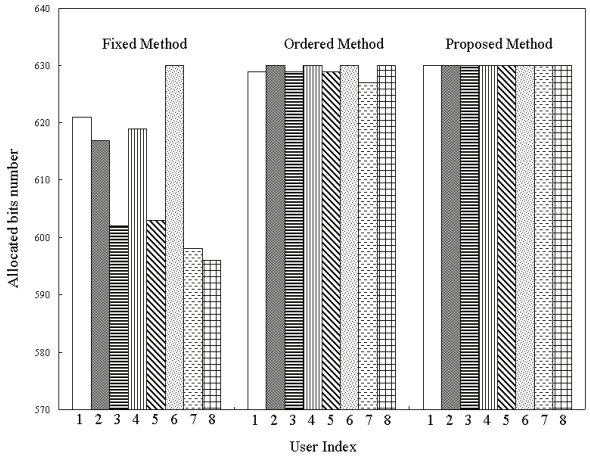


Fig. 8. Allocated bits number for all users with difference methods when M=8 and *SNR*=25dB

Since Fig. 8 is only an example, further work was carried out to prove the advantage of the proposed method. Since there exist channel fading and AWGN for all links, the rightly demodulated bits (or transmittable bits) should be concerned most to evaluate the fairness performance. Therefore, we take the transmittable bits variance (TBV) among all users to evaluate fairness performance of all the methods. TBV can be calculated through

$$TBV = E[|\beta_m \varepsilon_m - E(\beta_m \varepsilon_m)|^2] \tag{12}$$

where $E[\cdot]$ denotes expectation operation; and ε_m denotes BER for the m-th user. Hence, if TBV is large, the transmittable bits numbers for all the users vary much, and such method cannot bring good fairness to the system. Low TBV denotes that the numbers for transmittable bits for all the users are quite near, which means that such method is a fair resource allocation solution.

Transmittable bits variances for the 3 methods versus user number for *SNR*=15dB are shown in Fig. 9. As the number of users increases, the average allocated bits number decreases, so TBV is reduced. For the fixed method, the transmittable bit variance is quite high, which can be explained by the fact that channel fading for different users varies greatly. As for the ordered method, since fairness is guaranteed through allocating the same number of sub-carriers according to CSI of all users, TBV is lower than the fixed method. But the variance is still high, when there are 4 users in the system, TBV can reach more than 10000. The proposed method can perform excellent in fairness. Fig. 9 indicates that it obtains much lower TBV than the fixed method and the ordered method. When the number of users is 4, TBV can be further reduced by about 80% compared to the ordered method.

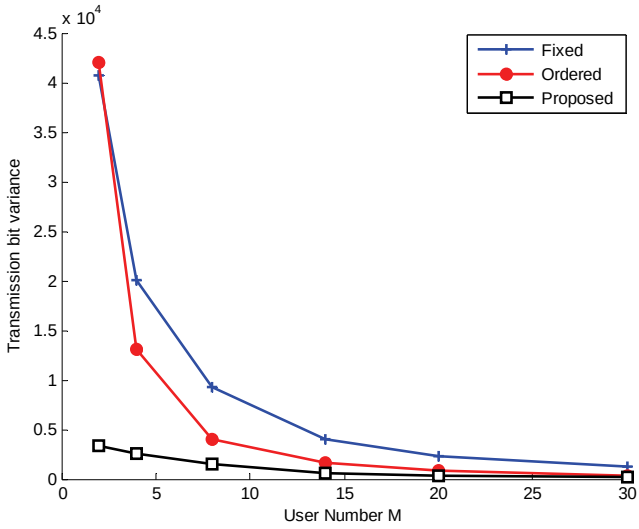


Fig. 9. Transmitted bits variance among users v. s. *M* for *SNR*=15dB

From the simulation results and analysis we can see that, the proposed method takes advantage of Greedy algorithm for many times, and can obtain nice BER performance, so it can provide required QoS guaranteeing. It can bring much larger throughput (more than 40%) than the fixed method. It performs a little worse than the ordered method, but when fairness is concerned, the cost is worthy. The proposed method with application of Greedy algorithm for many times can bring excellent fairness performance over the other methods. It is recommended to be adopted for services with tight fairness requirement among all users.

5. Application of greedy algorithm in OFDM relaying systems

The Greedy algorithm can be also adopted in OFDM relaying systems. In this section, we carry out research into the adaptive bit and power allocation technique in relaying system. Since there are several modes for relaying system, we take amplify-and-forward (AF) mode for example.

5.1 AF-OFDM relaying system model

AF-OFDM relaying system model is shown in Fig. 10, where a two-hop system is considered for its implementation advantages. In the first hop, source bits are modulated, and are allocated with a certain power at source station (SS). Then the modulated signals construct OFDM symbols, which are transmitted to relay station (RS). At RS, power amplification is carried out after signals on all sub-carriers are obtained. In the second hop, these signals are transmitted to the destination station (DS). After demodulation to the received signals on all the sub-carriers, source bits are recovered finally. During both hops, signals experience channel fading and AWGN. Channel transfer functions for the two hops are independent with each other since the two hops happen between different transmitters and receivers on different time slots.

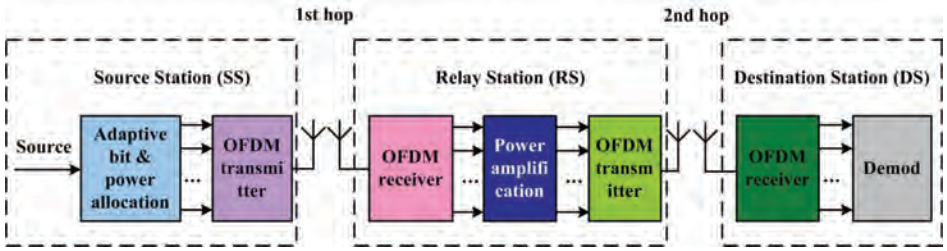


Fig. 10. AF-OFDM relaying system model

We first introduce the basic model. Assume there are K sub-carriers in the OFDM system, and the channel transfer function of the k-th sub-carrier in the i-th hop is $H_{i,k}$. Then the first hop can be expressed as:

$$R_k = H_{1,k}\sqrt{P_{1,k}}S_k + n_{1,k} \tag{13}$$

where S_k and $P_{1,k}$ denote the modulated signal and allocated power on the k-th sub-carrier, respectively; $E[|S_k|^2]=1$; $n_{1,k}$ denotes AWGN with variance of σ_1^2; R_k denotes the received signal at RS.

Amplification is carried out at RS. Assume the amplification factor is ρ_k. Therefore, the average transmit power of the k-th sub-carrier at RS on the second hop is $P_{2,k} = |\rho_k R_k|^2$. Hence, ρ_k can be calculated as

$$\rho_k = \sqrt{P_{2,k} / |R_k|^2} = \sqrt{P_{2,k} /[|H_{1,k}|^2 P_{1,k} + \sigma_1^2]} \tag{14}$$

The second hop can be expressed as:

$$D_k = H_{2,k}\rho_k R_k + n_{2,k} = H_{1,k}H_{2,k}\rho_k\sqrt{P_{1,k}}S_k + H_{2,k}\rho_k n_{1,k} + n_{2,k} \tag{15}$$

where D_k denotes the received signal on the k-th sub-carrier at DS, and $n_{2,k}$ denotes AWGN with variance of σ_2^2. In the section, we consider $\sigma_1^2=\sigma_2^2=\sigma^2$. From (15) we can obtain the SNR of D_k as follows:

$$\gamma_k = [|H_{1,k}H_{2,k}|^2 \rho_k^2 P_{1,k}]/[|H_{2,k}|^2 \rho_k^2 \sigma_1^2 + \sigma_2^2] \tag{16}$$

We take adaptive bit allocation into account, i.e. modulation on all sub-carriers can be adjusted according to the channel condition. We assume that L candidate modulations can be selected, and every signal with the l-th (l=1, 2, …, L) modulation is loaded with $b(l)$ bits. The candidate modulations in the research and their throughputs are shown in Table 2. We assume that the k-th sub-carrier adopts the l_k-th modulation. Hence, the adaptive bit and power allocation (ABPA) problem in a two-hop AF-OFDM relaying system is described as how to allocate bits and power in the transmission (i.e. l_k, $P_{1,k}$ and $P_{2,k}$), so as to maximize system throughput subject to BER requirement and power constraint. In the section, we assume the target BER (BER_{tgt}) to be 10^{-3}. Hence γ_k should be higher than a certain threshold so as to achieve BER_{tgt}. We assume the threshold for the l_k -th modulation is $T(l_k)$, which is also provided in Table I for BER_{tgt} =10^{-3}.

v	Modulation	Number of bits $b(v)$	$T(v)$ (dB)
0	No transmission	0	0
1	QPSK	2	9.78dB
2	16QAM	4	16.52dB
3	64QAM	6	22.52dB
4	256QAM	8	28.4
5	1024QAM	10	35.0

Table 2. Candidate modulation and parameters

Since power constraint may be different for different application scenarios, we research into IPC problem and APC problem in the section, respectively.

5.2 ABPA problem and solutions with IPC

From the analysis above, the ABPA problem with IPC can be described as

$$\arg\max_{l_k, P_{1,k}, P_{2,k}} \sum_{k=1}^{K} b(l_k) \tag{17}$$

subject to

$$\begin{aligned} 1/K \sum_{k=1}^{K} (P_{1,k} + P_{2,k}) \le P \quad &\text{(a)} \\ BER_k = BER_{tgt} \quad &\text{(b)} \end{aligned} \tag{18}$$

Equation (18a) denotes the IPC requirement, i.e. the instantaneous total power cannot be larger than P. And (18b) illustrates BER requirement. From (16) and Table 2, (18b) can be replaced by (19).

$$|H_{1,k}H_{2,k}|^2 \rho_k{}^2 P_{1,k} / [|H_{2,k}|^2 \rho_k{}^2 \sigma_1^2 + \sigma_2^2] = T(l_k) \quad (19)$$

Assume that $G_{i,k}=|H_{i,k}|/\sigma_i^2$, ($i$=1,2). According to (14), we can further rewrite (19) as

$$G_{1,k}G_{2,k}P_{1,k}P_{2,k} / [G_{1,k}P_{1,k} + G_{2,k}P_{2,k} + 1] = T(l_k) \quad (20)$$

Usually, such problem can be solved with Greedy algorithm [13], whose main concept is to achieve the global optimal allocation with many local optimal allocations. During every local optimal allocation, the least additional bits are allocated to the sub-carrier which requires the least additional power to satisfy BER requirement. Such allocations continue until the remaining power is not enough to support more bits. According to Table 2, the least additional bit number for every allocation is 2 in the section.

Since the allocation of power relates to $P_{1,k}$ and $P_{2,k}$, we discuss the problem for different cases: 1) Adaptive PA at SS or RS; 2) Adaptive PA at SS and RS.

A. Adaptive PA at SS or RS

When $P_{2,k}$ is fixed and $P_{1,k}$ can be adjusted according to channel condition, we assume

$$P_{2,k} = P^{RS}, \frac{1}{K}\sum_{k=1}^{K} P_{1,k} = P^{SS} = P - P^{RS} \quad (21)$$

From (20), when the k-th sub-carrier adopts the l_k-th modulation, we can get the required power as follows.

$$P_{1,k}^{SS}(l_k) = T(l_k)[G_{2,k}P^{RS} + 1] / G_{1,k}[G_{2,k}P^{RS} - T(l_k)] \quad (22)$$

Hence, in order to further load additional 2 bits on the k-th sub-carrier, the required additional power is

$$\Delta P_k^{SS} = P_{1,k}^{SS}(l_k + 1) - P_{1,k}^{SS}(l_k) \quad (23)$$

According to Greedy algorithm, the sub-carrier with the minimum $\Delta P_k{}^{SS}$ will be allocated with 2 bits during every allocation. Such allocations will be terminated when the remaining power is not enough to support more bits.

When $P_{1,k}$ is fixed and $P_{2,k}$ can be adjusted, we assume

$$P_{1,k} = P^{SS}, 1/K\sum_{k=1}^{K} P_{2,k} = P^{RS} = P - P^{SS} \quad (24)$$

We take the similar measures to the scheme with adaptive PA at SS. In this case, when the k-th sub-carrier adopts the l_k-th modulation, the required power for the sub-carrier is as follows.

$$P_{2,k}^{RS}(l_k) = T(l_k)[G_{1,k}P^{SS} + 1] / G_{2,k}[G_{1,k}P^{SS} - T(l_k)] \quad (25)$$

And the least additional power to further load additional 2 bits on the k-th sub-carrier is

$$\Delta P_k^{RS} = P_{2,k}^{RS}(l_k + 1) - P_{2,k}^{RS}(l_k) \quad (26)$$

The sub-carrier with minimum ΔP_k^{SS} is allocated with additional 2 bits in every allocation. The courses continue until the remaining power is not enough to support more bits.

B. Adaptive PA at SS and RS

When $P_{1,k}$ and $P_{2,k}$ can both be adjusted, we assume $P_k = P_{1,k} + P_{2,k}$. According to [14], the optimal power allocation strategy to achieve the highest capacity is described as follows.

$$P_{1,k} = P_k / \{1 + \sqrt{[G_{1,k}P_k + 1]/[G_{2,k}P_k + 1]}\}, P_{2,k} = P_k / \{1 + \sqrt{[G_{2,k}P_k + 1]/[G_{1,k}P_k + 1]}\} \quad (27)$$

We combine (27) with (20), and get the following equation.

$$G_{1,k}G_{2,k}P_k^{\,2} / [\sqrt{G_{1,k}P_k + 1} + \sqrt{G_{2,k}P_k + 1}]^2 = T(l_k) \quad (28)$$

According to Greedy algorithm, once we obtain the required power P_k on the k-th sub-carrier with the l_k-th modulation for k=1, 2, …, K, we can determine the sub-carrier which requires the least additional power to support additional 2 bits for every local optimal allocation. From (28), we obtain a quadratic equation regarding P_k as follows.

$$\alpha_2 P_k^{\,2} + \alpha_1 P_k + \alpha_0 = 0 \quad (29)$$

where

$$\begin{aligned}
\alpha_2 &= G_{1,k}{}^2 G_{2,k}{}^2 \\
\alpha_1 &= -2G_{1,k}G_{2,k}T(l_k)(G_{1,k} + G_{2,k}) \\
\alpha_0 &= -4G_{1,k}G_{2,k}T(l_k) + T(l_k)^2 (G_{1,k} - G_{2,k})^2
\end{aligned}.$$

Consequently, when the k-th sub-carrier adopts the l_k-th modulation, the least required power is

$$P_k^{SS,RS}(l_k) = \begin{cases} (-\alpha_1 - \sqrt{\alpha_1^2 - 4\alpha_0\alpha_2}) / 2\alpha_2, if \ \ -\alpha_1 \geq \sqrt{\alpha_1^2 - 4\alpha_0\alpha_2} \\ (-\alpha_1 + \sqrt{\alpha_1^2 - 4\alpha_0\alpha_2}) / 2\alpha_2, if \ -\alpha_1 < \sqrt{\alpha_1^2 - 4\alpha_0\alpha_2} \end{cases} \quad (30)$$

5.3 ABPA Problem and solutions with APC

Unlike the ABPA problem with IPC, the problem with APC requires the average total power to be limited to P. So the problem can be described as

$$\arg\max_{l_k, P_{1,k}, P_{2,k}} \sum_{k=1}^{K} \iint_{(G_{1,k}, G_{2,k})} b(l_k)\, p(G_{1,k}, G_{2,k}) dG_{1,k} dG_{2,k} \quad (31)$$

subject to

$$\begin{aligned}
1/K \sum_{k=1}^{K} \iint_{(G_{1,k}, G_{2,k})} (P_{1,k} + P_{2,k})\, p(G_{1,k}, G_{2,k}) dG_{1,k} dG_{2,k} = P \quad \text{(a)} \\
G_{1,k}G_{2,k}P_{1,k}P_{2,k} / [G_{1,k}P_{1,k} + G_{2,k}P_{2,k} + 1] = T(l_k) \quad \text{(b)}
\end{aligned} \quad (32)$$

where $p(G_{1,k}, G_{2,k})$ denotes the probability of $(G_{1,k}, G_{2,k})$; (32a) and (32b) illustrates APC and BER requirement, respectively.

According to [15], distributions of channel transfer functions for all sub-carriers are the same in an OFDM system. So we omit the subscript k in the following description. Note that the following operation is carried out for all sub-carriers. We assume that Φ_l denotes (G_1, G_2) set whose elements support the l-th modulation. Hence we rewrite the problem as follows.

$$\arg\max_{l,P_1,P_2} \sum_{l=1}^{L} \iint_{(G_1,G_2)\in\Phi_l} b(l)\, p(G_1,G_2)\, dG_1 dG_2 \tag{33}$$

subject to

$$\begin{aligned} &\iint_{(G_1,G_2)} (P_1+P_2)\, p(G_1,G_2)\, dG_1 dG_2 = P \qquad (a) \\ &G_1G_2P_1P_2 / [G_1P_1 + G_2P_2 + 1] = T(l) \qquad (b) \end{aligned} \tag{34}$$

Similar to the problem with IPC, we discuss the problem for different cases: 1) Adaptive PA at SS or RS; 2) Adaptive PA at SS and RS.

A. Adaptive PA at SS or RS

Assume that P_2 is fixed, and P_1 can be adjusted according to channel condition, we assume that

$$P_2 = P^{RS}, \iint_{(G_1,G_2)} P_1\, p(G_1,G_2)\, dG_1 dG_2 = P^{SS} = P - P^{RS} \tag{35}$$

Similar to (25), when the l-th modulation is adopted, the required power is

$$P_1^{SS}(l) = T(l)[G_2P^{RS} + 1] / G_1[G_2P^{RS} - T(l)] \tag{36}$$

Then (35) and (36) can be combined to be

$$\sum_{l=1}^{L} \iint_{(G_1,G_2)\in\Phi_l} \frac{T(l)[G_2P^{RS}+1]}{G_1[G_2P^{RS}-T(l)]}\, p(G_1,G_2)\, dG_1 dG_2 = P^{SS} \tag{37}$$

Hence, the problem of (34) becomes to be which (G_1, G_2) elements belong to Φ_l (l=1, 2, …, L), i.e. which (G_1, G_2) area belongs to Φ_l, so as to maximize average throughput under the constraint of BER and transmit power. Usually, such kind of problem may be solved with aid of Lagrange method. We assume that

$$\begin{aligned} J = &\sum_{l=1}^{L} \iint_{(G_1,G_2)\in\Phi_l} b(l)\, p(G_1,G_2)\, dG_1 dG_2 \\ &+\lambda \left[P^{SS} - \sum_{l=1}^{L} \iint_{(G_1,G_2)\in\Phi_l} \frac{T(l)[G_2P^{RS}+1]}{G_1[G_2P^{RS}-T(l)]}\, p(G_1,G_2)\, dG_1 dG_2 \right] \end{aligned} \tag{38}$$

However, from (38) we can notice that double integrals referring to G_1 and G_2 are involved, and it is impossible to transfer (38) into another problem with single integral. Hence the (G_1,

G_2) area for Φ_l cannot be determined with Lagrange method. In order to solve the problem, we utilize the definition of integral to consider another problem as follows.

$$\sum_{l=1}^{L} \sum_{\substack{i,\, j\in(1,2,\ldots,N) \\ (i\Delta G,\, j\Delta G)\in\Phi_l}} b(l)\, p(i\Delta G,\, j\Delta G)\Delta G^2 \tag{39}$$

subject to

$$\sum_{l=1}^{L} \sum_{\substack{i,\, j\in(1,2,\ldots,N) \\ (i\Delta G,\, j\Delta G)\in\Phi_l}} \frac{T(l)[\, j\Delta G P^{RS} + 1]}{i\Delta G[\, j\Delta G P^{RS} - T(l)]}\, p(i\Delta G,\, j\Delta G)\Delta G^2 = P^{SS} \tag{40}$$

where ΔG is a small real number. Consequently, when N approaches infinity and ΔG approaches infinitesimal, the problem of (39) is just the same as the problem of (33).
We observe the problem of (39), and can find that it can be described as how to allocate bits and power for N^2 different (G_1, G_2) elements, so as to maximize the system throughput. Such problem is similar to the problem in Section 5.2, and can also be solved with Greedy algorithm. With aid of computer simulation, we can obtain the (G_1, G_2) area to adopt the l-th modulation, i.e. Φ_l for all l. The areas of Φ_l when P/σ^2=30dB for Rayleigh fading channel are depicted in Fig. 11, where different colour denotes (G_1, G_2) areas for different modulations.

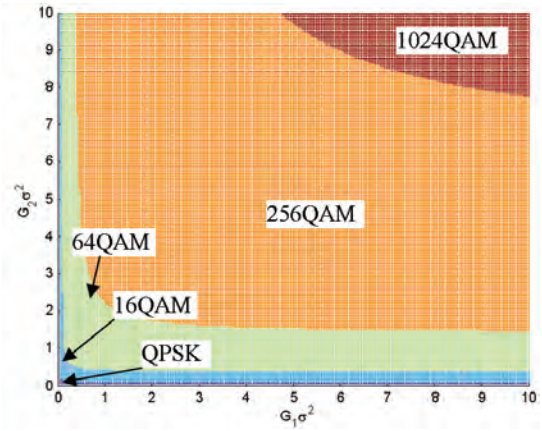


Fig. 11. Area of Φ_l for scheme with adaptive PA at SS

When P_1 is fixed and P_2 can be adjusted according to channel condition, we assume that

$$P_1 = P^{SS},\ \iint_{(G_1,G_2)} P_2\, p(G_1,G_2)\, dG_1 dG_2 = P^{RS} = P - P^{SS} \tag{41}$$

With the same method, we can obtain the (G_1, G_2) area to adopt the l-th modulation, i.e. Φ_l for all l. The areas of Φ_l when P/σ^2=30dB for Rayleigh fading channel are depicted in Fig. 12.

B. Adaptive PA at SS and RS

When P_1 and P_2 can both be adjusted, we have to determine l, P_1, and P_2 jointly. We assume $P_A = P_1 + P_2$, and follow the similar allocation to (27) to allocate power at SS and RS. In order

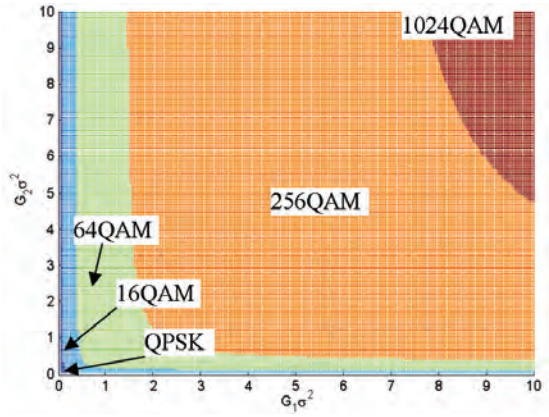


Fig. 12. Area of Φ_l for scheme with adaptive PA at RS

to determine the (G_1, G_2) area for the l-th modulation, we consider another problem as follows.

$$\sum_{l=1}^{L} \sum_{\substack{i,\, j\in(1,2,\ldots,N) \\ (i\Delta G,\, j\Delta G)\in\Phi_l}} b(l)\, p(i\Delta G,\, j\Delta G)\Delta G^2 \tag{42}$$

subject to

$$\sum_{l=1}^{L} \sum_{\substack{i,\, j\in(1,2,\ldots,N) \\ (i\Delta G,\, j\Delta G)\in\Phi_l}} b(l)\, p(i\Delta G,\, j\Delta G)\Delta G^2 \tag{38a}$$

With the same method above, we can obtain Φ_l for all l in the case. The areas of Φ_l when P/σ^2=30dB for Rayleigh fading channel are depicted in Fig. 13.

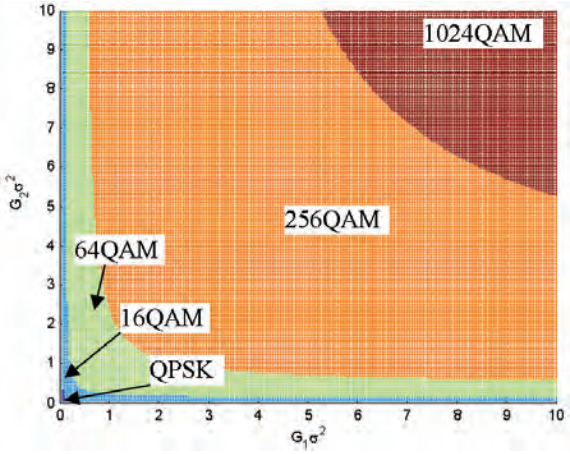


Fig. 13. Area of Φ_l for scheme with adaptive PA at SS and RS

From Fig. 11 to Fig. 13, we can see that the boundaries between the areas of Φ_l are like hyperbolas. The areas of Φ_l for the scheme with adaptive PA at SS and the scheme with adaptive PA at RS are reverse. As for the scheme with adaptive PA at SS and RS, the areas to adopt higher order modulation (e.g. 1024QAM) are wider than the other two schemes. That is to say, this scheme is more likely to adopt higher order modulation (allocated with more bits), because both P_1 and P_2 can be adjusted and more adaptation can be obtained.

5.4 Simulation and analysis

In order to evaluate the performance of the proposed solutions in the section, numerical simulation is carried out, in which both large-scale fading and small-scale fading are both taken into consideration. The number of sub-carriers is 32. We assume $H_{1,k} = h_{1,k}\sqrt{D_{SR}^{-\eta}}$, and $H_{2,k} = h_{2,k}\sqrt{D_{RD}^{-\eta}}$, where $h_{1,k}$ and $h_{2,k}$ denote the small-scale fading, and they follow Rayleigh distribution; D_{SR} and D_{RD} denote the distance between SS and RS, and distance between RS and DS, respectively; $D_{SR} + D_{RD} = 1$. η denotes path loss exponent for large-scale fading, which is 4 in the simulation; $H_{1,k}$ and $H_{2,k}$ are assumed to be available for SS and RS. For reason of fairness, we assume $P_{SS} = P_{RS} = P/2$. For the following description, we define $SNR = P/\sigma^2$. The text in the legend is made short, e.g. "adaptive SS" is short for "adaptive PA at SS".

1) BER performance: First, we investigate BER performance of the proposed schemes in the section. Fig. 14 gives out their BER performance vs. *SNR* for the schemes. When *SNR* is low, low order modulation (e.g. QPSK) may be adopted frequently; when *SNR* is high, higher order modulation (e.g. 256QAM) may be adopted, so BER remains almost the same as that when *SNR* is low. It can be seen that all solutions can bring to BER performance close to the target BER 10^{-3}. This is because the power allocation at SS and RS can make the SNR of received signals to be the threshold to satisfy $BER_{tgt.}$ Hence, the proposed schemes can perform well in BER performance.

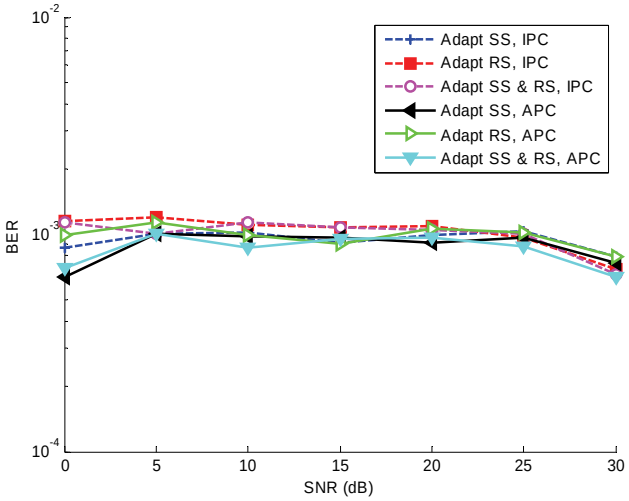


Fig. 14. BER vs. *SNR* for the solutions when D_{SR}=0.5

2) Transmit power constraint: Fig. 15 shows the cumulative distribution function (CDF) of the normalized total power P_A/P, where $P_A = 1/K\sum_{k=1}^{K}(P_{1,k}+P_{2,k})$. As for the three schemes with IPC, the distributions of P_A are almost the same. The value of P_A/P ranges from 0.92 to 1, which means that the instantaneous total power P_A is always lower than P, which is the requirement of IPC. As for the three schemes with APC, the distributions of P_A are almost the same, too. Hence, for the scheme of adaptive PA at SS and RS, though the transmit power can both be adjusted at SS and RS, the range for the total power is not improved, i.e. more adaptation on power doesn't cause the total power to vary more. From the figure, P_A of the schemes with APC can range more widely than that with IPC, from 0.75 to 1.25 approximately, but its mean value is restricted to be P. Compared with the scheme with IPC, the scheme with APC can take use of more power in average.

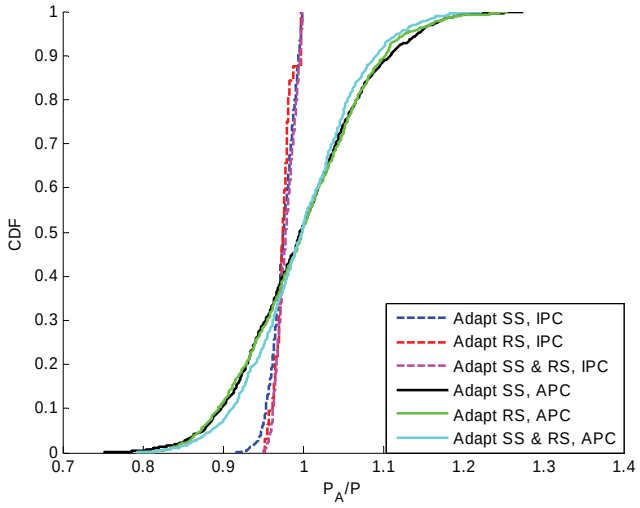


Fig. 15. CDF of total power for the solutions when D_{SR}=0.5 and *SNR*=20dB

3) System throughput: Fig. 16 and Fig. 17 gives out the system throughput vs. DSR when *SNR* equals to 0dB and 20dB, respectively. The throughput is calculated as the average correctly transmitted bit number per OFDM symbol divided by the number of sub-carriers. Throughput may be 0 because the channel condition is terrible and "no transmission" is adopted. The schemes with IPC has lower throughput than that with APC, due to the fact that they obtains different constraint for total transmit power. When *SNR* equals to 0dB, the difference can achieve 10% (0.04bits/symbol). When *SNR* is higher, throughputs for IPC and APC solutions are quite close, because the impact of transmit power constraint is not dominant for throughput when *SNR* is high.

As a conclusion, the Greedy algorithm can be applied into OFDM relaying system to achieve the maximum throughput subject to the BER and transmit power constraint.

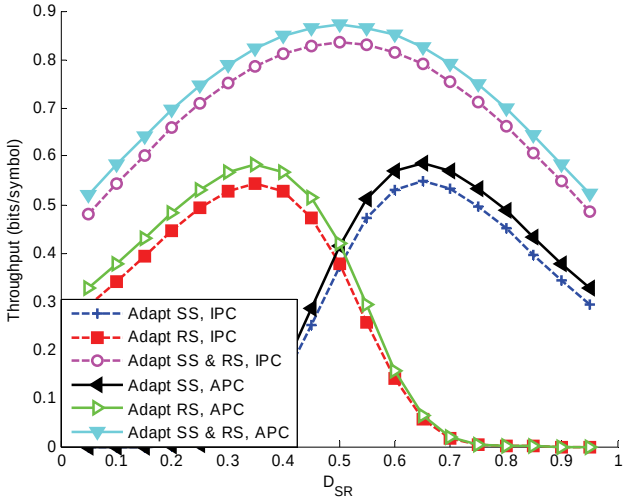


Fig. 16. System throughput vs. D_{SR} for the solutions when SNR=0dB

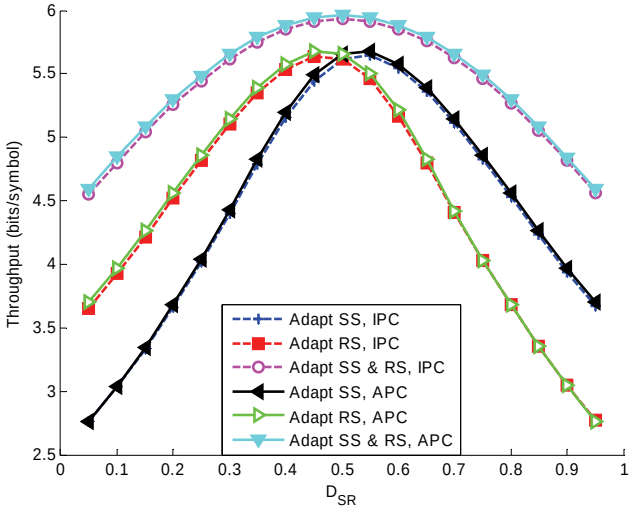


Fig. 17. System throughput vs. D_{SR} for the solutions when SNR=20dB

6. Conclusion

In this chapter we have presented the potential of link adaptation technique in wideband wireless communication systems with aid of Greedy algorithm. Link adaptation in OFDM systems is introduced firstly in section 2, i.e. how to allocation bits and power in all sub-carriers to maximize the throughput. The optimal solution can be obtained with Greedy algorithm. The detail description is given out in section 3. Simulation results are shown, which indicate that the throughput is maximized with guaranteed BER performance. When multiple user case is concerned, the problem gets more complex. The section 4 provides a

novel solution, which takes use of Greedy algorithm for several times to obtain better fairness than the existing schemes. Simulation is also carried out to show the performance. Furthermore, due to the fact that relaying technique is an important component for future systems, the link adaptation in relaying systems is researched in section 5. AF-OFDM systems are investigated. The problem can be described as how to allocate bits and power in SS and RS to obtain the highest throughput subject to BER and transmit power constraint. Greedy algorithm helps to achieve the optimal result. Numeral simulation results indicate the proposed scheme perform well with satisfied BER and transmit power constraint. Further conclusions are obtained through the research. As a conclusion, with aid of Greedy algorithm, the potential of link adaptation technique is explored greatly.

7. References

Alamouti, S. M. & Kallel, S. (1994). Adaptive trellis-coded multiple-phase-shift keying for Rayleigh fading channels, *IEEE Trans. on Comm.*, Vol. 42, No. 6, 1994, pp. 2305-2314

Goldsmith J. & Ghee C. S. (1998). Adaptive coded modulation for fading channels, *IEEE Trans. on Comm.*, Vol. 46, No. 5, 1998, pp. 595-602

Goldsmith J. & Ghee C. S. (1997). Variable–rate variable-power MQAM for fading channels, IEEE Trans. on Comm., Vol. 45, Oct. 1997, pp. 1218-1230

Bahai R. S. & Saltzberg B. R. (1999). *Multi-carrier digital communications: theory and applications of OFDM*, Kluwer Academic / Plenum publishers, 2nd ed, New York

Keller T. & Hanzo L. (2000) Adaptive multicarrier modulation: a convenient framework for time-frequency processing in wireless communications, *in Proceedings of the IEEE*, Vol. 88, No. 5, 2000, pp. 611-640

Campello J. (1998). Optimal discrete bit loading for multicarrier modulation systems, *IEEE International Symposium on Information Theory*, Aug. 1998, pp. 193

Wong C. Y.; Cheng R.; Lataief K. B.; et al. (1999). Multiuser OFDM system with adaptive subcarrier, bit, and power allocation, *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 10, Oct. 1999, pp. 1747-1758

Rhee W. & Cioffi J. M. (2000). Increase in capacity of multiuser OFDM system using dynamic subchannel allocation, *Vehicular Technology Conference Proceedings, VTC 2000-Spring*, Vol. 2, May 2000, pp. 1085-1089

Wong C. Y.; Cheng R. & Letaief K. B. (1999). Multiuser subcarrier allocation for OFDM transmission using adaptive modulation, *Vehicular Technology Conference, VTC 1999-Spring*, Vol. 1, May 1999, pp. 16-20

Fu J. & Karasawa Y. (2002). Fundamental analysis on throughput characteristics of orthogonal frequency division multiple access (OFDMA) in multipath propagation environments, *IEICE transaction*, Vol. J85-B, No. 11, Nov. 2002, pp. 1884-1894

Otani Y.; Ohno S.; Teo K. D.; et al., (2005). Subcarrier allocation for multi-user OFDM system, *Asia-Pacific Conference on Communications*, Oct. 2005, pp. 1073 – 1077

3GPP, TS 45.005, Radio transmission and reception (Release 7), 2006

Chung S. T. & Goldsmith A. J. (2001). Degrees of freedom in adaptive modulation: a unified view, *IEEE Trans. Comm.*, Vol. 49, No. 9, Sep. 2001, pp. 1561-1571

Yu G.; Zhang Z.; Chen Y. etc. (2005). Power allocation for non-regenerative OFDM relaying channels, *Wireless Comm. Networking and Mobile Computing*, Vol. 1, 2005, pp. 185-188

Zhou M.; Li L.; Wen N.; etc. (2006). Performance of LDPC coded AOFDM under frequency-selective fading channel, *International Conference on Communication, Circuits and Systems*, Vol. 2, June 2006, pp. 861-865

11

Greedy Algorithms for Mapping onto a Coarse-grained Reconfigurable Fabric[1]

Colin J. Ihrig, Mustafa Baz, Justin Stander, Raymond R. Hoare, Bryan A. Norman, Oleg Prokopyev, Brady Hunsaker and Alex K. Jones
University of Pittsburgh, United States

1. Introduction

This book chapter describes several greedy heuristics for mapping large data-flow graphs (DFGs) onto a stripe-based coarse-grained reconfigurable fabric. These DFGs represent the behavior of an application kernel in a high-level synthesis flow to convert computer software into custom computer hardware. The first heuristic is a limited lookahead greedy approach that provides excellent run times and a reasonable quality of result. The second heuristic expands on the first heuristic by introducing a random element into the flow, generating multiple solution instances and selecting the best of the set. Finally, the third heuristic formulates the mapping problem of a limited set of rows using a mixed-integer linear program (MILP) and creates a sliding heuristic to map the entire application. In this chapter we will discuss these heuristics, their run times, and solution quality tradeoffs.

The greedy mapping heuristic follows a top-down approach to provide a feasible mapping for any given application kernel. Starting with the top row, it completely places each individual row using a limited look-ahead of two rows. After each row is mapped, the mapper will not modify the mapping of any portion of that row. This mapping approach is deterministic as it uses a priority scheme to determine which elements to place first based on factors such as the number of nodes to which it connects and second based on the desirability of a particular location in the row. While the limited information available to the mapper does not often allow it to produce optimal or minimum-size mappings, its runtime is typically a few seconds or less. We use a fabric interconnect model (FIM) file in the mapping flow to define a set of restrictions on what interconnect lines are available, the capabilities of particular functional units (e.g. dedicated vertical routes versus computational capabilities) in the system, etc.

The greedy heuristic is deterministic in the priority system which it uses to place nodes. The second mapping heuristic we explore is based on this greedy algorithm and introduces randomness into the heuristic to make decisions along the priority list. In the first implementation the node selection order is selected randomly. In the second version, weights are assigned to nodes based on the deterministic placement order. Since the heuristic runs so quickly, we can run the heuristic 10's or possibly 100's of times and select the best result. This method is also parameterizable with the FIM.

[1] This work partially supported by The Technology Collaborative.

Finally, we present a sliding window algorithm where groups of rows are placed using an MILP. This heuristic starts with an arbitrary placement where operations are placed in the earliest row possible and the operations are left justified. Starting from the top, a window of rows is selected and the IP algorithm adjusts column locations where the optimization criteria is to only use allowed routes specified by the architecture. If the program cannot find a feasible mapping, it tries to push violated edges (i.e. edges that do not conform to what is allowed in the architecture) down in the window so that subsequent windows may be able to find a solution. If no feasible solution can be found in the current window, then a row of pass-gates is added to increase the flexibility, and the MILP is run again. However, introducing a row of pass-gates delays the critical path and is undesirable from a power and performance perspective. This technique is also parameterizable within the FIM.
In this chapter, these three heuristics will be explained in detail and numerous performance evaluations (including feasibility) will be conducted for different architectural configurations. Section 2 provides a background on the reconfigurable fabric concept and the process of mapping as well as related work. Section 3 introduces the Fabric Interconnect Model, an XML representation of the fabric. In Section 4 the greedy heuristic is described in detail. In particular, the algorithms for row and column placement are discussed. Section 5 extends the greedy heuristic by introducing an element of randomness into the algorithm. Several methods of randomizing the greedy heuristic are explored, including completely random decisions and weighted decisions. In Section 6 the sliding partial MILP heuristic is introduced. In addition, several techniques for improving the execution time of the MILP are discussed. These techniques are based on decomposing the problem into smaller, simpler linear programs. Finally, Section 7 compares the different mapping techniques and provides some conclusions.

2. Background and literature review

A general trend seen during application profiling is that 90% of application execution time in software is spent in approximately 10% of the code. The idea of our reconfigurable device is to accelerate high incidence code segments (e.g. loops) that require large portions of the application runtime, called kernels, while assigning the control-intensive portion of the code to a core processor.

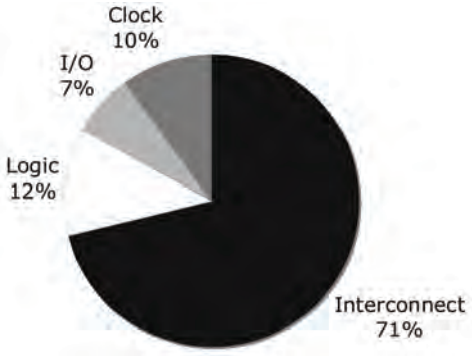


Fig. 1. Power consumption features of a Xilinx Virtex-2 3000 FPGA (Sheng et al., 2002).

A tremendous amount of effort has been devoted to the area of hardware acceleration of these kernels using Field Programmable Gate Arrays (FPGAs). This is a particularly popular method of accelerating computationally intensive Digital Signal Processing (DSP) applications. Unfortunately, while FPGAs provide a flexible reconfigurable target, they have poor power characteristics when compared to custom chips called Application Specific Integrated Circuits (ASICs). At the other end of the spectrum, ASICs are superior in terms of performance and power consumption, but are not flexible and are expensive to design.

The dynamic power consumption in FPGAs has been shown to be dominated by interconnect power (Sheng et al., 2002). For example, as shown in Figure 1, the reconfigurable interconnect in the Xilinx Virtex-2 FPGA consumes more than 70% of the total power dissipated in the device. Power consumption is exacerbated by the necessity of bit-level control for the computational and switch blocks.

Thus, a reconfigurable device that exhibits ASIC-like power characteristics and FPGA-like flexibility is desirable. Recently, the development and use of coarse-grained fabrics for computationally complex tasks has received a lot of attention as a middle ground between FPGAs and ASICs because they typically have simpler interconnects. Many architectures have been proposed and developed, including MATRIX (Mirsky & Dehon, 1996), Garp (Hauser & Wawrzynek, 1997), PipeRench (Levine & Schmit, 2002), and the Field Programmable Object Array (FPOA) (MathStar, MathStar).

Our group has developed the SuperCISC reconfigurable hardware fabric to have low-energy consumption properties compared to existing reconfigurable devices such as FPGAs (Mehta et al., 2006; Jones et al., 2008; Mehta et al., 2006, 2007, 2008). To execute an application on the SuperCISC fabric, the software kernels are converted into entirely combinational hardware functions represented by DFGs, generated automatically from C using a design automation flow (Jones et al., 2005, 2006; Hoare et al., 2006; Jones et al., 2006). Stripe-based hardware fabrics are designed to easily map DFGs from the application into the device. The architecture of the SuperCISC fabric (and other stripebased fabrics such as PipeRench) work in a similar way, retaining a data flow structure, which allows computational results to be computed in one multi-bit functional-unit (FU) and flow onto others in the system. FUs are organized into rows or *computational stripes*, within which each functional unit operates independently. The results of these operations are then fed into *interconnection stripes* which are constructed using multiplexers. Figure 2 illustrates this top-down data flow concept. The process of mapping these DFGs onto the SuperCISC fabric is described in the next section.

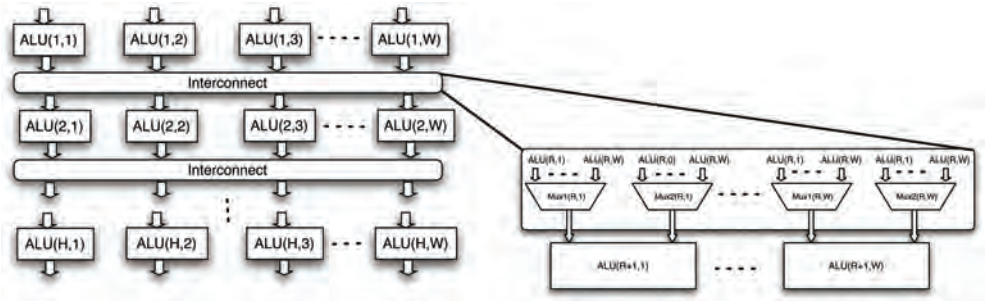
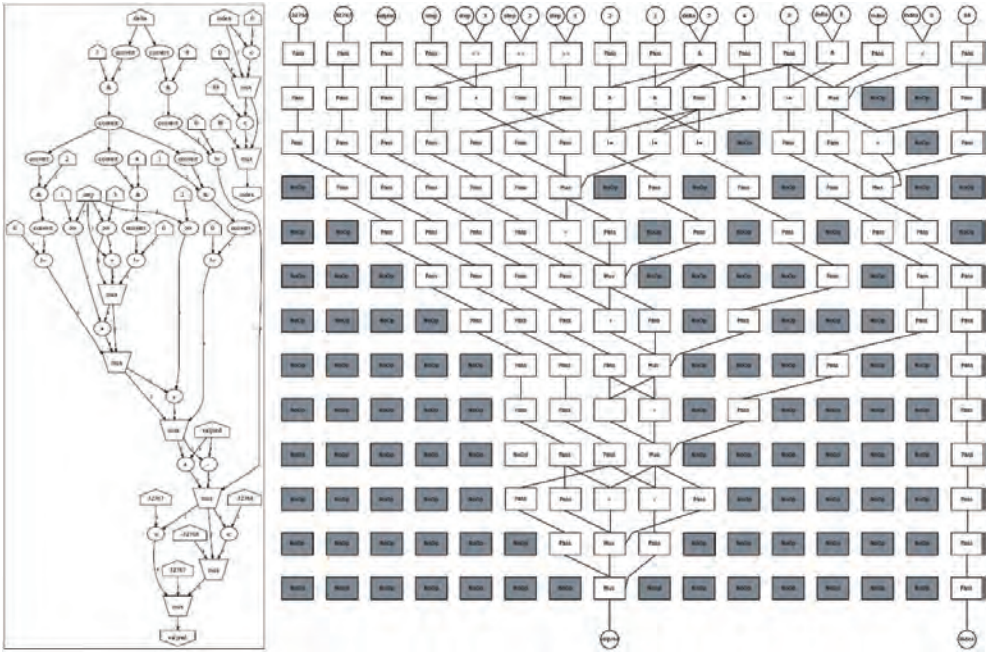


Fig. 2. Fabric conceptual model.

2.1 Mapping

A mapping of a DFG onto a fabric consists of an assignment of operators in the DFG to FUs in the fabric such that the logical structure of the DFG is preserved and the architectural constraints of the fabric are respected. This mapping problem is central to the use of the fabric since a solution must be available each time the fabric is reprogrammed for a different DFG. Because of the layered nature of the fabric, the mapping is also allowed to use FUs as "pass-gates," which take a single input and pass the input value to one or more outputs. In general, not all of the available FUs and edges will be used. An example DFG and a corresponding mapping are shown in Figure 3.



(a) Example data flow graph (DFG). (b) Example mapping.

Fig. 3. Mapping problem overview.

The interconnect design—that is, the pattern of available edges—is the primary factor in determining whether a given DFG can be mapped onto the fabric. For flexibility, it would make sense to provide a complete interconnect with each FU connected to every FU in the next row. The reason for limiting the interconnect is that the cardinality of the interconnect has a significant impact on energy consumption. Although most of the connections are unused, the increased cardinality of the interconnect requires more complicated underlying hardware, which leads to greater energy consumption. For a more detailed description of this phenomenon, see (Mehta et al., 2006), which indicates that this energy use can be significant. Therefore, we consider limited interconnects, which have better energy consumption but make the mapping problem more challenging.

We consider the mapping problem in three forms. We call these problems Minimum Rows Mapping, Feasible Mapping with Fixed Rows and Augmented Fixed Rows. These problems are briefly described in the following subsections.

2.1.1 Minimum rows mapping

Given a fixed width and interconnect design, a fabric with fewer rows will use less energy than one with more rows. As data flows through the device from top to bottom it traverses FUs and routing channels, consuming energy at each stage. The amount of energy consumed varies depending on the operation that an FU performs. However, even just passing the value through the FU consumes a significant amount of energy. Thus, the number of rows that the data must traverse impacts the amount of energy that is consumed. If the final result has been computed, the data can escape to an early exit, which bypasses the remaining rows of the fabric and reduces the energy required to complete the computation. Therefore, it is desirable to use as few rows as possible. Given a fabric width, fabric interconnect design, and data flow graph to be mapped, the Minimum Rows Mapping problem is to find a mapping that uses the minimum number of rows in the fabric. The mapping may use pass-gates as necessary.

We initially formulated a MILP to solve this problem, however, it has only been able to solve nearly trivial instances in a reasonable amount of time (Baz, 2008). We have since developed two heuristic approaches to solve this problem: a deterministic top-down greedy heuristic described in Section 4 and a heuristic that combines the top-down approach with randomization, described in Section 5.

2.1.2 Feasible mapping with fixed rows

One of the more complicated parts of creating a mapping is the introduction of pass-gates to fit the layered structure of the fabric. One approach that we have used is to work in two stages. In the first stage, pass-gates are introduced heuristically and operators assigned to rows so that all edges go from one row to the next. The second stage assigns the operators to columns so that the fabric interconnect is respected. This second stage is called Feasible Mapping with Fixed Rows. Note that depending on the interconnect design, there may or may not exist such a feasible mapping.

We have formulated a MILP approach to solve this problem described in detail in (Baz et al., 2008; Baz, 2008). This formulation can provide us with a lower bound with which to compare our heuristic solutions.

2.1.3 Augmented fixed rows

This problem first tries to solve the Feasible Mapping with Fixed Rows problem. If this is infeasible, then it may add a row of pass-gates to gain flexibility. It then tries to solve Feasible Mapping with Fixed Rows on the new problem. This is repeated until a solution is found or a limit is reached on the number of rows to add.

We have developed a partial sliding MILP heuristic in Section 6 to solve this problem.

2.1.4 Related work

There are two problems in graph theory related to the mapping problems we present. First, Feasible Mapping with Fixed Rows may be viewed as a special case of subgraph isomorphism, also called subgraph containment. The DFG (modified to have fixed rows) may be considered as a directed graph G, and the fabric may be considered as a directed graph H. The problem is to identify an isomorphism of G with a subgraph of H.

Most of the work on subgraph isomorphism uses the idea of efficient backtracking, first presented in (Ullmann, 1976). Examples of more recent work on the problem include

(Messmer & Bunke, 2000; Cordella et al., 2004; Krissinel & Henrick, 2004). In each of these cases, algorithms are designed to solve the problem for arbitrary graphs. In contrast, the graphs for our problem are highly structured, and our approaches take advantage of this structure. Subgraph isomorphism is NP-complete (Garey & Johnson, 1979).

If we fix the number of rows in the fabric, then finding a feasible mapping (but not minimizing the number of rows) may be viewed as a special case of a problem known as directed minor containment (Diestel, 2005; Johnson et al., 2001). The DFG may be considered as a directed graph *G*, and the fabric may be considered as a directed graph *H*. Directed minor containment (also known as butterfly minor containment) is the problem of determining whether *G* is a directed minor of *H*. Unlike subgraph isomorphism, *G* may be a directed minor without being a subgraph; additional nodes (corresponding to "pass-gates" in our application) may be present in the subgraph of *H*. Directed minor containment is also NP-complete. We are not aware of any algorithms for solving directed minor containment on general graphs or graphs similar to our fabric mapping problem.

2.2 Routing complexity

The fundamental parameter in the design of a coarse-grain reconfigurable device for energy reduction is the flexibility and resulting complexity of the interconnect. For example, a simpler interconnect can lead to architectural opportunities for energy reduction (fewer wires, simpler selectors, etc.) but can also make the mapping problem more difficult. As discussed in Section 2.1, the quality of the mapping solution also impacts the energy consumed by the design. Thus, to effectively leverage the architectural energy saving opportunities the mapping algorithms must become increasingly sophisticated.

As previously mentioned, the interconnection stripes are constructed using multiplexers. The cardinality of these multiplexers determines the routing flexibility and the maximum sources and destinations allowed for nodes in the DFG. This is shown in Figure 4. The interconnect shown in Figure 4(a) is built using 2:1 multiplexers, and is said to have a cardinality of two. Similarly, the interconnect in Figure 4(b) is comprised of 4:1 multiplexers, and is said to have a cardinality of four. By comparing these figures, it is obvious that the higher cardinality interconnect is more flexible because each functional unit can receive input from a larger number of sources. Essentially, a higher cardinality interconnect has fewer restrictions, which leads to a simpler mapping problem.

(a) Cardinality of two. (b) Cardinality of four.

Fig. 4. Interconnects of two different multiplexer cardinalities.

While the flexibility of higher cardinality multiplexers is desirable for ease of mapping, these multiplexers are slower, more complex, and dissipate more power than lower cardinality multiplexers. A detailed analysis of the power consumption versus cardinality is conducted in (Jones et al., 2008; Mehta et al., 2007, 2006).

Additionally, when mapping a DFG to a stripe-style structure, data dependency edges often traverse multiple rows. In these fabrics, FUs must often pass these values through without doing any computation. We call these operations in the graph, pass-gates. However, these FUs used as pass-gates are an area and energy-inefficient method for vertical routing. Thus, we explored replacing some percentage of the FUs with dedicated vertical routes to save energy (Mehta et al., 2008; Jones et al., 2008). However, these dedicated pass-gates can make mapping more difficult because it places restrictions on the placement of operators. The work in (Mehta et al., 2008; Jones et al., 2008) only uses the first of the three greedy heuristics presented here and required that the interconnect flexibility be relaxed when introducing dedicated vertical routes. The more sophisticated greedy algorithms were designed in part to improve the mapping with the more restrictive multiplexer cardinalities along with dedicated pass-gates.

The purpose of these heuristics is to provide high quality of solution mappings onto the low-energy reconfigurable device. One way to measure the effectiveness is to examine the energy consumed from executing the device with various architectural configurations and different data sets, etc. We obtain these energy results from extremely time consuming power simulations using computer-aided design tools. However, in this paper we chose to focus our effort on achieving a high quality of solution from the mapping algorithms. Conducting power simulations for each mapping would significantly limit the number of mapping approaches we could consider.

Thus, we can examine two factors to evaluate success: the increase in the total path length of the mapped algorithm and the number of FUs used as pass-gates. The total path length in the mapped design is the sum of the number of rows traversed from each input to each output. Thus, the path length increase is the increase in the total path length from a solution where each computation is completed as early as possible limited only by the dependencies in the graph (see Section 4.1). The number of FUs used as pass-gates is useful in judging success in cases where the fabric contains dedicated pass-gates. Dedicated pass-gates are more energy efficient than complex functional units at passing a value (more than an order of magnitude (Jones et al., 2008)). Thus, when using dedicated-pass gates the fewer Fus used as pass-gates, the better.

To demonstrate that these factors influence the energy consumption of the device, we ran a two-way analysis of variance (ANOVA) on the energy with the number of FUs used as pass-gates and path length as factors to determine the correlation. Using an alpha value of 0.05, both factors significantly influenced the energy (p<0.01 and p=0.031, respectively).

3. The Fabric Interconnect Model (FIM)

As various interconnection configurations were developed, redesigning the mapping flow and target fabric hardware by hand for each new configuration was impractical (Mehta et al., 2007). Additionally, we envision the creation of customizable fabric intellectual property (IP) blocks that can be used in larger system-on-a-chip (SoC) designs. To make this practical, it is necessary to create an automation flow to generate these custom fabric instances.

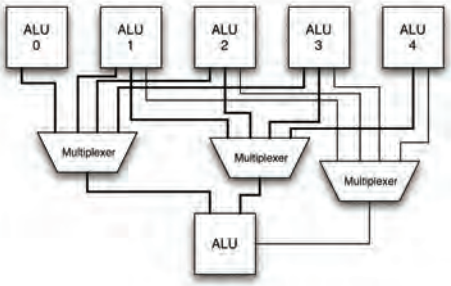
To solve this problem, we created the FIM, a textual representation used to describe the interconnect and the layout and make-up of the FUs in the system. The FIM becomes an input file to the mapper as well as the tool that generates a particular instance of the fabric with the appropriate interconnect.

The FIM file is written in the Extensible Markup Language (XML) (Bray et al., 2006). XML was selected as it allowed the FIM specification to easily evolve as new features and

descriptions were required. For example, while the FIM was initially envisioned to describe the interconnect only, it has evolved to describe dedicated pass-gates and other heterogeneous functional unit structures.

Figure 5(a) shows an example partial FIM file that describes a cardinality five interconnect. A cardinality five interconnect is a specially designed interconnect, which is actually constructed using mirrored 4:1 multiplexers. In Figure 4(b) a single multiplexer is depicted as providing all three inputs to each FU, also known as an ALU (arithmetic logic unit). In reality, each of the three inputs has its own individual multiplexer. By allowing the multiplexers to draw their inputs from different locations, 4:1 multiplexers can be used to create the illusion of a limited 5:1 multiplexer. This limited 5:1 multiplexer provides a surprisingly higher flexibility over a cardinality four interconnect with no cost in terms of hardware complexity.

```
<rowpattern repeat="forever">
 <row>
  <ftupattern repeat="forever">
   <FTU type="ALU">
    <operand number="0">
     <range left ="-2" right ="1"/>
    </operand>
    <operand number="1">
     <range left ="-1" right ="2"/>
    </operand>
    <operand number="2">
     <range left ="-1" right ="2"/>
    </operand>
   </FTU>
  </ftupattern>
 </row>
</rowpattern>
```



(a) FIM file example for 5:1 style interconnect. (b) 5:1 style interconnect implementation.

Fig. 5. Describing a 5:1 multiplexing interconnect using a FIM file.

The pattern in Figure 5(a) repeats the interconnect pattern for `ALU`, whose zeroth operand can read from two units to the left, the unit directly above, and one unit to the right. The first operand is the mirror of the zeroth operand, reading from two units to the right, the unit directly above, and one unit to the left. The second operand, which has the same range as the first operand, serves as the selection bit if the FU is configured as a multiplexer. The resulting cardinality five interconnect implementation is shown in Figure 5(b). As specified in the FIM, the zeroth operand of `ALU` can access `ALU0` through `ALU3`, while the first and second operands can access `ALU1` through `ALU4`.

The ranges in the FIM can be discontinuous by supplying additional range flags. The file can contain a heterogeneous interconnect by defining additional Fabric Topological Units (`FTU`s) with different interconnect ranges. The pattern can either repeat or can be arbitrarily customized without a repeating pattern for a fixed size fabric.

The design flow overview using the FIM is shown in Figure 6. The SuperCISC Compiler (Hoare et al., 2006; Jones et al., 2006) takes C code input, which is compiled and converted into a Control and Data Flow Graph (CDFG). A technique known as *hardware predication* is applied to the CDFG in order to convert control dependencies (e.g. *if-else* structures) into data dependencies through the use of selectors. This post-predication CDFG is referred to as

a *Super Data Flow Graph* (SDFG). The SDFG is then mapped into a configuration for the fabric described by the FIM.

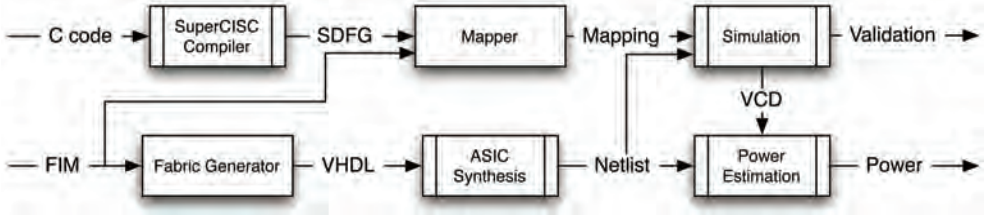


Fig. 6. Interconnect evaluation tool flow.

The FIM is also used to automatically generate a synthesizable hardware description of the fabric instance described by the FIM. For testing and energy estimation, the fabric instance can be synthesized using commercial tools such as Synopsys' Design Compiler to generate a netlist tied to ASIC standard cells. This netlist and the mapping of the application are then fed into ModelSim where correctness can be verified. The mapping is communicated to the simulator to program the fabric device in the form of ModelSim `do` files. A value change dump (VCD) file output from the simulation of the design netlist can then be used to determine the power consumed in the design. However, due to the effort required to generate a single power result we will use mapping quality metrics such as path length increase and FUs used as pass-gates rather than energy consumption to evaluate the quality of our mapping heuristics as described in Section 2.2.

The FIM is incorporated into the mapping flow as a set of restrictions on both the interconnect and the functional units in each row. In addition to creating custom interconnects, the FIM can be used to introduce heterogeneity into the fabric's functional units. This capability is used to allow the introduction of dedicated pass-gates into the target architecture and greedy mapping approaches.

4. Deterministic greedy heuristic

A heuristic mapping algorithm overviewed in Algorithm 1 was developed to solve the problem of Minimum Rows Mapping. The instantiation of this algorithm reads both the DFG and the FIM to generate its mapping result. The heuristic is comprised of two stages of row assignment followed by column assignment, which follows a top-down mapping approach using a limited look-ahead of two rows. In the first line of the algorithm each node is assigned to a row as described in Section 4.1. In the second stage, as shown in the algorithm, the column locations for nodes in each row are assigned starting with the top row. This is described in Section 4.2. After each row is mapped, the heuristic will not modify the locations of any portion of that row.

While the limited information available to the heuristic does not often allow it to produce optimal minimum-size mappings, its relative simplicity provides a fast runtime. By default the heuristic tries to map the given benchmark to a fabric with a width equal to the largest individual row, and a height equal to the longest path through the graph representing the input application. Although the width is static throughout a single mapping, the height can increase as needed.

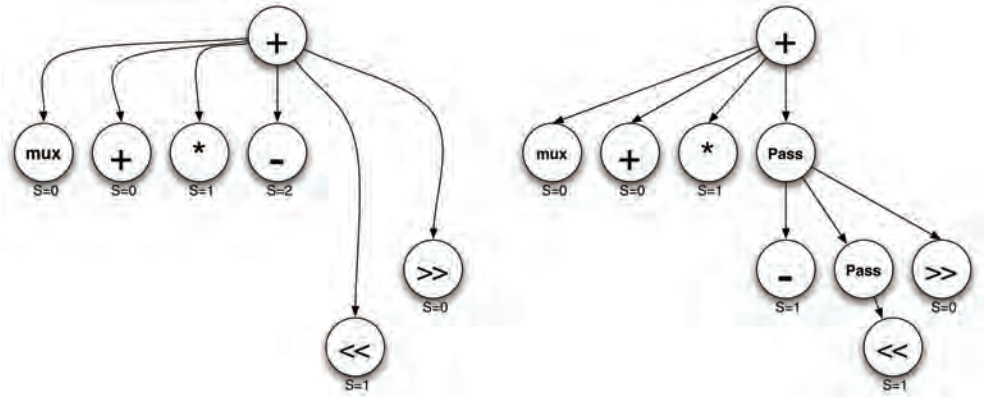
Algorithm 1 Deterministic Heuristic Mapping Algorithm

1: Create initial row assignments based on as-soon-as-possible layout.
2: $r \leftarrow 1$
3: **while** operators are assigned to row r **do**
4: Assign Columns in Row r (see Algorithm 2), possibly pushing some operators to later rows.
5: [pass-gate centering (see description in text).]
6: $r \leftarrow r+1$
7: **end while**

4.1 Row assignment

Initially, the row of each node is set to its row assignment in an as soon as possible (ASAP) "schedule" of the graph. Beginning with the first row and continuing downward until the last, each node in the given row is checked to determine if any of its children are non-immediate (i.e. the dependency edge in the DFG spans multiple rows) and as a result they cannot be placed in the next row. If any non-immediate children are present, a pass-gate is created with an edge from the current node. All non-immediate children nodes are disconnected from the current node and connected to the pass-gate. This ensures that after row assignment, there are no edges that span multiple rows of the fabric.

After handling the non-immediate children, each node is checked to determine if its fanout exceeds the maximum as defined by the FIM. If a node's fanout exceeds the limit, a pass-gate is created with an edge from the current node. In order to reduce the node's fanout, children nodes are disconnected from the current node and connected to the pass-gate. To minimize the number of additional rows that must be added to the graph we first move children nodes with the highest slack from the current node to the pass-gate. If the fanout cannot be reduced without moving a child node with a slack of zero, then the number of rows in the solution is increased by one causing an increase of one slack to all nodes in the graph. This process continues for each node in the current row, then subsequently for all rows in the graph as shown in Figure 7. Once row assignment is complete, the minimum fabric size for each benchmark is known. These minimum sizes are shown in Table 1.



(a) Before row assignment. (b) After row assignment.

Figure 7. Row assignment example showing pass-gate insertion.

	GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
Fabric Width	14	17	16	20	17	10	15
Fabric Height	18	16	13	12	10	9	8

Table 1. Minimum fabric sizes with no interconnect constraints.

4.2 Column assignment

The column assignment of the heuristic follows Algorithms 2–4 where items in square brackets [] are included in the optimized formulation. Many of these bracketed items are described in Section 4.3. During the column assignment for each row, the heuristic first determines viable locations based on the dependencies from the previous row. Then, the heuristic considers the impact of dependencies of nodes in the two following rows. The heuristic creates location windows that describe these dependencies as follows:

The *parent dependency window* (PDW) lists all FU locations that satisfy the primary constraint that the current node must be placed such that it can connect to each of its inputs (parents) with the interconnect specified in the FIM. The construction of the PDW is based on the location of each parent node, valid mapping locations due to the interconnect, and the operations supported by each FU (e.g. computational FU versus dedicated pass-gate). Figure 8 shows an example of a PDW dictated by the interconnect description. In this example, an operation that depends on the result of the subtraction in column 6 and addition in column 8 can only be placed in either `ALU 6` or `ALU 7` due to the restrictions of cardinality four interconnect.

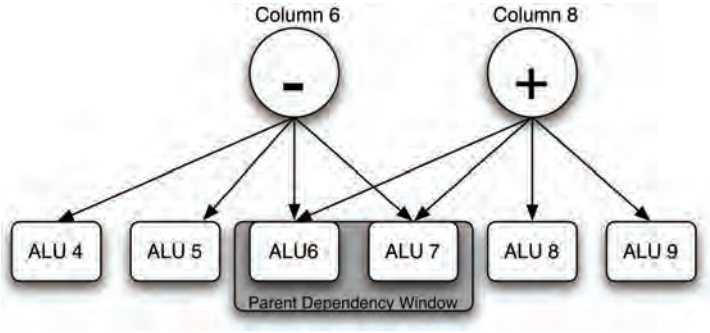


Fig. 8. Parent dependency window.

The *child dependency window* (CDW) lists all FU locations that satisfy the desired but non-mandatory condition that a node be placed such that each of its children nodes in the proceeding row will have at least one valid placement. The construction of the CDW is based on the PDW created from the potential locations of a current node as well as the PDW created from potential locations of any nodes that share a direct child with the current node. Nodes which share a direct child are referred to as *connected nodes*. Again the FIM is consulted to determine if there will be any potential locations for the children nodes based on the locations of the current node and connected nodes. A child dependency window example is shown in Figure 9. In this example, a left shift operation and a right shift operation are being assigned columns. Due to parent dependency window constraints, the

Algorithm 2 Assign Columns in Row r

```
1: Priority set ← ∅
2: while not all operators have column assignments do
3:    Calculate PDW, CDW, [and GDW] for all unplaced operators.
4:    For each unplaced operator, calculate FU Desirability [and Potential Connectivity] for each FU in its PDW.
5:    if there is a unary operator in the Priority set with empty PDW then
6:       Mapping has failed. Abort the process.
7:    else if there is a non-unary operator in the Priority set with empty PDW then
8:       Push it to the next row (r + 1).
9:       Create pass-gates for each of its inputs in the current row.
10:      Cancel all column assignments (for this row).
11:   else if there is an operator not in the Priority set with empty PDW then
12:      Add it to the Priority set.
13:      Cancel all column assignments (for this row).
14:   else
15:      Choose an operator i to map according to the following priorities: those in Priority set with |PDW| = 1 and
         smallest |CDW|, those in Priority set with smallest |CDW|, those not in Priority set with |PDW| = 1 and
         smallest |CDW|, those not in Priority set with smallest |CDW|. [Break any ties based on |GDW|.]
16:      Make Column Assignment for Operator i (see Algorithm 3).
17:   end if
18: end while
```

left shift can be placed in either `ALU 10` or `ALU 11`. Similarly, the right shift can be placed in either `ALU 6` or `ALU 7`. There is a third node (not pictured) which takes its inputs from the two shift operations. In order for this shared child to have a valid placement, the left shift must be placed in `ALU 10` and the right shift must be placed in `ALU 7`. Using this placement, the shared child will have a single possible placement in its PDW, `ALU 8`.

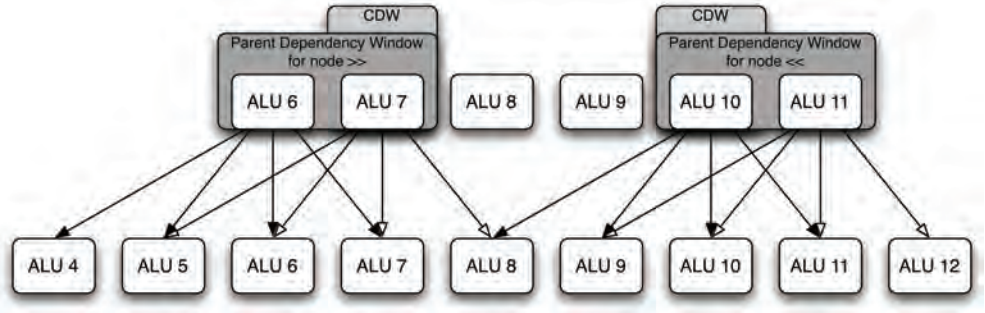


Fig. 9. Child dependency window.

The grandchild dependency window (GDW) provides an additional row of look-ahead. The GDWlists all FU locations that satisfy the optional condition that a node be placed such that children nodes two rows down (grandchildren) will have at least one valid placement. It is constructed using the same method as the CDW.

As nodes are mapped to FU locations, newly taken locations are removed from the dependency windows of all nodes (since no other node can now take those locations), and the child and grandchild windows are adjusted to reflect the position of all mapped nodes. In addition to tracking the PDWs, CDWs, and GDWs of each node, a desirability value is associated with each location in the current row. The desirability value is equal to the number of non-mapped nodes that contain the location in their PDW, CDW, or GDW.

Algorithm 3 Make Column Assignment for Operator i

```
if |PDW| = 1 then
    Assign to the unique column in PDW.
else {|PDW| > 1}
    if |CDW| = 1 then
        Assign to the unique column in CDW.
    else if |CDW| > 1 [and no shared grandchildren] then
        Assign to the column in the CDW with lowest FU Desirability, [ties broken by highest Potential Connectivity,
        further ties broken by lowest Distance to Center.]
    else if |CDW| > 1 and shared grandchildren, or shared grandchildren but no shared children then
        [Use Algorithm 4 to determine column assignment.]
    else if shared children but |CDW| = 0 then
        [Compute Nearness Measure for each FU in PDW based on common children.]
        [Assign to the column with the highest Nearness Measure.]
    else if i does not share children [or grandchildren] with other operators then
        Assign to the column in the PDW with lowest FU Desirability.
    end if
end if
```

Algorithm 4 Assignment Based on Shared Grandchildren

```
if |GDW| = 1 then
    Assign to the unique column in GDW.
else if |GDW| > 1 then
    Assign to the column in the GDW with highest Potential Connectivity, ties broken by lowest FU Desirability,
    further ties broken by lowest Distance to Center.
else if |GDW| = 0 then
    Compute Nearness Measure for each FU in CDW based on common grandchildren.
    Assign to the column with highest Nearness Measure.
end if
```

The mapper then places each node one at a time. To select the next node to place, the mapper first checks for any nodes with an empty PDW, then for any nodes with a PDW that contains only one location. Then it checks for any high-priority nodes in the current row, as these are nodes designated as difficult to map. Finally, it selects the node with the smallest CDW, most connected nodes, and lowest slack. This node is then placed within the overlapping windows while attempting to minimize the negative impact to other nodes.

Column placement also uses the concept of a *priority set*. In the process of placing operators, the algorithm may find that an operator has become impossible to place. If this happens, the algorithm is placed into the priority set and column placement for the row is restarted. Operators in the priority set are placed first. Even then, it may be impossible to place some operators. The last resort for the algorithm is to reassign the operator to the next row and add pass-gates to the current row for the operator's inputs. Unary operators cannot be reassigned because placing the pass-gate for the input would also be impossible. If a unary operator (or pass-gate) in the priority set cannot be placed, then the algorithm aborts.

4.3 Extensions

The initial algorithm was not always able to produce high quality mappings for some of the benchmarks when using more restrictive interconnects such as 5:1. Several extensions to the heuristic were implemented in an effort to increase its effectiveness.

Potential Connectivity: When determining the location to place an operator we consider which locations provide the most potential connectivity to child operators.

Potential connectivity is defined as the number of locations each shared child operation could be placed when the current operation is placed in a particular location.

Nearness Measure: This measure is used when an operator has shared children but the CDW is empty. The goal is to push the operators which share a child as close together as possible; this allows the algorithm to eventually place the child operators in some later row. The measure is the sum of the inverses of the distances from the candidate FU to the operators with common children.

Distance to Center: Used as a tie-breaker only to prefer placing operators closer to the center of the fabric.

Pass-gate centering: The initial algorithm tended to push pass-gates that have no shared child operators toward the edges of the fabric. This makes it harder for their eventual non-pass-gate descendants to be mapped, since their pass-gate parent is so far out of position. After placing an entire row the mapper pushes pass gates toward the center by moving them into unassigned FUs. This is the extension shown in Algorithm 1.

4.4 Results

Higher cardinality interconnects such as 8:1 and higher were easily mapped using the deterministic greedy algorithm. We show results using a 5:1-based interconnect as it exercised the algorithm well. The mapper was tested on seven signal and image processing benchmarks from image and signal processing applications. A limit of 50 rows was used to determine if an instance was considered un-mappable with the given algorithm. Mapping quality was judged on three criteria. The first is fabric size, represented in particular by the number of rows in the final solution. The second is total path length, or the sum of the paths from all inputs to all outputs as described in Section 2.2. The third metric is mapping time, which is the time it takes to compute a solution.

The fabric size is perhaps the most important factor in judging the quality of a solution. The number of columns is more or less fixed by the size of the largest row in a given application. However, the number of additional rows added to the minimum fabric heights listed in Table 1 reflects directly on the capability of the mapping algorithm. Smaller fabric sizes are desirable because they require less chip area, execute more quickly, and consume less power.

As described in Section 2.2, the total path length increase is a key factor in the energy consumption of the fabric executing the particular application. However, fabric size and total path length are related. A mapping with a smaller fabric size will typically have a considerably smaller total path length and thus, also have a lower energy consumption. Thus, the explicit total path length metric is typically most important when comparing mappings with a similar fabric size.

The mapping time is important because it evaluates practicality of the mapping algorithm. Thus, the quality of solution of various mapping algorithms is traded off against the execution time of the algorithms when comparing these mapping algorithms.

We compared two versions of the greedy algorithm. The initial algorithm makes decisions based on the PDW and the CDW and uses functional unit desirability to break ties. This heuristic is represented by Algorithms 1–3 without the sections denoted by square brackets []. The final version of the algorithm is shown in Algorithms 1–4 including the square

bracket [] regions. This version of the heuristic builds upon the initial algorithm by including the GDW, potential connectivity, and centering. The results of the comparison are shown in Table 2.

		GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
Initial Algorithm	Rows Added	3	13	11	*no solution*	*no solution*	1	2
	Time (s)	18	79	9	37	15	<1	<1
Final Algorithm	Rows Added	1	5	1	8	4	1	2
	Time (s)	<1	12	<1	1	<1	<1	<1

Table 2. Number of rows added and mapping times for the greedy heuristic mapper using a 5:1 interconnect.

Using the initial algorithm, Sobel, Laplace, and GSM can be solved fairly easily, requiring only a few added rows in order to find a solution. However, the solutions for ADPCM Encoder and Decoder require a significant number of additional rows and both IDCT-based benchmarks were deemed unsolvable.

The final algorithm is able to find drastically better solutions more quickly. For example the number of rows added for ADPCM Encoder and Decoder went from 13 to 5 and 11 to 1, respectively. It is also able to find feasible solutions for IDCT Row and IDCT Column. For the other four benchmarks, the final algorithm performs equally well or better than the initial algorithm. The final algorithm is faster in every case decreasing the solution time for all benchmarks to within 1 second except ADPCM Encoder which was reduced from 79 to 12 seconds.

We tried the final deterministic algorithm on a variety of more restrictive interconnects including a cardinality five interconnect with every third FU (33%)replaced with a dedicated pass-gate. The results are shown in Table 3. The fabric size results are actually quite similar in terms of rows added to the 5:1 cardinality interconnect without dedicated pass-gates.

		GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
Final Algorithm	Rows Added	1	6	2	7	5	0	2
	Time (s)	<1	7	<1	2	1	<1	<1

Table 3. Greedy heuristic mapper results using a 5:1 interconnect and 33% dedicated pass-gates.

While the deterministic heuristic provides a fast valid mapping, it does add a considerable number of rows from the ASAP (optimal) configuration, which leads to considerable path length increases and energy overheads. In the next section we explore a technique to improve the quality of results through an iterative probabilistic approach.

5. Greedy heuristic including randomization

Another flavor of greedy algorithms are greedy randomized algorithms. Greedy randomized algorithms are based on the same principles guiding purely greedy algorithms, but make use of randomization to build different solutions on different runs (Resende & Ribeiro, 2008b). These algorithms are used in many common meta-heuristics such as local

search, simulated annealing, and genetic algorithms (Resende & Ribeiro, 2008a). In the context of greedy algorithms, randomization is used to break ties and explore a larger portion of the search space. Greedy randomized algorithms are often combined with multi-iteration techniques in order to enable different paths to be followed from the same initial state (Resende & Ribeiro, 2008b).

The final version of the deterministic greedy algorithm is useful due to its fast execution time and the reasonable quality of its solutions. However, because it is deterministic it may get stuck in local optimums which prevent it from finding high quality global solutions. By introducing a degree of randomization into the algorithm, the mapper is able to find potentially different solutions for each run. Additionally, since the algorithm runs relatively quickly, it is practical to run the randomized version several times and select the best solution.

The column assignment phase of the mapping algorithm was chosen as the logical place to introduce randomization. This area was selected as the column assignments not only affect the layout of the given row, but also affect the layouts of subsequent rows. In the deterministic algorithm, nodes are placed in an order determined by factors including smallest PDW, CDW, GDW, etc. and once placed, a node cannot be removed. In contrast, the randomized heuristic can explore random placement orders, which leads to much more flexibility.

We investigated two methods for introducing randomization into the mapping heuristic. The first approach makes ordering and placement decisions completely randomly. We describe this approach in Section 5.1. The second leverages the information calculated in the deterministic greedy heuristic by applying this information as weights in the randomization process. Thus, the decisions are encouraged to follow the deterministic decision but is allowed to make different decisions with some probability. We describe this approach in Section 5.2.

5.1 Randomized heuristic mapping

The biggest difference between the deterministic heuristic and the heuristics that incorporate randomization is that the deterministic is run only once and the random oriented heuristics are run several times to explore different solutions. The basic concept of the randomized heuristic is shown in Algorithm 5. First the deterministic algorithm is run to determine the initial "best" solution. Then the randomizer mapper is run a fixed number of times determined by I. If an iteration discovers a better quality solution (better height or same height and better total path length) it is saved as the new "best" solution. This concept of saving and restoring solutions is common in many multi-start meta-heuristics, including simulated annealing and greedy randomized adaptive search procedures (GRASP) (Resende & de Sousa, 2004).

Algorithm 5 Randomized Heuristic Mapping Algorithm

1: Execute deterministic heuristic mapper (Algorithm 1) to create solution S and determine height H and total path length P
2: **for** I iterations **do**
3: Execute random heuristic mapper to determine solution s and calculate height h and total path length p
4: **if** $h < H$ or $(h = H$ and $p < P)$ **then**
5: $S \leftarrow s, H \leftarrow h, P \leftarrow p$
6: **end if**
7: **end for**

The randomized mapping heuristic follows the same algorithmic design as the deterministic heuristic from Algorithm 2. The only major change is to line 15, in which the new algorithm selects the next node to map in a column randomly and ignores all other information.
Although the introduction of randomization allows the mapper to find higher quality solutions, it also discovers many lower quality solutions, which often take a long time to complete. In order to mitigate this problem, one other divergence from the deterministic algorithm allows the mapper to terminate a given iteration once the fabric size of the current solution becomes larger than the current best solution.

5.2 Weighted randomized heuristic mapping

Using entirely random placement order did discover better solutions (given enough iterations) than the deterministic heuristic. Unfortunately, the majority of the solutions discovered were of poorer quality than the deterministic approach. Thus, we wanted to consider a middle ground algorithm that was provided some direction based on insights from the deterministic algorithm but also could make other choices with some probability. This resulted in a weighted randomized algorithm.
Weights are calculated based on the deterministic algorithm concepts of priorities and dependency windows. Again the modification of the basic deterministic algorithm to create the weighted randomized algorithm is based on line 15 of Algorithm 2. The weighted randomized algorithm replaces this line with Algorithm 6 to select the next node to place. The algorithm begins by dividing the unplaced operators into sets based on their PDW size. Each set is then assigned a weight by dividing its PDW size by the sum of all of the unique PDW sizes. Because nodes with small parent dependency windows are more difficult to place, it is necessary to assign them a larger weight. This is accomplished by subtracting the previously computed weight from one. Each set is then further subdivided in a similar fashion based first on CDW sizes and then node slack. The result of this operator grouping process is a weighted directed acyclic graph (DAG) with a single vertex as its root. Starting at the root, random numbers are used traverse the weighted edges until a leaf vertex is reached, at which point an operator will be selected for column assignment.

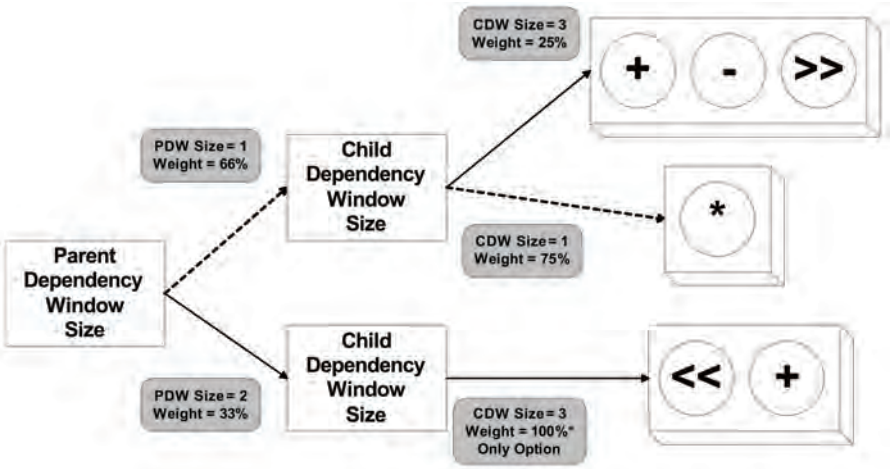


Fig. 10. Heuristic weight system.

An example of the weighting system used in the randomized mapper is shown in Figure 10. In this example, nodes priorities are assigned based on PDW size followed by CDW size. Slack is not considered in this example for simplicity. The deterministic heuristic would always assign the highest priority to the multiplier because it has the smallest parent window as well as the smallest child window. In Figure 10 this behavior is indicated by the dashed arrows. By introducing probability into the heuristic, the multiplier is still given the highest priority with a 50% chance of being selected. However, a node from the top group has a 17% chance of being selected and a node from the bottom group has a 33% chance of being selected instead. In the event that several nodes are assigned the same priority level, one node is chosen randomly with equal weight from the set.

Algorithm 6 Weighted Randomized Heuristic Operator Selection Algorithm

```
1: sum_p ← 0
2: for each unplaced operator OP do
3:   if W_|PDW| is ∅ then
4:     sum_p ← sum_p + |PDW|
5:   end if
6:   add OP to W_|PDW|
7: end for
8: Assign normalized weights based on formula 1 − (|PDW| ÷ sum_p)
9: for each key i in W do
10:  sum_c,i ← 0
11:  for each operator OP in W_i do
12:    if W_i,|CDW| is ∅ then
13:      sum_c,i ← sum_c,i + |CDW|
14:    end if
15:    add OP to W_i,|CDW|
16:  end for
17:  Assign normalized weights based on formula 1 − (|CDW| ÷ sum_c,i)
18: end for
19: for each key i in W do
20:  for each key j in W_i do
21:    sum_s,i,j ← 0
22:    for each operator OP in W_i,j do
23:      if W_i,j,slack is ∅ then
24:        sum_s,i,j ← sum_s,i,j + slack
25:      end if
26:      add OP to W_i,j,slack
27:    end for
28:    Assign normalized weights based on formula 1 − (slack ÷ sum_s,i,j)
29:  end for
30: end for
31: Select W_i based on weighted random selection of i
32: Select W_i,j based on weighted random selection of j
33: Select W_i,j,k based on weighted random selection of k
34: Select a node randomly from W_i,j,k for placement
```

5.3 Mapper early termination

The randomized and weighted randomized mappers require significantly longer run times than the deterministic heuristic due to their multiple iterations. Additionally, the runtime of these algorithms is hampered by another effect. For any given row, it is possible that the mapper will not be able to place all of the nodes. When this happens, the mapper will start

the row over, attempting to place the problem node(s) first, this is shown in Algorithm 2 lines 11–13. We call each occurrence of this behavior a *row placement failure*. The random oriented algorithms tend to have iterations that have a large volume of row placement failures, exacerbating run times.

To limit the runtime overheads from row placement failure, during the deterministic mapper run we record the number of row placement failures across all of the rows. The randomized versions of the mapper then use this n value as an upper limit on the number of total row placement failures permitted per iteration. Once this limit of n row placement failures is eclipsed, the mapper aborts and moves on to the next iteration.

5.4 Results

Table 4 compares the fabric size, path length increase, and mapping time for the deterministic, randomized, and weighted randomized mappers. The two iterative mappers were run for 500 iterations each. In order to further gauge the performance of the randomized and weighted randomized mappers, the average mapping time per iteration is also reported. The completely randomized mapper outperformed or equaled the deterministic mapper in terms of fabric size and total path length for all of the benchmarks. The weighted randomized mapper was able to find significantly better solutions than both of the other mappers. The weighted randomized mapper was also, on average, 48% faster than the randomized mapper.

		GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
Deterministic Algorithm	Rows Added	1	5	1	8	4	1	2
	Path Length Increase	2	11	1	42	26	1	2
	Total Time (s)	< 1	12	< 1	1	< 1	< 1	< 1
Random Algorithm	Rows Added	0	5	0	6	3	0	0
	Path Length Increase	0	7	0	34	18	0	0
	Total Time (s)	978	5,109	1,378	3,006	1,169	51	75
	sec/iteration	2	10	3	6	2	< 1	< 1
Weighted Algorithm	Rows Added	0	0	0	3	1	0	0
	Path Length Increase	0	1	0	25	9	0	0
	Total Time (s)	366	4,019	394	920	549	36	55
	sec/iteration	< 1	8	< 1	2	1	< 1	< 1

Table 4. Deterministic, randomized, and weighted randomized mapper comparison using a 5:1 interconnect.

Table 5 illustrates the effects of implementing the early termination mechanism. Once again, the random and weighted randomized mappers were run for 500 iterations. The randomized mapper was still able to outperform the deterministic mapper in some cases, but not by the same margin as before. The reduction in execution time was roughly 38% for the randomized mapper. The weighted randomized mapper still performed well, only requiring one additional row to be added in IDCT Row. The weighted randomized mapper, which was already faster than the randomized mapper, saw a 12% improvement in terms of mapping time.

The randomized and weighted randomized algorithms were also tested using the 5:1 interconnect with 33% of the functional units replaced by dedicated pass-gates. These results are presented in Table 6. Again the random and weighted randomized mappers were run for 500 iterations with early termination enabled. As with the basic 5:1 interconnect, the randomized mapper performed as well or better than the deterministic mapper in terms of

		GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
Deterministic Algorithm	Rows Added	1	5	1	8	4	1	2
	Path Length Increase	2	11	1	42	26	1	2
	Total Time (s)	< 1	12	< 1	1	< 1	< 1	< 1
Random Algorithm	Rows Added	0	4	0	8	4	0	0
	Path Length Increase	0	9	1	42	26	0	0
	Total Time (s)	561	4,262	338	1,148	781	40	63
	sec/iteration	1	9	< 1	2	2	< 1	< 1
Weighted Algorithm	Rows Added	0	0	0	3	2	0	0
	Path Length Increase	0	1	0	22	18	0	0
	Total Time (s)	327	3,803	216	731	517	36	56
	sec/iteration	< 1	8	< 1	1	1	< 1	< 1

Table 5. Deterministic, randomized, and weighted randomized mapper comparison using a 5:1 interconnect and early termination.

fabric size and path length increase. However, the weighted randomized mapper was superior to both of the other mappers. Since this interconnect contains dedicated pass-gates, the number of FUs utilized as pass-gates are also included in the results. A similar trend was observed where the weighted randomized mapper performed the best (fewest FUs used as pass-gates), followed by the randomized mapper and the deterministic mapper.

		GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
Deterministic Algorithm	Rows Added	1	6	2	7	5	0	2
	Path Length Increase	2	13	3	37	33	0	2
	ALUs as Pass-gates	75	116	37	77	47	2	3
	Total Time (s)	< 1	7	< 1	2	1	< 1	< 1
Random Algorithm	Rows Added	1	6	2	7	3	0	1
	Path Length Increase	2	13	3	37	27	0	1
	ALUs as Pass-gates	75	116	37	77	39	2	5
	Total Time (s)	500	5,578	331	2,052	949	42	100
	sec/iteration	1	11	< 1	4	2	< 1	< 1
Weighted Algorithm	Rows Added	0	2	0	5	3	0	2
	Path Length Increase	0	4	1	44	23	0	2
	ALUs as Pass-gates	74	83	30	65	36	2	3
	Total Time (s)	367	3,989	233	1,397	618	42	92
	sec/iteration	< 1	8	< 1	3	1	< 1	< 1

Table 6. Deterministic, randomized, and weighted randomized mapper comparison using a 5:1 interconnect, 33% dedicated passgates, and early termination.

6. Sliding partial MILP heuristic

The sliding partial MILP heuristic is a greedy approach to solve the augmented fixed rows mapping problem. As discussed in Section 2.1.2, we created a MILP that solved the feasible mapping with fixed rows problem for the entire device in a single formulation but the run times were prohibitively long (hours to days). However, an MILP that solves a limited scope of the mapping problem can run much faster (seconds). Thus, the partial sliding MILP heuristic creates the middle ground between the greedy heuristics and the full MILP

formulation. It has similarities to the deterministic and randomized heuristics from Sections 4 and 5 in that it follows a top to bottom approach, and rows that have been visited cannot be adjusted. However, while these earlier heuristics place a single node at a time, the sliding approach uses an MILP for an optimal solution for an entire row or multiple rows in one step. Thus, the sliding heuristic, while still greedy, has a much larger scope than the earlier greedy algorithms. Pseudocode for the sliding partial MILP heuristic is shown in Algorithm 7.

Algorithm 7 Sliding Partial MILP Heuristic

```
1:  while there are violated edges do
2:    Find a violated edge (highest row).
3:    Solve a partial MILP to fix the violation(s) or to push the violation(s) down.
4:    if the violation(s) is not fixed and cannot be pushed down then
5:      if the number of pass-gates < limit then
6:        Add a row of pass-gates.
7:      else
8:        Exit (heuristic cannot generate a mapping).
9:      end if
10:   end if
11: end while
```

The heuristic starts with an "arbitrary placement" where operations are placed in the earliest row possible (ASAP) and the operations are left justified. The heuristic follows a top-down approach and continues until it fixes all of the violations. We define violations as edges connecting FUs between rows that cannot be realized using the routing described in the FIM. When a violated edge is located, a window of rows is selected as shown in Figure 11. Within this window, an MILP is solved to attempt to correct the violations by moving the column locations of the nodes. We call this a partial MILP as it only solves the mapping problem for part of the fabric (i.e. the limited window). Because the heuristic takes a top-down approach, any previously existing violations above the MILP window should have already been corrected. However, it is possible that violations exist below the window. Selection of the window of rows is discussed in Section 6.2.

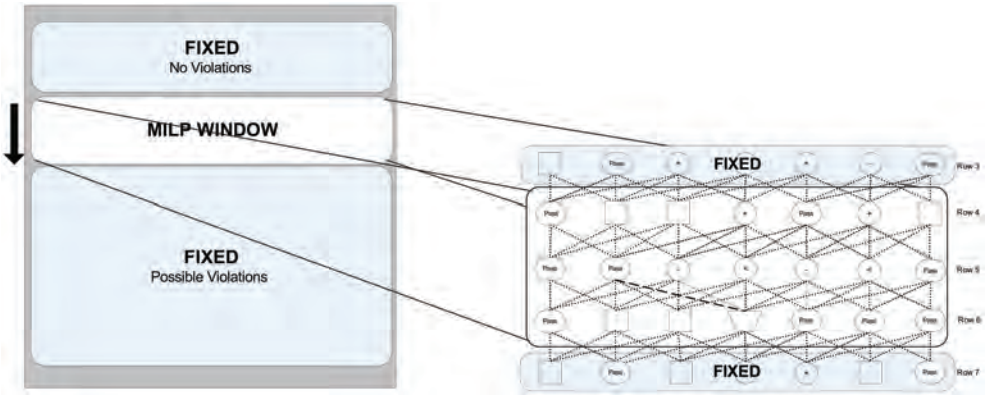
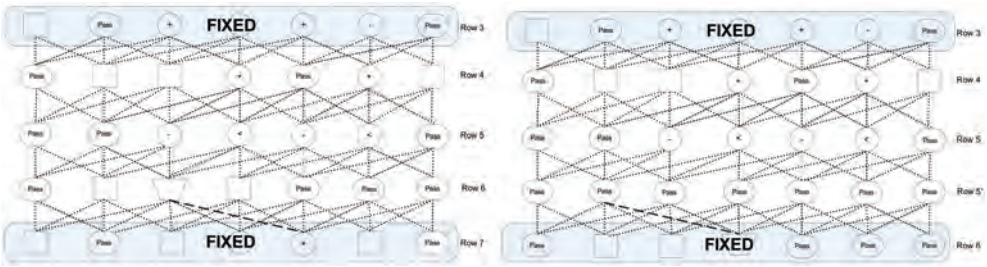


Fig. 11. Nearly feasible solution.

There are three possibilities at the end of the run on a particular window:
Case 1: The partial MILP fixes the violation(s).

Case 2: The partial MILP cannot fix the violation(s), but pushes the violation(s) down in the window so that subsequent windows may be able to find a solution. This case is shown in Figure 12(a) where the violating edge, represented by the bold dashed line between rows 5 and 6 in Figure 11, has been pushed down between rows 6 and 7.

Case 3: If the partial MILP cannot fix or push down the violation then a row of pass-gates is added to increase the flexibility, and the partial MILP is run again. This is illustrated with the addition of row 5′ containing all pass-gates in Figure 12(b).



(a) Violation from Figure 11 is pushed down.

(b) A row of pass-gates is added to the window from Figure 11.

Fig. 12. Resolutions when the partial MILP cannot eliminate a violation.

6.1 Partial MILP formulation

The heuristic generates valid mappings by using small, fast MILPs on the selected rows (Figure 11). The partial MILP formulation, parameters, sets, and variables used in the heuristic are described below:

The objective function (0) minimizes the number of edges used that violate the interconnect design. Constraint (1) ensures that an edge can be located in only one place. Constraint (2) ensures that an operator can be placed in only one column. Constraint (3) states that there can be at most one operator in a column for a given row k. The final two constraints relate the operator, x, and edge, z, variables. Constraint (4) states that edge (i, t) can only be placed starting at column j if operator i is at column j. Constraint (5) states that edge (i, t) can only be placed to end at column k if the ending operator t is at column k. We fix the locations of the operators in the first and last row of the MILP window by setting $x_{i\,j} = 1$ where i is the operator and j is the column.

Parameters:

r: Number of rows

c: Number of columns

$p_{i,t,\,j,k}$: Objective coefficients

a_i: Index of the first operator in row i

Sets:

C: Set of columns (1 to c)

V: Set of operators in the MILP window

E: Set of edges in the MILP window

R: Set of rows in the MILP window (1 to r)

Variables:

$x_{i\,j}$: Binary variable for operator assignment. If operator i is in column j, then $x_{i\,j} = 1$, otherwise it is 0.

$z_{i,t,\,j,k}$: Binary variable for edge assignment. If starting operator i is in column j and ending operator t is in column k, then $z_{i,t,\,j,k} = 1$, otherwise it is 0.

Partial MILP formulation:

$$\begin{aligned}
& \min \quad \sum_{(i,t)\in E}\sum_{j\in C}\sum_{k\in C} p_{i,t,j,k} z_{i,t,j,k} && && (0)\\
s.t. \quad & \sum_{j\in C}\sum_{k\in C} z_{i,t,j,k} = 1 && \forall (i,t)\in E && (1)\\
& \sum_{j\in C} x_{ij} = 1 && \forall i \in V && (2)\\
& \sum_{i\in V, a_k \le i < a_{k+1}} x_{ij} \le 1 && \forall k \in R, j \in C && (3)\\
& \sum_{k\in C} z_{i,t,j,k} \le x_{ij} && \forall (i,t)\in E, j\in C && (4)\\
& \sum_{j\in C} z_{i,t,j,k} \le x_{tk} && \forall (i,t)\in E, k\in C && (5)\\
& x_{ij} \ge 0 && \forall i \in V, j \in C && (6)\\
& z_{i,t,j,k} \ge 0 && \forall (i,t)\in E, j\in C, k\in C && (7)
\end{aligned}$$

The effect of "pushing a violation down" when solving the MILP can be achieved with proper objective coefficients ($p_{i,t,\,j,k}$) in the formulation. The objective coefficients for the violations in the upper rows are much higher than the ones in the lower rows. For example, assume there is a violation between rows 5 and 6, and that the MILP window size is three rows as shown in Figure 11. In other words, the columns of the operators in rows 4, 5, and 6 can be adjusted while the locations of the operators in rows 3 and 7 are fixed. The objective coefficients are 10,000 for edges between rows 3 and 4 and between rows 4 and 5, 100 for the edges between rows 5 and 6, and 1 for the edges between rows 6 and 7. This avoids having violations in the higher rows, and may push the violation down to the rows between 6 and 7. Thus, objective values > 10,000 show an improvement. Of course, if a objective value of 0 is reached, then all violations are eliminated.

6.2 Determining window size

In the sliding partial MILP heuristic, a window of rows is selected for optimization. We tested different alternatives of numbers of rows to optimize in this window and used a cardinality five interconnect as the target for the tests. The more rows that are optimized simultaneously, the longer the MILP takes. However, it may not be possible to solve the violations if too few rows are included in the optimization window. We consider window sizes from one to five rows, as well as some approaches that change the number of rows. We did not exceed five rows since it started to take too long to solve the MILP.

Optimizing a single row is a special case. Since all of the variables are binary and the locations of the operators in other rows are fixed, this formulation can be solved directly as a Linear Program (LP), which can be solved much more efficiently than an MILP. Since a violated edge connects two rows, either of these rows can be targeted for optimization. Thus, we attempt to optimize the top row first, and if unsuccessful, attempt to optimize the bottom row. Unfortunately, the single row method was not capable of solving the majority of the benchmarks. Using such a small window size often resulted in the LP being unable to find a feasible solution for a given row. In these cases, rows of pass-gates would be continuously added in the same location and the algorithm was unable to make progress.

When optimizing two rows, the window would contain the two rows connected by the violating edge. Additionally, we attempted to correct the violations by optimizing previous rows. For example, if there is a violation between rows 5 and 6, first rows 4 and 5 are optimized. If the violation is not fixed, rows 5 and 6 are optimized. This example is shown in Figure 13. Unfortunately, most of the benchmarks also could not be solved by using a window size of two.

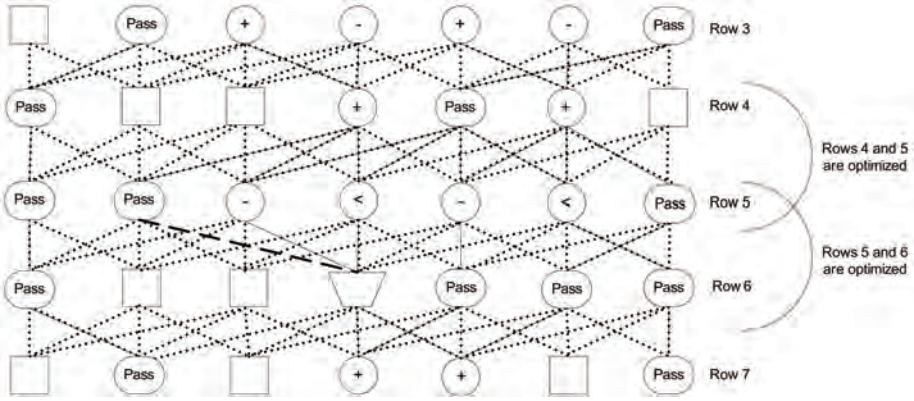


Fig. 13. Optimization of two rows.

Next, a window size of three was tested as shown in Figure 11. In this example, there is a violation between rows 5 and 6. The MILP fixes the operators in rows 3 and 7, while attempting to optimize the operators in rows 4, 5, and 6. This approach was able to solve five of the seven benchmarks. The other two benchmarks could not be solved because they each have an operator with grandparents that are prohibitively far from each other. This case could not be solved by our algorithm even by adding rows of pass-gates.

Having grandparents or great-grandparents far from each other causes problems. To solve this problem we introduced graded objective coefficients. The farther the violation is, the more it will cost. Based on this idea, the objective coefficients are multiplied by the absolute distance difference between the column numbers of the edges' operators. Thus, operators with the same grandchildren are more likely to placed close to each other. Even if the grandparents or great-grandparents are far from each other, violations can be fixed with the graded objective function by adding enough rows of pass-gates. After adding this feature, "graded optimize three" can solve all of the instances. However, it adds seven rows of pass-gates for one of the benchmarks.

Optimizing three rows is successful because rows of pass-gates are utilized efficiently. When a row of pass-gates is added, the locations of the operators in the preceding and succeeding rows can be adjusted. In contrast, when two rows are optimized, only the locations of the row of pass-gates and another row can be adjusted, which does not always help. In other words, sometimes the violation cannot be fixed and rows of pass-gates are constantly added between the same rows.

The four row optimization approach performs well. All of the instances were solved by adding at most five rows of pass-gates. Using the increased window size allows rows of pass-gates to be utilized more efficiently than in the three row optimization scenario. Finally, optimizing five rows can solve all of the benchmarks by adding two rows at most. However, the solution times are significantly longer than those of the "optimize four rows" version.

Based on the tests "graded optimize four" is chosen as the best option. Optimizing a window of four rows does not take long. Additionally, it does not add too many rows of pass-gates since the rows of pass-gates are utilized efficiently.

6.2.1 Dedicated pass-gates

Based on the "graded optimize four" window size, the same window size was applied to interconnects with dedicated pass-gates. While the heuristic was successful with more relaxed interconnects (e.g. 8:1 cardinality with various percentages of dedicated pass-gates) the results for the more restrictive 5:1 cardinality interconnect with dedicated pass-gates led to several infeasible solutions. This is a limitation of the algorithm operating on the entire row and not being able to move individual operations into different rows like the deterministic and randomized heuristics.

6.3 Extensions

To improve the quality of the partial MILP heuristic, we explored some logical extensions. In the next two subsections, respectively, we describe an iterative method to improve the runtime and retain similar quality of results and a premapping step to potentially improve the quality of results.

6.3.1 Iterative approaches

Solving partial MILPs with smaller window sizes is much faster but is less effective at removing violations than larger window sizes. Thus, in the iterative approach we use variable sized windows starting with small window sizes and escalating to larger window sizes if necessary. Thus, the window size is increased if the violation(s) cannot be fixed or pushed down with the current size. For instance, in "iterative 1234" first one row is optimized. If the violation(s) cannot be removed, the window size is increased to two rows and the MILP is run again. This continues through a window size of four rows. If the MILP cannot be solved for four rows, a row of pass-gates is added. These iterative approaches perform well and are competitive with the "optimize four rows" version.

6.3.2 Two-pass sliding partial MILP heuristic

We discovered that the sliding partial MILP heuristic can be more effective if it starts with a nearly feasible solution when compared with an arbitrary solution. Thus, we created a two-pass extension of the sliding partial MILP heuristic. The one-pass heuristic sometimes requires adding a row of pass-gates to fix violations. Thus, in the first pass of the two-pass heuristic, the option to add a row of pass-gates is removed and this pass runs partial MILPs to minimize the number of violated edges. However, some violations may remain. We used this pass on the arbitrary solutions to create better starting points for the sliding partial MILP heuristic (i.e. the second pass). We tested this heuristic approach for one, two, and three row windows, respectively for the first pass and "graded optimize four rows" in the second pass.

6.4 Results

This section presents the results of tests on the sliding partial MILP heuristic for 5:1 cardinality interconnects for both the one-pass and two-pass instantiation. Table 7 summarizes the number of rows added and the run times for the heuristics "optimize one row," "optimize two rows," "optimize three rows," "graded optimize three rows," "optimize four rows," "graded optimize four rows," and "graded optimize five rows" starting from an arbitrary solution. The "optimize one row" and "optimize two rows" methods only solve Sobel and Laplace, the two smallest benchmarks. The other benchmarks cannot be solved even after adding 20 rows of pass-gates. The "optimize three rows" method solves five out of seven benchmarks. The "graded optimize three rows" approach

solves all of the instances. The longest run time for "graded optimize three rows" is 102 seconds and it adds at most seven rows. The "graded optimize four rows" solutions are always as good as the "optimize four rows" solutions in terms of fabric size. For ADPCM Encoder and both of the IDCT benchmarks, "graded optimize four rows" adds fewer rows than the "optimize four rows" approach. There are not significant differences in the run times. Even though arbitrary solutions are used as the starting points, the run times are not significantly long. IDCT Column has the longest runtime of approximately ten minutes. On the other hand, "graded optimize five rows" adds fewer rows than the other heuristics but at the price of much longer run times. It adds at most two rows of pass-gates, however, the run times are long enough to not be a practical option in many cases.

		GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
Optimize	Rows Added	> 20	> 20	> 20	> 20	> 20	4	3
One Row	Time (s)	-	-	-	-	-	< 1	< 1
Optimize	Rows Added	> 20	> 20	> 20	> 20	> 20	3	3
Two Rows	Time (s)	-	-	-	-	-	4	3
Optimize	Rows Added	2	> 20	3	6	9	0	> 20
Three Rows	Time (s)	24	-	37	88	88	4	-
Graded Optimize	Rows Added	1	3	2	7	2	0	1
Three Rows	Time (s)	31	40	31	102	22	5	95
Optimize	Rows Added	0	3	0	6	2	0	0
Four Rows	Time (s)	43	152	150	674	343	3	12
Graded Optimize	Rows Added	0	2	0	5	1	0	0
Four Rows	Time (s)	80	143	32	635	373	4	87
Graded Optimize	Rows Added	0	1	0	2	1	0	0
Five Rows	Time (s)	258	318	34	1,598	5,721	15	159

Table 7. Tests on arbitrary instances.

The results for the iterative "graded 1234," "graded 234," and "graded 34" heuristics are shown in Table 8. The "iterative graded 1234" and "iterative graded 234" heuristics behave similarly since optimizing one row rarely eliminates violations. They add at most six rows. When comparing the "graded optimize four rows" and "iterative graded 34" heuristics, "graded optimize four" is better for GSM while "iterative graded 34" is better for IDCT Column in terms of the number of rows added.

Based on all of the computational tests, graded objective coefficients helped to find better mappings in terms of rows added and mapping time. The "graded optimize four rows" and "iterative graded 34" approaches were found to be the best of this group. Thus, to examine the heuristic extensions we will retain the "graded optimize four rows" method for the remaining tests.

		GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
Iterative	Rows Added	1	3	0	6	3	0	2
Graded 1234	Time (s)	18	109	37	209	401	3	12
Iterative	Rows Added	1	3	0	6	3	0	2
Graded 234	Time (s)	17	103	37	195	398	2	12
Iterative	Rows Added	1	2	0	3	1	0	0
Graded 34	Time (s)	40	93	43	242	48	7	90

Table 8. Tests of iterative versions on arbitrary instances.

In the two-stage sliding partial MILP heuristic, a nearly feasible solution could be found in the first stage using either "heuristic one row," "heuristic two rows," or "heuristic three rows." Table 9 summarizes the number of rows added and the run times of the two-stage heuristic, with the run times separated into time for the first stage and second stage. The run times of the first stage are less than one second for "heuristic one row," at most ten seconds for "heuristic two rows," and less than two minutes for "heuristic three rows." Starting the sliding partial MILP heuristic from solutions found using "heuristic one row" is not much better than starting from an arbitrary solution. However, "heuristic two rows" and "heuristic three rows" solutions provide more benefit in the first stage. The sliding partial MILP heuristic starting from "heuristic two rows" or "heuristic three rows" adds fewer rows with shorter solution times than starting from an arbitrary solution.

		GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
Heuristic	Rows Added	0	2	0	3	1	0	0
One Row	Time (s)	< 1 + 4	< 1 + 78	< 1 + 48	< 1 + 195	< 1 + 112	< 1 + 1	< 1 + 13
Heuristic	Rows Added	1	1	1	2	1	0	0
Two Rows	Time (s)	7 + 4	10 + 104	4 + 108	9 + 140	5 + 151	1 + 5	1 + 36
Heuristic	Rows Added	0	0	0	1	1	0	0
Three Rows	Time (s)	36 + 3	42 + 81	51 + 39	106 + 90	42 + 25	6 + < 1	10 + 1

Table 9. Tests on optimized instances.

When the sliding partial MILP heuristic starts from an arbitrary solution, it adds more rows and solution times are 0-11 minutes. The two-stage version adds fewer rows and total run times are less than four minutes. So, the best option was found to be running the "heuristic three rows" to generate a starting point and then using the "graded optimize four rows" sliding partial MILP heuristic to generate a valid mapping.

Table 10 shows that 8 out of 21 instances cannot be solved by the sliding partial MILP heuristic for cardinality five interconnect with dedicated pass-gates (for 25%, 33%, and 50%). However, by adding rows of pass-gates the heuristic can solve four of the instances—ADPCM Encoder and Laplace for 33% dedicated pass-gates and ADPCM Decoder and Laplace for 50% dedicated pass-gates—that are proven infeasible for the feasible mapping with fixed rows solution (Baz, 2008). The heuristic adds at most two rows of pass-gates for these solutions and they are shown in bold in Table 10. When we consider the instances solved by the heuristic, the longest run time is 2,130.5 seconds. The two-stage heuristic did not find mappings not found by the one-stage heuristic. This is due to dense structures in these graphs that cannot be separated without moving individual nodes across rows (see Section 7).

		GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
25% Dedicated	Rows Added	0	0	0	-	-	0	0
Pass-gates	Time (s)	< 1	19	5	-	-	< 1	< 1
33% Dedicated	Rows Added	0	**2**	0	-	-	0	**1**
Pass-gates	Time (s)	60	322	42	-	-	7	203
50% Dedicated	Rows Added	1	-	**2**	-	-	-	**2**
Pass-gates	Time (s)	40	-	2,130	-	-	-	116

Table 10. Tests on 5:1 interconnect with 25%, 33%, and 50% dedicated pass-gates.

7. Conclusion

In this chapter we have presented three greedy heuristics for mapping applications onto a reconfigurable device oriented for low-energy execution. The three heuristics are a deterministic top-down greedy algorithm described in Section 4, a greedy algorithm with randomization discussed in Section 5 based on the deterministic algorithm flow, and a partial MILP greedy heuristic presented in Section 6.

Here we compare the deterministic, randomized, weighted randomized, and sliding partial MILP heuristics described in Section 4, Section 5.1, Section 5.2, and Section 6, respectively. The comparisons are made using a 5:1-based interconnect and are shown in Table 11. The results compare the different heuristics in terms of fabric size, path length increase, and mapping time.

		GSM	ADPCM Encoder	ADPCM Decoder	IDCT Column	IDCT Row	Sobel	Laplace
Deterministic Algorithm	Rows Added	1	5	1	8	4	1	2
	Path Length Increase	2	11	1	42	26	1	2
	Time (s)	< 1	12	< 1	1	< 1	< 1	< 1
Random Algorithm	Rows Added	0	4	0	8	4	0	0
	Path Length Increase	0	9	1	42	26	0	0
	Total Time (s)	561	4,262	338	1,148	781	40	63
	sec/iteration	1	9	< 1	2	2	< 1	< 1
Weighted Algorithm	Rows Added	0	0	0	3	2	0	0
	Path Length Increase	0	1	0	22	18	0	0
	Total Time (s)	327	3,803	216	731	517	36	56
	sec/iteration	< 1	8	< 1	1	1	< 1	< 1
Sliding MILP Heuristic	Rows Added	0	3	0	6	2	0	0
	Path Length Increase	0	6	0	40	8	0	0
	Time (s)	79	142	32	635	373	3	87
Two-stage Sliding MILP Heuristic	Rows Added	0	0	0	2	1	0	0
	Path Length Increase	0	0	0	8	8	0	0
	Time (s)	37	123	89	196	67	6	11

Table 11. Comparison of greedy mapping techniques targeting a 5:1 cardinality interconnect.

Each heuristic provides different advantages and disadvantages. For example the deterministic approach provides a solution quickly but not of the highest quality as measured by required fabric size and total path length. The partial MILP heuristic was able to out perform the deterministic approach due to its much larger window size considering entire rows of nodes versus a single node, respectively. Actually, the weighted randomized algorithm provides better qualities of solution than the partial MILP heuristic but the run times are much higher. The two-stage partial MILP heuristic performs the best overall with reasonable run times (actually better than the one-stage partial MILP heuristic in many cases). Thus, if generating mappings in seconds is essential, the deterministic heuristic can be used. If energy consumption is critical and run times in minutes are acceptable, the two-stage sliding partial MILP heuristic should be used.

However, the large multi-row window size for the MILP heuristic became a disadvantage for restrictive interconnects with dedicated pass-gates, for which the randomized greedy heuristic provides the best results and the partial MILP heuristic is not able to solve many cases.

To better understand the sliding partial MILP heuristic performance for this interconnect, we analyzed the eight instances which cannot be solved by the heuristic. The benchmarks IDCT Row and IDCT Column are infeasible because they have nodes which have four commutative

children. In a situation like this, dedicated pass-gates with a cardinality five interconnect is too restrictive. In fact for a cardinality five interconnect with 50% dedicated pass-gates, the partial MILP was unable to map a majority of the benchmarks including one of the smallest ones (Sobel). To be able to solve these cases the operator assignments must be revised such that each node can have at most three children. This requires a pre-processing step to use the sliding partial MILP heuristic that will enforce these input and output restrictions, which will increase the overall path length and possibly the number of rows in the solution.

7.1 Future work

From the exploration of the heuristics described in this chapter there are clear tradeoffs between the three main heuristics. The deterministic approach is fast but far from optimal. The partial MILP heuristic (particularly the two-stage version) is strong for cardinality five and requires a reasonable time (seconds to minutes) to map but has problems when introducing dedicated pass-gates. The weighted randomized heuristic performed reasonably well for solution quality and could map the dedicated pass-gate interconnect, but the run times were too long.

In our future work we plan to investigate methods to improve the runtime of the weighted randomized heuristic. For example, currently the heuristic re-evaluates the weights after the placement of each node. To make the decision faster, the heuristic could select multiple nodes to place based on the current weights before recalculating. Additionally, the early termination can be revised to avoid losing good solutions but also to create more candidates for early termination to improve performance. One example might be to relax the number of row placement failures but to terminate if the path length increase exceeds the current best solution (currently we use solution size).

To improve the performance of the partial MILP heuristic, we can develop a pre-processing pass that relaxes infeasible constructs so that they can be mapped. We also may consider expanding the MILP to allow nodes to move between rows as well as columns. However, this is expected to significantly increase the runtime of the partial MILPs and may require use of a smaller window size. Additionally, we plan to explore other "first-stage" passes for the two-stage heuristic. We may explore using the full fabric MILP to generate a nearly feasible fixed rows mapping or investigate other approaches such as simulated annealing for this first stage.

8. References

Baz, M. 2008. Optimization of mapping onto a flexible low-power electronic fabric architecture. Ph.D. thesis, University of Pittsburgh, Pittsburgh, Pennsylvania.

Baz, M., Hunsaker, B., Mehta, G., Stander, J., and Jones, A. K. 2008. Application mapping onto a coarsegrained computational device. *European Journal of Operations Research*. in submission and revision since April 2007.

Bray, T., Paoli, J., C. M. Sperberg-McQueen, E. M., and Yergeau, F. 2006. Extensible markup language (xml) 1.0 (fourth edition) - origin and goals. Tech. Rep. 20060816, World Wide Web Consortium.

Cordella, L. P., Foggia, P., Sansone, C., and Vento, M. 2004. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26,* 10 (October), 1367– 1372.

Diestel, R. 2005. *Graph Theory*. Springer-Verlag: Heidelberg. 3rd edition.

Garey, M. and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W. H. Freeman.

Hauser, J. R. and Wawrzynek, J. 1997. Garp: A MIPS processor with a reconfigurable coprocessor. In *IEEE Symposium on FPGAs for Custom Computing Machines*, K. L. Pocek and J. Arnold, Eds. IEEE Computer Society Press, Los Alamitos, CA, 12–21.

Hoare, R., Jones, A. K., Kusic, D., Fazekas, J., Foster, J., Tung, S., and McCloud, M. 2006. Rapid VLIW processor customization for signal processing applications using combinational hardware functions. *EURASIP Journal on Applied Signal Processing 2006*, Article ID 46472, 23 pages.

Johnson, T., Robertson, N., Seymour, P. D., and Thomas, R. 2001. Directed tree-width. *Journal of Combinatorial Theory. Series B 82,* 1, 138–154.

Jones, A. K., Hoare, R., Kusic, D., Fazekas, J., and Foster, J. 2005. An FPGA-based VLIW processor with custom hardware execution. In *ACM International Symposium on Field-Programmable Gate Arrays (FPGA)*.

Jones, A. K., Hoare, R., Kusic, D., Mehta, G., Fazekas, J., and Foster, J. 2006. Reducing power while increasing performance with supercisc. *ACM Transactions on Embedded Computing Systems (TECS) 5,* 3 (August), 1–29.

Jones, A. K., Hoare, R. R., Kusic, D., Fazekas, J., Mehta, G., and Foster, J. 2006. A vliw processor with hardware functions: Increasing performance while reducing power. *IEEE Transactions on Circuits and Systems II 53,* 11 (November), 1250–1254.

Jones, A. K., Mehta, G., Stander, J., Baz, M., and Hunsaker, B. 2008. Interconnect customization for a hardware fabric. *ACM Transactions on Design Automation for Electronic Systems (TODAES)*. in press.

Krissinel, E. B. and Henrick, K. 2004. Common subgraph isomorphism detection by backtracking search. *Software – Practice and Experience 34*, 591–607.

Levine, B. and Schmit, H. 2002. Piperench: Power and performance evaluation of a programmable pipelined datapath. In *Presented at Hot Chips 14*.

MathStar. Field programmable object array architecture. http://www.mathstar.com/literature.html.

Mehta, G., Hoare, R. R., Stander, J., and Jones, A. K. 2006. Design space exploration for low-power reconfigurable fabrics. In *Proc. of the Reconfigurable Architectures Workshop (RAW)*.

Mehta, G., Ihrig, C. J., and Jones, A. K. 2008. Reducing energy by exploring heterogeneity in a coarse-grain fabric. In *Proc. of the IPDPS Reconfigurable Architecture Workshop (RAW)*.

Mehta, G., Stander, J., Baz, M., Hunsaker, B., and Jones, A. K. 2007. Interconnect customization for a coarse-grained reconfigurable fabric. In *Proc. of the IPDPS Reconfigurable Architecture Workshop (RAW)*.

Mehta, G., Stander, J., Lucas, J., Hoare, R. R., Hunsaker, B., and Jones, A. K. 2006. A low-energy reconfigurable fabric for the SuperCISC architecture. *Journal of Low Power Electronics 2,* 2 (August).

Messmer, B. T. and Bunke, H. 2000. Efficient subgraph isomorphism detection: A decomposition approach. *IEEE Transactions on Knowledge and Data Engineering 12,* 2, 307–323.

Mirsky, E. and Dehon, A. 1996. Matrix: A reconfigurable computing architecture with configurable instruction distribution and deployable resources. In *Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines*.

Resende, M. and de Sousa, J. 2004. *Metaheuristics: Computer Decision-Making*. Kluwer Academic Publishers.

Resende, M. and Ribeiro, C. 2008a. *Handbook of Metaheuristics, 2nd Edition*. Springer Publishers.

Resende, M. and Ribeiro, C. 2008b. *Search Methodologies, 2nd Edition*. Springer Publishers.

Sheng, L., Kaviani, A. S., and Bathala, K. 2002. Dynamic power consumption in virtex-II FPGA family. In *FPGA*.

Ullmann, J. R. 1976. An algorithm for subgraph isomorphism. *J. ACM 23,* 1, 31 42.

12

Greedy Algorithms for Spectrum Management in OFDM Cognitive Systems - Applications to Video Streaming and Wireless Sensor Networks

Joumana Farah[1] and François Marx[2]

[1]*Faculty of Sciences & Computer Engineering, Holy-Spirit University of Kaslik, Jounieh,*
[2] *France Telecom, Paris,*
[1]*Lebanon*
[2]*France*

1. Introduction

Since the beginning of regulatory planning in radio communications, new advancements in technology have been driving spectrum management procedures. Sophisticated techniques were introduced to improve the systems spectral efficiency while keeping pace with an increasing demand for new services and higher transmission rates. However, a new paradigm emerged recently in which regulation has driven technology. The exploding success of the first experiments in "open spectrum" using the ISM (Industrial/Scientific/Medical) bands gave rise to a tremendous interest in finding new strategies of spectrum management that will permit a more flexible and opportunistic utilization of the spectrum, without causing harm to existing services. This challenge is of a great concern to the proponents of new generations of communication systems because of the scarcity of spectrum resources.

For this reason, the Federal Communication Commission published, in the last few years, several documents (FCC RO, 2003; FCC NOI and NPRM, 2003) that aimed to improve the radio spectrum allocation, using two different strategies: spectrum leasing and cognitive or smart radio. In the first one, a trading market for spectrum usage is proposed, and users can be dynamically rented the access to a piece of spectrum using a certain arrangement. The second type of dynamic spectrum assignment is the Open Spectrum approach, which allows users to sense available and unallocated spectrum. In this case, the overall spectrum is shared among all users and spectrum etiquette is used to limit harmful interference.

In both cases, but for different motivations - financial stakes in the case of spectrum leasing and voluntary rules for the spectrum etiquette - optimizing the spectrum usage has become of major importance. Therefore, a mechanism must be set in each access point of the communication system in such a way to utilize the unused spectrum in an intelligent way, while not interfering with other incumbent devices in already licensed frequency bands. The spectrum usage will be minimized by an optimization of the channels allocation scheme, so that the spectrum freed by an operator may be used by another operator.

Therefore, new technical challenges must be overcome to enable the success of the cognitive radio paradigm: supporting different air interface standards, operating in multiple environments, adapting to several radio access techniques, counteracting the influence of channel impairments (multipath, fading, noise), coping with user mobility, and guaranteeing a minimum quality of service with an affordable transmission power.
To reach these goals, this chapter investigates the problem of dynamic multiuser subchannel allocation for minimizing spectrum usage. The system overall bandwidth is supposed to be equally divided into a set of frequency bands, therefore assuming Orthogonal Frequency Division Multiplexing (OFDM).
In former studies, most of the work dealing with dynamic spectrum allocation aimed either at maximizing the total system capacity or at minimizing the total transmission power. In (Rhee & Cioffi, 2000), an iterative algorithm was proposed that attributes subchannels to the users in such a way to maximize the smallest user capacity. However, an equal amount of power is allocated to each subcarrier. In (Kim et al., 2004), a maximization of the rate-sum capacity was realized by iterative subcarriers assignment followed by water-filling for power allocation. In (Toufik & Knopp, 2004), a graph theory approach was used to assign a fixed number of subcarriers to each user. Two strategies were considered: maximization of the total transmission rate under the constraint of a fixed amount of transmission power or minimization of the total transmission power while guaranteeing a set of users data rates. The second strategy was also the subject of study in (Wong et al., 1999) and (Kivanc et al., 2003). In (Wong et al., 1999), a set of subcarriers is first assigned to each user based on the Lagrange optimization resolved by parameter relaxation. The transmission power and the number of bits in each subcarrier are then determined by a greedy approach. In (Kivanc et al., 2003), a number of subcarriers is first allocated to each user based on its average Signal-to-Noise Ratio, assuming a flat-fading channel for the user. The best assignment scheme of the subchannels is then determined by an iterative algorithm.
In this chapter, we propose novel techniques based on greedy algorithms for the optimization of the spectrum efficiency of an OFDM transmission system. The aim is to minimize the total allocated bandwidth while guaranteeing a certain transmission data rate to each user, under the constraint of a total transmission power.
We begin, in section 2, by a description of the overall downlink transmission system using OFDM. Then, in section 3, we present two classical approaches for spectrum management based on Frequency Division Multiple Access (FDMA) and Time Division Multiple Access (TDMA). After discussing the disadvantages of these approaches, we explain how the spectrum allocation can be optimized by a proper formulation of a combinatorial assignment problem. Since exact solutions of this problem are impossible to obtain, we present, in section 4, a solution to this problem based on the Hungarian approach (or Munkres algorithm). Then, we propose an enhanced version of this solution in section 5. A quasi-optimal solution is investigated in section 6, based on a simulated annealing approach. A comparative analysis of the simulation results as well as the computational complexity of the different algorithms can be found in section 7. Finally, section 8 is an overture to different applications of our greedy approaches. For this reason, two application examples are presented: optimization of the terminals autonomy in a Wireless Sensor Network and optimization of a multi-user video streaming system where source and channel encoded video sequences are transmitted using OFDM.

2. Description of the OFDM downlink transmission system

The system consists of K mobile users, each requesting a download data rate R_k (k = 1, …, K) and uniformly distributed over the cell coverage area. We assume that all users have access to all time slots and that a given subchannel of the OFDM uplink system is allocated to only one user in each time slot (subchannels sharing is not allowed).
After demultiplexing of each original user's binary sequence, subcarriers' symbols are modulated and transformed in the time domain by inverse fast Fourier transform (Figure 1). Cyclic extension is then added (IEEE, 1999) before pulse shape filtering. After transmission through a frequency selective Rayleigh fading channel, each subcarrier will experience a different channel gain $H_{k,n}$ in the frequency domain (Haykin, 2001). In the case of a multiuser system, these channel gains will constitute a channel matrix as in Figure 2, since each channel is seen "from a different point of view", depending on the user to which it is attributed. We assume that the base station receives channel information from all users and can perfectly estimate the channel state on each subcarrier using pilots inserted in a scattered manner in the transmitted OFDM symbols (IEEE, 1999).

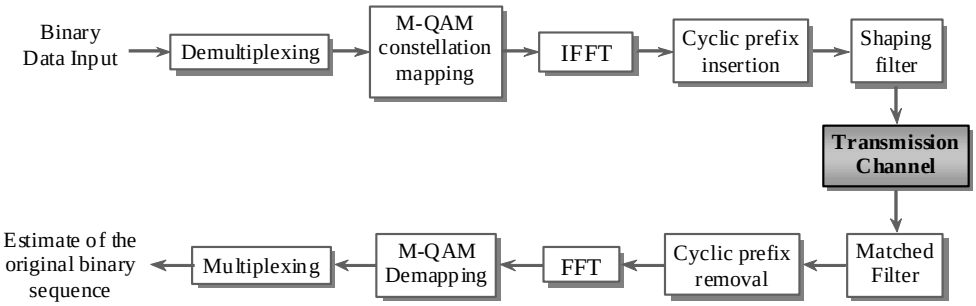


Fig. 1. Overall baseband model of an OFDM transceiver.

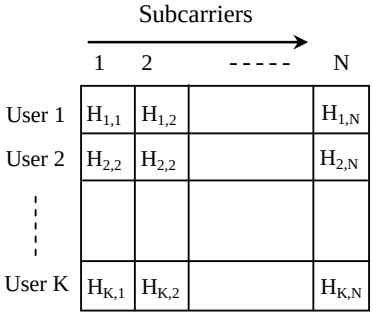


Fig. 2. Channel matrix of the downlink multiuser OFDM system.

The following notations will be used throughout the paper:

- N is the maximum number of available subchannels which form a set S.
- B is the total system bandwidth.
- S_k is the set of subcarriers allocated to user k.
- P_{max} is the maximum allowed transmission power by the base station.
- $P_{k,n}$ is the power transmitted on the subcarrier n allocated to the user k.

- N_0 is the power spectral density of the additive white Gaussian noise (assumed to be constant over all subcarriers).

3. Classical OFDM spectrum allocation approaches and problem formulation

3.1 OFDM-FDMA approach

In this approach, which uses FDMA in conjunction with OFDM, users are treated sequentially in time (Figure 3): for each user, subchannels are allocated one by one until the user transmission rate becomes at least equal to the target data rate R_k. In this strategy, subchannels are assigned to users without any consideration for the users channel state information. Subcarrier gains are only taken into account in the calculation of the users actual data rates while the transmission powers $P_{k,n}$ are all equal to the same constant, independently from the subcarriers or the users. If we suppose that all users transmit at their capacity limit, the k^{th} user's total transmission rate, at a certain stage of the allocation process, can be written as:

$$R_{k,tot} = \sum_{n \in S_k} \frac{B}{N} log_2 \left(1 + \frac{P_{k,n} H_{k,n}^{\ 2}}{N_0 \frac{B}{N}} \right)$$

Iterative algorithm for subcarrier allocation in an OFDM-FDMA system

$n = 1$ // Initialization
$k = 0$
(L1) $k = k + 1$
$R_{k,tot} = 0$
while $(R_{k,tot} < R_k)$
  If $(n > N)$ // Test if S is empty
    Break
  End If
  $S_k = S_k \cup \{n\}$ //Attribute subcarrier n to user k
  $S = S \cap \{n\}^C$ // Remove n from the subcarriers set S
  $P_{k,n} = P_{max} / N$ // Equal power allocation
  $R_{k,tot} = \sum_j \frac{B}{N} log_2 \left(1 + \frac{P_{k,j} H_{k,j}^{\ 2}}{N_0 \frac{B}{N}} \right)$ // Estimate the user's actual transmission rate
  $n = n + 1$
End while
If $(S = \phi)$ // Test if S is empty
  Return // User target rate cannot be reached: End the attribution process
End If
If $(k = K)$
  Return // All users target rates were reached: End the attribution process
Else
  Goto **L1**
End If

Fig. 3. Description of the iterative subcarrier allocation algorithm in OFDM-FDMA.

3.2 OFDM-TDMA approach

Classically, in an OFDM system using TDMA (Rohling & Grunheid, 1997), the spectrum bandwidth is entirely allocated to only one user during a certain period of time. The transmission frame is divided into a number of time slots equal to the number of users K. However, in order to permit a fair comparison with the optimized greedy approaches, and for the respect of spectrum etiquette (as explained in section 1), the user must only be assigned subcarriers that can actually increase its transmission rate. For this purpose, each step of the allocation algorithm consists (Figure 4) in assigning the user currently under consideration the best possible subcarrier, i.e. the subcarrier with the highest channel gain $H_{k,n}$. The subcarrier is then removed from the set of available subcarriers S, and power allocation is applied to the subcarriers so far allocated to the user (subcarriers in set S_k). This process is iterated until the user's target rate is achieved, unless the set S is empty.

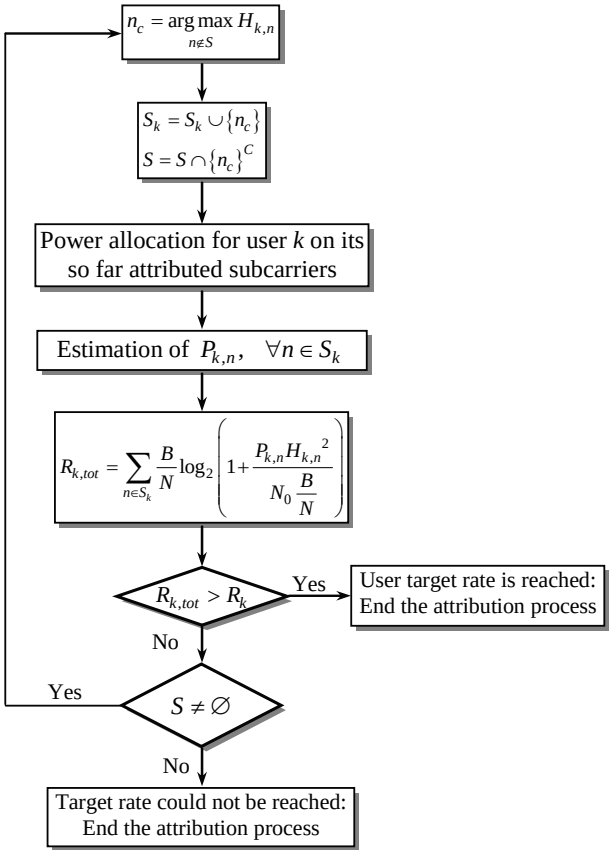


Fig. 4. Subcarrier allocation for user k using OFDM-TDMA.

As for the power allocation in OFDM-TDMA, it consists on a distribution of the total transmission power P_{max} on the user k allocated subcarriers. In other words, $\{P_{k,n}, \ n \in S_k\}$ are to be determined in such a way to maximize the total transmission rate for user k, under the transmission power constraint:

$$\max_{\{P_{k,n},n\in S_k\}} \sum_{n\in S_k} \frac{B}{N} log_2\left(1+\frac{P_{k,n}{H_{k,n}}^2}{N_0\frac{B}{N}}\right),$$

under the constaints:

$P_{k,n} \geq 0, \forall k, \forall n \in S$

$$\sum_{n\in S_k} P_{n,k} \leq P_{max}$$

It can be proven (Haykin, 2001; Cioffi, 2008) that the solution to this constrained optimization problem is:

$$\begin{cases} P_{k,n} + v_n = \alpha, & \forall n \in S_k \\ \sum_{n\in S_k} P_{n,k} = P_{max} \end{cases}$$

where α is a constant and $v_n = \frac{N_0 \cdot B/N}{{H_{k,n}}^2}$, $\forall n \in S_k$.

An illustration of this solution is given in Figure 5 for the example of five subcarriers attributed to user *k*.

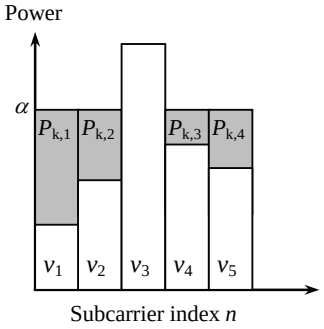


Fig. 5. Representation of the water-filling solution to the power allocation problem in OFDM-TDMA.

This graphical interpretation clearly justifies the name of "water-filling" given to the power allocation technique. It can be seen that highly attenuated subcarriers (small $H_{k,n}$) will be allocated much less power than subcarriers with great amplitudes, therefore the transmission rate on those subcarrier will be much smaller than on others. In case v_n is higher than α (ex: subcarrier 3), no power will be allocated to this subcarrier (i.e. corresponding data rate is null).

In the sequel, we propose a gradual water-filling algorithm for the realization of power allocation. For this purpose, let:

w the current waterline level,

P_{tot} the total transmission power for the current waterline,

P_{tol} the absolute tolerable error on the total transmission power,
l_k the number of subcarriers so far allocated to the user k (l_k is the number of elements in the set S_k), and
$v_{n,min}$ the vector having the smallest amplitude among the vectors $v_n, \forall n \in S_k$.
At the beginning of the power allocation algorithm, the transmission powers $P_{k,n}$ are all set to zero. Water-filling is performed in an iterative way (Figure 6) such that the absolute difference between the powers P_{tot} and P_{max} does not exceed the tolerable error P_{tol}. The initialization of the waterline is chosen in such a way that, in case all vectors v_n are equal, the waterline remains constant and the amount of power attributed to each user is P_{max} / l_k.

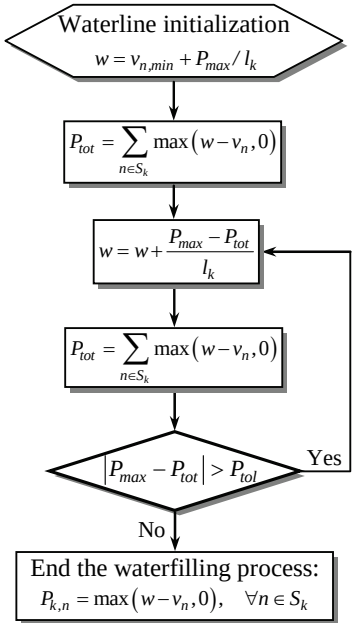


Fig. 6. Water-filling technique for power allocation in OFDM-TDMA.

3.3 Disadvantages of the classical approaches and formulation of the optimization problem

By analyzing the OFDM-FDMA assignment, we notice that a major drawback of this technique is that a subcarrier n can be assigned to a certain user k_1 while there exists a user k_2 for whom the attribution of this subcarrier would be much more profitable ($H_{k_2,n} > H_{k_1,n}$).
In other words, the additional user rate obtained by the attribution of subcarrier n to user k_1 can be negligible compared to the one that could be achieved by its attribution to user k_2. In our attempt to reach each user's target rate, a large number of subcarriers and a high transmission power will be needed. Furthermore, since users are processed in order, subsequent ones will have smaller chances to reach their target rate. This problem will appear more often as the users target rates increase.
As for the OFDM-TDMA attribution scheme, one of its disadvantages is that each user actually transmits data at a rate of $K \cdot R_k$ [bit/s] during one time slot of the transmission

frame and remains inactive during the other time slots. This effect can be disturbing in the case of real-time applications because of the high latency, especially when the number of users increases. Another important disadvantage is that the necessary number of sub-carriers at a certain moment can reach important values, particularly for high user rates or short time slot durations. Therefore, the risk of failing to satisfy all users target rates is very high.

These disadvantages of the OFDM-FDMA and OFDM-TDMA assignment techniques can be greatly alleviated if the subcarriers were allocated to the users in such a way to take into account the channel states of all users. Therefore, the prohibitive spectrum usage necessitated by those simple techniques, especially for important target rates, can be overcome by applying a dynamic assignment strategy that aims at minimizing the total number of allocated subcarriers under the constraints of the target data rates and the maximum allowed transmission power.

The corresponding optimization problem can be formulated as follows:

$$\min_{P_{n,k},S_k} \sum_{k=1}^{K} card\left(S_k\right)$$

subject to the following constraints:

$$\sum_{n\in S_k} \frac{B}{N} log_2\left(1+\frac{P_{k,n}{H_{k,n}}^2}{N_0\frac{B}{N}}\right) = R_k, \forall k$$

$$\sum_{k=1}^{K}\sum_{n\in S_k} P_{n,k} \le P_{max}$$

$$P_{k,n} \ge 0, \forall k, \forall n \in S$$

$$S_i \bigcap_{i\neq j} S_j = \varnothing$$

$$\bigcup_{k=1}^{K} S_k \subseteq \{1,2,...N\}$$

The first constraint specifies the target transmission rate per user. The second and third conditions specify the power constraints. The last two conditions specify the maximum number of allocated subcarriers and that each subcarrier can be allocated to only one user at a certain time.

We can clearly note that the optimization problem formulated above is a combinatorial assignment problem since set selection is involved. Therefore, it does not form a convex problem. In the literature, several attempts have been made to transform it into a convex optimization problem. In (Kim et al., 2004), specifications were relaxed by introducing a new parameter representing a portion of a subchannel assigned to a user. In (Wong et al., 1999), time sharing of a subcarrier among different users is considered. In either case, the solution obtained constitutes a lower bound to the combinatorial optimization problem. However, a general formulation of this solution is not obvious and since it only provides a lower bound, it is preferable to strive after a quasi-optimal solution to the real assignment problem.

In the following, we describe several possible strategies to determine a quasi-optimal solution of the combinatorial optimization problem, using a greedy approach.

4. Greedy technique for the optimization of spectrum resources using the Munkres algorithm (GOSM)

In order to determine the best system configuration for the optimization problem presented in section 3, we came out with a greedy iterative algorithm that determines the best spectrum allocation scheme by applying a separate optimization algorithm that assigns the subcarriers to the users in such a way to minimize a cost function. This assignment is then followed by power allocation. The optimization algorithm is the well-known Munkres assignment algorithm (Munkres, 1957), also known by the Hungarian method. In our application case, the cost function is the sum of the opposite channel amplitudes $-H_{k,n}$ and it is applied independently from the users actual transmission rates.

At a certain stage of the optimization algorithm, we consider:
U: the set of users whose target data rates have not been reached so far,
k_U: the number of users in U,
S_U: the set of subcarriers attributed to the users in the set U,
l_U: the number of subcarriers in S_U,
P_{rem}: the remaining allowed transmission power after a user has reached its target rate,
P_{tot}: the total transmission power for the current waterline, corresponding to the users in the set U,
P_{tol}: the absolute tolerable error on the total transmission power,
R_{tol}: the required precision on the users target data rates.

At the beginning of the greedy channel allocation, the transmission powers $P_{k,n}$ are all initialized to zero and P_{rem} is initialized to P_{max}.
Our proposed greedy iterative algorithm (Figure 7) can be described as follows:
In each iteration, the Munkres algorithm is used to allocate a subcarrier n_k to each user k that has not reached so far its target data rate R_k (users in the set U). The allocated subcarriers are removed from the set S. Then, water-filling is applied on the allocated subcarriers, using the available remaining power P_{rem}. The water-filling is performed by taking into account only users in the set U. After water-filling, the transmission power is estimated on all allocated subcarriers as well as the actual total transmission rate $R_{k,tot}$ for each user k in the set U. If $R_{k,tot}$ is higher than the target rate R_k, the transmission power on the allocated subcarrier for user k with the least channel amplitude has to be reduced in such a way to adjust the total rate to the exact target rate. Finally, user k is removed from U and the remaining power P_{rem} is updated. The algorithm is iterated with the remaining users, unless the set S of available subcarriers is empty.
By analyzing this allocation technique, it can be seen that, in each iteration, a local optimum is chosen in the hope of reaching the global optimum at the output of the overall algorithm. Therefore, it is indeed a greedy allocation algorithm.
As for the adjustment of the transmission rate for user k before its removal from the set U, it is realized using the following equations:

$$m = \underset{n \in S_k}{arg\,min}\, H_{k,n}$$

$$R_m = R_{k,tot} - \frac{B}{N} log_2\left(1 + \frac{P_{k,m}}{\nu_m}\right)$$

$$P_{k,m} = \left(2^{(R_k - R_m).N/B} - 1\right).\nu_m$$

Finally, water-filling is performed in the same manner as in Figure 6 except that P_{max} is replaced by P_{rem}, l_k by l_U, and S_k by S_U.

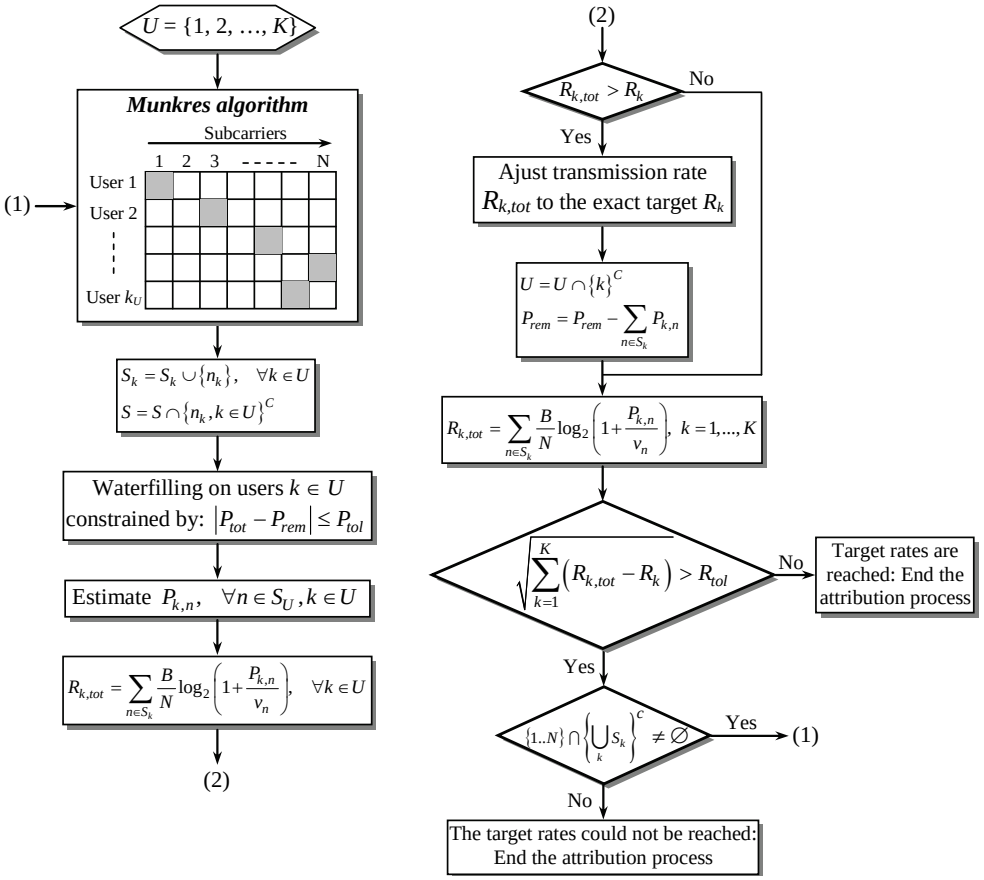


Fig. 7. Greedy iterative technique for dynamic spectrum allocation using the Munkres algorithm (GOSM).

5. Enhanced greedy algorithm for spectrum optimization (EGAS)

As it will be seen in section 7, the GOSM allocation technique has the disadvantage of attributing subchannels to the users without taking into account their actual transmission rates. For this reason, we propose the following enhanced greedy algorithm for dynamic spectrum allocation:

***Step 1*:** start by identifying the user k_c, whose actual total transmission rate is the farthest from its target data rate.

$$k_c = \underset{k \in U}{arg\,max} \left(R_k - R_{k,tot} \right).$$

***Step 2*:** attribute to this user the most favorable subcarrier n_c.

$$n_c = \underset{n \notin \bigcup_k S_k}{arg\,max} H_{k_c,n}.$$

$$S_{k_c} = S_{k_c} \cup \{n_c\}$$

***Step 3*:** remove n_c from the set S.

$$S = S \cap \{n_c\}^C$$

***Step 4*:** perform water-filling on the allocated subcarriers for all users in the set U, using the available remaining power P_{rem}.

***Step 5*:** estimate the transmission power on all allocated subcarriers as well as the actual total data rate for user k_c ($R_{k_c,tot}$).

***Step 6*:** Check if $R_{k_c,tot}$ exceeds the target rate R_{k_c}. If yes, adjust the transmission power of user k_c on the subcarrier with the least channel amplitude to reach the exact target rate R_{k_c} (as described earlier in section 4) and go to *Step 7*. Otherwise, go to *Step 8*.

***Step 7*:** Remove user k_c from the set U and update the remaining power P_{rem}.

***Step 8*:** Evaluate all users' actual rates. End the attribution process in case the target rates have been reached with a precision R_{tol}. Otherwise, loop on *Step 1*, unless no more subcarriers are available (in this case, the attribution process has failed to satisfy the users target rates).

6. Optimization of the EGAS algorithm by simulated annealing (SAS)

In the aim of determining an optimal solution to the combinatorial problem presented in section 3.3, one can think of an exhaustive search that would constitute a lower bound to the cost function under consideration. Unfortunately, due to the large number of parameters and constraints involved in the optimization problem, such a strategy appears to be of an impractical use, especially when the ranges of the system variables (i.e. the number of subcarriers and users) increase.

On the other side, in the greedy approaches we applied for resolving our optimization problem, the search can unfortunately get stuck in a local optimum, especially when the global optimum is hidden among many other poorer local optima. In order to overcome this difficulty, one possibility is to carry out the iterative process several times starting from different initial configurations. A similar strategy was applied in statistical mechanics (Metropolis et al., 1953) for determining the ground state of systems by a simulated annealing process. In (Kirkpatrick et al., 1983), it was shown how the analogy between

statistical mechanics and multivariate or combinatorial optimization can be exploited to resolve optimization problems of large scales.

This technique, also known by the Metropolis algorithm, simulates an aggregation of atoms in equilibrium at a certain temperature. The cost function is the system energy E. In each step of the simulation algorithm, an atom is randomly selected and displaced. If the resulting change in the system energy ΔE is negative, the modification in the system configuration is accepted, and this new configuration is retained as the initial state of the following iteration. If ΔE is positive, the modification is accepted with probability $\Pr(\Delta E) = \exp(-\Delta E / K_B \lambda)$, where λ is the system temperature and K_B the Boltzmann's constant. Therefore, the system will evolve into a Boltzmann configuration.

As for the temperature, it is used as a control parameter in the same unit as the cost function. When large, the control parameter allows the system to make transitions that would be improbable at low temperatures. Therefore, an annealing process is simulated by first "melting" the system and making coarse changes in the system configuration, and then gradually "freezing" the system by lowering the temperature to develop finer details until the optimal structure is attained.

In this study, we applied the Metropolis algorithm subsequently to the EGAS procedure presented in section 5. The temperature parameter is gradually decreased throughout the iterations.

In each iteration, a first step consists in a random selection of one of the three following possible actions:

Action 1: We randomly select two users k_1 and k_2 and interchange two random subcarriers between the two users. Next, a water-filling is realized separately for each of the two users, on their allocated subcarriers. The water-filling procedure is constrained by the user's target data rate and will be described in the sequel.

Action 2: A randomly chosen subcarrier is removed from a random user k. Water-filling is then performed for k on its allocated subcarriers, with a constraint on its target rate.

Action 3: A free subcarrier is randomly chosen and attributed to a random user k. Water-filling is then performed for k on its allocated subcarriers, with a constraint on its target rate.

The next step is to decide whether the action is accepted. For this purpose, we estimate the new total number of attributed subcarriers $L = \sum_{k=1}^{K} card\left(S_k\right)$ as well as the total transmission power $P = \sum_{k=1}^{K} \sum_{n \in S_k} P_{n,k}$. The action is accepted only in the three following possible cases:

Case 1: The total number of subcarriers L was decreased while the constraint on the total transmission power was still respected ($P \leq P_{max}$).

Case 2: The total transmission power P was decreased while maintaining the same number of subcarriers L.

Case 3: Neither the total number of subcarriers L nor the total transmission power P could be decreased. However, $P \leq P_{max}$. In this case, the action is accepted with probability $\Pr(\Delta L) = \exp(-\Delta L / K_B \lambda)$, where ΔL is the increase in the number of subcarriers.

Note that when an action is accepted, the actual system configuration, i.e. the allocation scheme of the subcarriers to the users, is adopted as the initial condition for the subsequent iteration. Besides, due to the stochastic nature of this procedure, the algorithm has to be executed several times in order to increase the chance of finding a global optimum.

As for the water-filling, it is constrained by the user data rate, instead of the available power as in section 3.2. It will be realized using a gradual dichotomy-based approach as described hereafter, where w_{step} is the current waterline step, l_k the number of subcarriers allocated to a user k, etc.

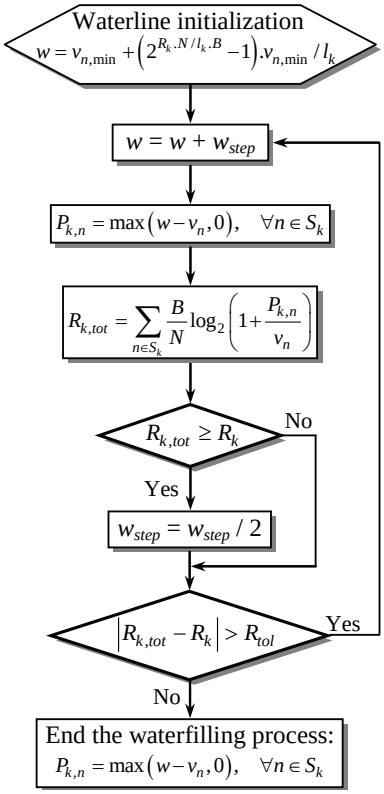


Fig. 8. Gradual dichotomy-based water-filling for the SAS algorithm.

As it can be seen in Figure 8, the waterline is increased using a variable step such that the absolute difference between the achieved data rate and the user's target rate does not exceed R_{tol}. As for the waterline initialization, it is chosen in such a way to satisfy the data rate constraint in the case where all subcarriers have the same SNR. This SNR's value is taken as the highest one among all subcarriers.

7. Performance analysis of the different allocation techniques

7.1 Simulation conditions

The performance of the allocation techniques for spectrum optimization were obtained by simulating a downlink OFDM system with a bandwidth B = 100 MHz and a maximum number of 1024 subcarriers per OFDM symbol. The simulations were conducted in the following conditions:

- The number of users ranges from 10 to 60.

- The total permissible transmission power (by the base station) is P_{max} = 1000 mW.
- The noise power spectral density, constant over all subcarriers, is $N_0 = 4 \cdot 10^{-18}$ Watt/Hz.
- The absolute tolerable error on the user target rate is $R_{tol} = 10^{-3}$ bit.
- The absolute tolerable error on the total transmission power is $P_{tol} = 10^{-5}$ mW.

The transmission medium is a frequency-selective Rayleigh fading channel with a root mean square delay spread (RMS) of 150 or 500 nanoseconds. The users geographic locations are uniformly distributed in a 10 Km radius cell with a maximum path loss difference of 20 dB between users.
The performance of the different methods is compared in terms of the average total number of allocated subcarriers. In the case of the OFDM-TDMA approach, we measured the median and the maximum of the necessary number of subcarriers (Median OFDM-TDMA and Max OFDM-TDMA).
In Tables 1 and 2, we represent the total number of subcarriers for different user data rates, whereas in Table 3, the results are represented as a function of the number of users. In Tables 1, 2 and 3, we consider that the requested data rates are the same among users. However, in Table 4, the results are obtained for different target rates between users:

$$R_k = R_0 \cdot 0.2 + 0.2 \cdot (k\text{-}1),\ k = 1, ..., K,$$

where R_0 is chosen such that, for K = 20 users, the total transmission rate ranges from 48 to 88 Mbit/s.

7.2 Analysis of the practical results

Rate (Mbit/s)	1	2	3	4	5	6	7	8	9	10
OFDM-FDMA	234	420	529	737	852	-	-	-	-	-
Max OFDM-TDMA	17	41	82	142	253	-	-	-	-	-
Median OFDM-TDMA	13	34	61	94	139	221	501	-	-	-
GOSM	19	36	58	81	111	151	206	249	-	-
EGAS	19	35	56	79	109	146	193	224	312	360
SAS	16	33	55	78	108	144	192	221	307	355

Table 1. Total number of subcarriers necessary to achieve different users data rates for K = 10, channel RMS = 150 ns.

Rate (Mbit/s)	1	2	3	4	5	6
OFDM-FDMA	362	689	-	-	-	-
Max OFDM-TDMA	43	144	-	-	-	-
Median OFDM-TDMA	33	89	193	-	-	-
GOSM	39	81	137	225	350	-
EGAS	40	78	134	214	306	414
SAS	38	76	130	211	303	407

Table 2. Total number of subcarriers necessary to achieve different users data rates for K = 10, channel RMS = 500 ns.

Number of users	10	20	30	40	50	60
OFDM-FDMA	196	455	633	799	-	-
Max OFDM-TDMA	16	44	87	148	334	-
Median OFDM-TDMA	14	34	57	89	139	205
GOSM	19	40	66	95	131	164
EGAS	17	39	71	121	147	177
SAS	16	37	57	88	123	153

Table 3. Total number of subcarriers necessary to achieve a data rate of 1 Mbit/s for different numbers of users.

Total Rate (Mbit/s)	48	58	68	78	88
Median OFDM-TDMA	106	148	268	334	-
GOSM	109	143	189	248	-
EGAS	106	134	169	211	245
SAS	103	131	167	208	243

Table 4. Total number of subcarriers as a function of the total transmission rates for the case of different classes of services between users (K = 20).

We can see that our greedy strategies for spectrum optimization clearly outperform the OFDM-FDMA approach in both cases of the channel RMS. For RMS = 150 ns, starting from R_k = 6 Mbit/s, the OFDM-FDMA technique fails to satisfy the users target rates under the transmission power constraint. At R_k = 5 Mbit/s and K = 10, the gain of the three iterative techniques (GOSM, EGAS, and SAS) towards the OFDM-FDMA reaches almost 750 subcarriers (more than 70 % of the available subcarriers). In order to allow a fair comparison, we also tested the performance of the OFDM-FDMA technique in case the subchannels are randomly allocated to users in order to avoid the attribution of several neighboring subcarriers in deep fade to the same user. We noticed that the interleaved OFMA-FDMA technique outperforms the non-interleaved OFDM-FDMA by less than 10 % of the total number of subcarriers. Besides, when the channel RMS increases to 500 ns, both the OFDM-FDMA and OFDM-TDMA techniques fail to satisfy users starting from R_k = 3 Mbit/s.

On the other hand, at R_k = 5 Mbit/s and K = 10, the gain of our greedy techniques is approximately 300 towards the Median OFDM-TDMA. Starting from 9 Mbit/s at RMS = 150 ns and 6 Mbit/s at RMS = 500 ns, even the GOSM technique fails to determine a possible solution for the allocation problem, whereas the EGAS continues to perform well and presents similar results to the quasi-optimal SAS. In fact, the SAS technique permits an amelioration of the system performance, especially when the number of users is important (Tabe 3).

When the users present different data rates, the EGAS approach outperforms the GOSM, especially when the total transmission rate increases. Indeed, as explained in section 5, the EGAS greedy algorithm allows the user whose actual transmission rate is far from its target rate the "right to choose" a favorable subcarrier. Whereas in the GOSM method, all users that have not reached their target rate are allocated a subcarrier at each iteration, regardless of their actual transmission rate, even the ones that are close to reaching their target. This will lead to a higher global amount of allocated subcarriers.

However, as the number of users increases, the GOSM tends to present a slightly better performance than the EGAS (Table 3). This is due to the fact that when the number of

subcarriers per user decreases, there is a higher chance that certain subcarriers are favorable to more than one user. In these conditions, the application of an optimal assignment of the subcarriers by the Munkres algorithm can improve the overall system throughput. As the number of users approaches N, the performance of the GOSM algorithm will tend to be optimal.

As a conclusion to this comparative study, the GOSM technique has the advantage of optimizing the subcarriers allocation between users; however, it does not take into account the users actual transmission rates in the optimization process. Not only does the EGAS take these rates into account, but it also presents a much more affordable complexity towards the GOSM, as will be proven in section 7.3.

7.3 Analysis of the algorithms complexity

Most of the previously presented algorithms for spectrum allocation largely use the water-filling block which constitutes one of the most constraining part in terms of complexity. However, the direct estimation of the complexity of the iterative water-filling procedure presented in section 3.2 is rather impractical, mainly because of its dependence on the tolerance parameter P_{tol}. For this reason, we will replace it, in this part, by the exact water-filling distribution that can be derived as the solution of a linear system of N+1 equations:

$$P_{k,n} + v_n = \alpha, \qquad n = 1,...,N$$

$$\sum_{n=1}^{N} P_{k,n} = P_{max}$$

with N+1 unknowns ($P_{k,n}$ and α).

A simple algorithm was proposed in (Cioffi, 2008) to solve this system. It is summarized in Figure 9.

Simple water-filling algorithm for power allocation

$P_{k,n}$ = zeros(K,N) // Initialization of all subcarriers powers to zero
Sort the v_n vectors such that $v_1 < v_2 < \ldots < v_N$
Set $i = N$
while ($i > 0$)
  Compute $K_i = \frac{1}{i}\left[P_{max} + \sum_{n=1}^{i} v_n\right]$
  If ($K_i - v_i > 0$) // Check the vector corresponding to the lowest power
    $P_{k,n} = K_i - v_n, \forall n \le i$
    $P_{k,n} = 0, \forall n > i$ // Allocate an appropriate amount of power to each subcarrier
    Return
  Else
    $i = i - 1$ // Eliminate the most negative component and recalculate K_i (unless $i = 0$)
  End If
End while

Fig. 9. Estimation of the water-filling solution for power allocation.

In fact, the sorting step is performed only once. Therefore, the water-filling algorithm complexity is $O(N \cdot \log(N))$ if the sorting step is taken into account and $O(N)$ if not. The latter case will be considered in the sequel.

Now that the water-filling complexity has been studied, we can proceed with the complexity of all spectrum allocation algorithms.
First of all, OFDM-FDMA is basically a loop on each subcarrier; hence, its complexity is $O(N)$.
OFDM-TDMA is a loop on each subcarrier, with a water-filling power allocation each time a new subcarrier is allocated to the user. The number of subcarriers involved in the water-filling procedure is successively 1, 2, …, N. The complexity is therefore $O(N^2)$. However, since the algorithm must be run sequentially for each user, the total complexity of OFDM-TDMA is $O(K \cdot N^2)$.
On the other hand, the complexity of the GOSM technique is dominated by the Munkres assignment algorithm which has a complexity $O((\min(N,K))^2 \cdot \max(N,K))$ (Burgeois, 1971). It assigns K subcarriers at each stage. Since $K<N$, the total complexity of GOSM is $O(N/K \cdot K^2 \cdot N)=O(K \cdot N^2)$.
As for the EGAS technique, a new power allocation distribution (i.e. a water-filling step) is realized for each allocated subcarrier, leading to a total complexity of $O(N^2)$.
Finally, each iteration of the SAS algorithm consists of at least a water-filling step. Therefore, its complexity is approximately $O(n_{iter} \cdot N)$, where n_{iter} is the number of iterations in the SAS algorithm.
Since in general $n_{iter} >> K \cdot N$, it can be seen from Table 5 that the OFDM-TDMA and GOSM approaches present a similar complexity, which is much smaller than the one for the SAS algorithm, but higher than that of the EGAS approach.

Algorithm	OFDM-FDMA	OFDM-TDMA	GOSM	EGAS	SAS
Complexity	$O(N)$	$O(K \cdot N^2)$	$O(K \cdot N^2)$	$O(N^2)$	$O(n_{iter} \cdot N)$

Table 5. Complexity of the different spectrum allocation approaches.

8. Applications of the greedy spectrum allocation approach in two case studies

8.1 Optimization of wireless sensors' autonomies by greedy spectrum allocation in uplink transmission

Wireless Sensor Networks have lately gained a great deal of attention in the areas of video transmission, surveillance, remote monitoring, etc. In these applications, a certain number of sensors transmit data simultaneously, on a shared medium, to a central base station. Therefore, the terminals batteries are highly solicited by the multimedia functionalities, the radio-frequency part, the real-time execution of increasingly complex algorithms and tasks, etc. Hence, stringent requirements have been put on the wireless terminal battery in order to offer a proper autonomy.
Efficient techniques of dynamic spectrum allocation can help improve the autonomy of wireless terminals, especially in the case of the uplink transmission, where the power amplifier particularly solicits the sensor's battery. For this reason, we propose to apply a greedy approach, similar to the one used in the downlink transmission, to determine the best assignment of subchannels in such a way to maximize the mobiles autonomies by efficiently managing their power consumption. This optimization will enhance the network lifetime defined as the duration of communication between mobiles before a failure occurs due to battery depletion.
Let:
P_{max} the maximum allowed power per user.
Δt the time duration over which the subchannel attribution scheme is valid (the transmission channel response is assumed to be constant over Δt)

E_k the battery level for the k^{th} terminal.
$P_{k,n}$ the power transmitted by user k on the subcarrier n.
The optimization problem is the following:
Minimization of the power consumption of the least privileged user, i.e. the user suffering from the weakest initial battery level or the poorest channel conditions:

$$\max_{P_{n,k},S_k} \min_{k} \left(E_k - \Delta t \sum_{n \in S_k} P_{n,k} \right)$$

under the following constraints:

$$\sum_{n \in S_k} \frac{B}{N} log_2 \left(1 + \frac{P_{k,n} H_{k,n}^{\ 2}}{N_0 \frac{B}{N}} \right) = R_k, \forall k$$

$$\sum_{n \in S_k} P_{n,k} \leq P_{max}, \forall k$$

$$P_{k,n} \geq 0, \forall k, \forall n \in S$$

$$S_i \bigcap_{i \neq j} S_j = \varnothing$$

$$\bigcup_{k=1}^{K} S_k \subseteq \{1,2,...N\}$$

A greedy solution for this optimization problem is summarized in Figure 10.

Greedy algorithm for subcarrier allocation in OFDM uplink transmission

$P_{k,n} = \inf(K,N)$ // Initialization of all subcarriers powers to infinity
while $(S \neq \phi)$
$k_c = \arg\max_k \sum_{n \in S_k} P_{k,n}$
If $\left(\sum_{n \in S_{k_c}} P_{k_c,n} \leq P_{max} \right)$
$k_c = \arg\min_k \left(E_k - \Delta t . \sum_{n \in S_k} P_{k,n} \right)$
End If
$n_c = \arg\max_{n \notin \bigcup_k S_k} H_{k_c,n}$
$S_{k_c} = S_{k_c} \cup \{n_c\}$
$S = S \cap \{n_c\}^C$
Water-filling on S_{k_c} constrained by : $\left| R_{k_c,tot} - R_{k_c} \right| \leq R_{tol}$
Estimation of $P_{k_c,n}$, $n \in S_{k_c}$
End while

Fig. 10. Description of the greedy subcarrier allocation algorithm in OFDM uplink transmission.

The proposed algorithm can be described as follows:
While the set S of available subcarriers is non empty, we first identify the user k_c with the highest necessary transmission power. If all users required powers respect the power constraint, k_c is then the user with the weakest battery level. In other words, the power constraint has a higher priority over the optimization of the users battery levels. The second step consists in identifying the most interesting subcarrier for user k_c, among all available subcarriers, and assigning it to user k_c. Finally, we determine the power allocation scheme for user k_c in order to reach its target bit rate with an absolute tolerable error R_{tol}. The power allocation for user k_c is realized by performing water-filling on its so far allocated subcarriers. This water-filling is performed using the gradual dichotomy-based approach described in section 6.
In (Farah & Marx, 2007), we propose several enhanced versions of this greedy uplink spectrum allocation approach. We also prove the high superiority of the greedy solution to the classical OFDM-FDMA approach. The gain in the power consumption of the least privileged user is considerable, especially when the number of sensors increases.

8.2 Multiuser video streaming using greedy dynamic channel allocation

Video streaming on demand is becoming a popular application in new generations of wireless and mobile systems. However, due to user mobility and random channel variations, existing networks cannot provide end-to-end quality of service (QoS) to a large number of users using traditional approaches of spectrum allocation. In a former work (Yaacoub et al., 2006; 2007), we showed how optimal rate allocation can improve the overall performance of a multiuser streaming system where a certain number of mobile users, sharing a certain data transmission rate, request different video sequences from a streaming server. This optimization is achieved by unequal error protection (UEP) of the compressed video streams using variable channel coding, in such a way to allocate appropriate transmission rates to users experiencing different channel conditions. This study was conducted with a limited number of users and by assuming Frequency Division Multiplexing of the different users. The results showed a significant improvement in the overall system performance compared to a traditional system where all users are allocated equal channel resources.
In the following, we propose a framework for the optimal distribution of channel resources and transmission power among a large number K of users downloading video sequences from a streaming server, in the context of an OFDM cognitive system. The application of our EGAS allocation approach will permit an optimization of the necessary total bandwidth as well as the users decoding quality.
All video sequences are supposed to be compressed and stored on the streaming server. Each sequence is divided in a certain number of Group Of Pictures (GOP), with an IPPP…P structure (i.e. a GOP consists of an intra-coded I frame followed by a fixed number of predicted P frames). We assume H.264 (ITU-T & ISO/IEC JTC1, 2003) video coding with an error-concealment strategy described as follows: if an error occurs in a decoded frame, this frame and all the following ones in the same GOP are replaced by the last correctly decoded frame.
As for UEP, it is realized by applying a set of different puncturing schemes (Rowitch & Milstein, 2000) to the same channel coder output, in such a way to vary the amount of redundancy bits used to protect different parts of the video stream. This amount will depend on the level of video motion in each part of the stream, as well as on the variable transmission channel conditions.

Let $R_{Ctot,k}$ be the total source coding rate corresponding to the GOP of a user k. $R_{Ctot,k}$ of each user is proportional to the size of the H264-compressed GOP, and therefore to the level of motion in this GOP.

At each stage of the greedy allocation algorithm (Figure 11) , the user k_c whose actual rate is the farthest from its target rate R_{Ctot,k_c} is first identified. This user is attributed the most favorable subcarrier. Then, water-filling is performed on all so-far allocated subcarriers, as in section 5, for users who have not already reached their target rates (i.e. users from the set

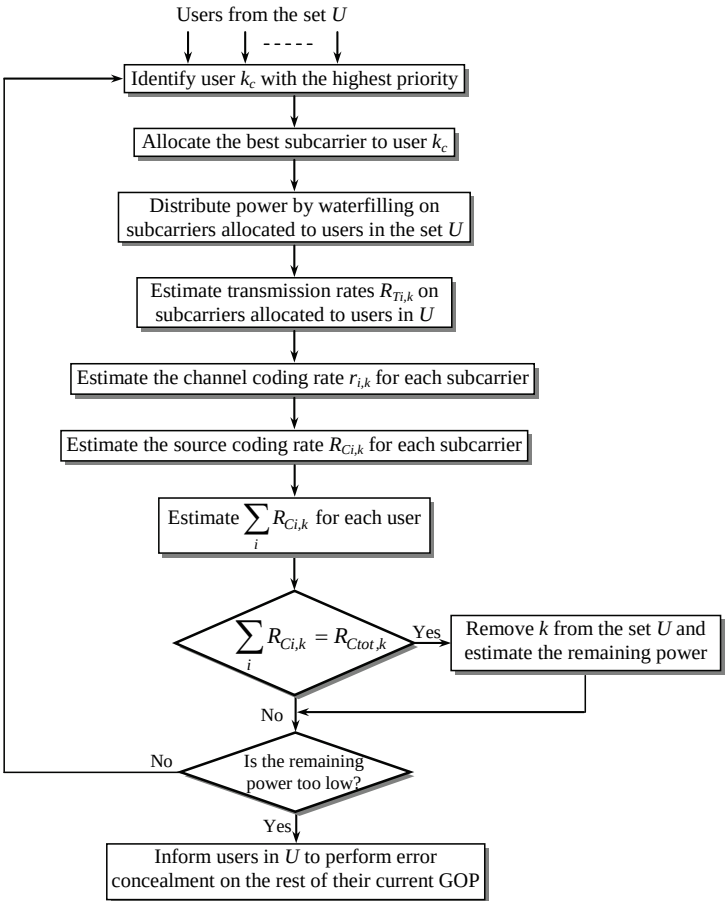


Fig. 11. Greedy channel allocation algorithm for multiuser video streaming in an OFDM cognitive system.

U). After water-filling, the actual transmission rate $R_{Ti,k}$ of each user k is estimated on each of its allocated subcarriers i. This data rate encloses the source coding rate $R_{Ci,k}$ of the GOP part of user k transmitted over the i^{th} subcarrier of user k, as well as the channel coding rate $r_{i,k}$ ($r_{i,kj}$ <1) necessary to achieve an almost correct decoding of the coded stream transmitted on this subcarrier (for ex, with a decoding Bit Error Rate of 10^{-6}):

$$R_{Ti,k} = R_{Ci,k} / r_{i,k} .$$

Note that $r_{i,k}$ depends on the subcarrier i transmission power, on the noise power spectral density N_0, and on the subcarrier attenuation $H_{i,k}$. It can be obtained using pre-determined performance curves of the particular channel coding scheme in use.

Therefore: $R_{Ci,k} = R_{Ti,k} \cdot r_{i,k}$.

A new iteration then begins by identifying the new user k_c such that: $k_c = \underset{k \in U}{arg\,max}\left(R_{Ctot,k} - \sum_i R_{Ci,k} \right)$. Note that in the first iteration: $k_c = \underset{k \in U}{arg\,max}\left(R_{Ctot,k} \right)$.

At the end of the iterative process, each user will have his GOP partitioned over a certain number of subcarriers. Besides, the protection level of each part against transmission errors is realized according to the channel state of each allocated subcarrier, thus achieving Unequal Error Protection of the user downloaded stream. Indeed, the first frames in each user GOP will be sent on the best subcarriers (subcarriers with the highest amplitudes), whereas the last ones will be sent on more attenuated subcarriers, but with a higher level of protection. Moreover, users who are treated at the end correspond to those whose GOP contains a low level of motion (i.e. users with a small $R_{Ctot,k}$). It can happen that, before the end of the iterative process, the remaining transmission power (distributed by water-filling) becomes insufficient to insure an acceptable error-protection of the last frames of a few users. In this case, the remaining power will be distributed on some of those frames and users who where not able to complete the download of their GOP are informed by the base station to perform error concealment on the remaining part of their GOP (i.e. replace the last few frames in the GOP by the last correctly received frame).

9. Conclusion

In this chapter, we proposed several greedy approaches for the optimization of spectral resources in a multiuser OFDM system. Through an optimal subcarrier allocation scheme, the total amount of occupied bandwidth can be decreased in the downlink transmission, while ensuring a certain quality of service for a large number of users. Our simulations show that our approaches permit considerable gains towards classical assignment methods based on either FDMA or TDMA techniques. In fact, with our enhanced greedy subcarrier allocation technique, the risk of failing to satisfy the users rate constraints is weak compared to that of the other techniques, especially when the users target rates increase. The achieved performance is almost similar to the one obtained with a quasi-optimum technique based on simulated annealing. However, the complexity of our proposed iterative algorithms is much lower than that of the Metropolis algorithm and certainly lower than the exhaustive exploration. Several application cases can benefit from the application of our greedy iterative approaches for spectrum allocation. For this reason, we proposed two general frameworks for the optimization of power consumption in a wireless sensor network and for the optimization of the decoding quality in the context of multiuser video streaming.

10. References

Burgeios F. & Lassalle J. C. (1971). An extension of Munkres algorithm for the assignment problem to rectangular matrices, *Commun. of the ACM*, vol. 14, pp. 802, 1971.

Cioffi J. M. (2008). Advanced Digital Communications, Available at: http://www.stanford.edu/class/ee379c.

Farah J. & Marx F. (2007). Combining Strategies for the Optimization of Resource Allocation in a Wireless Multiuser OFDM System, *AEU International Journal of Electronics and Communications*, Elsevier, 2007, Vol. 61, 2007, pp. 665-677.

FCC NOI and NPRM (2003). Establishment of an Interference Temperature Metric to Quantify and Manage Interference, *FCC-03-289*, November 2003.

FCC RO (2003). Promoting Efficient Use of Spectrum through Elimination of Barriers to the Development of Secondary Markets, *FCC 00-230*, October 2003.

Haykin S. (2001), *Communication Systems*, John Wiley and Sons, USA, 2001.

IEEE Standard 802.11a-1999, part 11. Wireless LAN Medium Access Control and Physical Layer Specifications.

ITU-T & ISO/IEC JTC1 (2003), Advanced Video Coding for Generic Audiovisual Services, *ITU-T Recommendation H.264 – ISO/IEC 14496-10 AVC*, 2003.

Kim K., Kim H., Han Y., and Kim S. L. (2004). Iterative and Greedy Resource Allocation in an Uplink OFDMA System, *Proc. International Symposium on Personal, Indoor and Mobile Radio Communications*, 2004, p. 2377-81.

Kirkpatrick S., Gelatt C.D., and Vecchi M.P. (1983). Optimization by simulated annealing, *Science*, May 1983, vol. 220, no. 4598, p. 671-80.

Kivanc D., Li G., and Liu H. (2003). Computationally Efficient Bandwidth Allocation and Power Control for OFDMA, *IEEE Trans. Wireless Communications*, vol. 2, no. 6, November 2003, p. 1150-58.

Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., and Teller E. (1953). Equations of state calculations by fast computing machines, *J. Chem. Phys.*, Vol. 21, 1953, pp. 1087-91.

Munkres J.R. (1957). Algorithms for the assignment and transportation problems, *J. Soc. Indust. Appl. Math.*, vol. 5, 1957, p. 32-8.

Rhee, W. & Cioffi , J. (2000). Increase in capacity of multiuser OFDM system using dynamic subchannel allocation, *Proc. Vehicular Technology Conference*, 2000, p. 1085-89.

Rohling H. & Grunheid R. (1997). Performance Comparison of Different Multiple Access Schemes for the Downlink of an OFDM Communication System, *Proc. Vehicular Technology Conference*, 1997, p. 1365-9.

Rowitch D.N. & Milstein L.B. (2000). On the performance of hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes, *IEEE Transactions on Communications*, pp. 948-959, Vol. 48, June 2000.

Toufik I. & Knopp R. (2004). Channel allocation algorithms for multi-carrier systems, *Proc. IEEE Vehicular Technology Conference*, September 2004, vol. 2, p. 1129-33.

Wong C. Y., Cheng R. S., Ben Letaief K., and Murch R. D. (1999). Multiuser OFDM with Adaptive Subcarrier, Bit, and Power Allocation, *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 10, October 1999, p. 1747-58.

Yaacoub C., Farah J., Rachkidy N., Marx F., Pesquet-Popescu B. (2006). Dynamic RCPT-Based Cross-Layer Optimization for Multi-User H.264 Video Streaming, *Proc. of 2nd International Computer Engineering Conference Engineering the Information Society, ICENCO'2006*, December 2006, Cairo, Egypt, pp. 65-70.

Yaacoub C., Farah J., Rachkidy N., Pesquet-Popescu B. (2007). A Cross-Layer Approach for Dynamic Rate Allocation in H.264 Multi-User Video Streaming, *Proc. of the 14th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2007*, Morocco, December 2007, pp.1244-1247.

13

Greedy Algorithms in Survivable Optical Networks

Xiaofei Cheng
Institute for Infocomm Research (I2R)
Singapore

1. Introduction

Greedy algorithm is a simple and straightforward heuristic algorithm which is usually used to solve optimization problems. It has been widely applied in communication networks. In this chapter, we focus on greedy algorithm and its applications in survivable optical networks. After introducing basic concept and design method of greedy algorithm, we build up a mathematic model and design a greedy algorithm to solve backup resource reservation problems for protection against multiple-link failures in survivable optical networks. The design of the greedy algorithm is introduced in detail. Computational complexity and performance of the greedy algorithm are analyzed. Advances in greedy algorithms are discussed.

2. Optimization problem

An optimization problem is a problem in which the object is to find, not just a solution, but the best solution from all feasible solutions. More formally, optimization problem refers to study of problems and find a solution which has the minimum or maximum objective function value by systematically choosing the values of real or integer variables within a feasible region while satisfying all the constraints. Optimization problems are made up of three basic ingredients: (a) an objective function which the object is to minimize or maximize; (b) a set of variables which affect the value of the objective function; (c) a set of constraints that allow the variables to take on certain values but exclude others. Constraints are sometime not necessary. Some optimization problems are unconstrained. However, most of optimization problems, in practice, do have some constraints. In mathematics, the optimization problem can be represented as follows:

$$\min f(x) \quad or \quad \max f(x)$$

$$\text{Subject to: } g(x) \geq 0$$

$$x \in D$$

where *f(x)* is the objective function. $g(x)$ represents constraint functions and often specified by a set of constraints, equalities or inequalities that the variable x has to satisfy. D is the

feasible region or search space. Typically, D is some subset of the Euclidean space and is a set with limited elements. The elements of D are called candidate solution or feasible solution. Such a formulation is an optimization problem and the feasible solution that minimizes or maximizes the objective function is the optimal solution.

3. Advanced algorithms

We can get the optimal solution by enumerating all possible candidate solutions and comparing the value of objective function. However, it is time-consuming and the computational time is sometime unacceptable for complicated optimization problems. When the general methods are not available to find the optimal solution in an acceptable computational time, advanced algorithms are required to solve these problems.
Two fundamental goals are needed to be considered to design advanced algorithms: (a) running time; (b) optimal solution quality. Heuristic algorithm refers to an algorithm that is constructed by intuition or prior experiences and the heuristic algorithm abandons one or both of these goals. For example, some heuristic algorithms give up finding the optimal solution for an improvement in run time. Advanced algorithms belong to heuristic algorithm. Recently, advanced algorithms develop very quickly. These advanced algorithms include tabu search, simulated annealing, genetic algorithms, neural networks algorithms, greedy algorithm, etc. Among these advanced algorithms, greedy algorithms are used to solve optimization problems and sometimes work very well. Greedy algorithms are very simple and straightforward. They are widely used as heuristic algorithms and sometimes are designed to embed into other heuristic algorithms to solve the optimization problem due to its rapid calculation speed and acceptable solution quality.

4. Greedy algorithm

Greedy algorithm is a heuristic algorithm which can be used to solve optimization problems. The principle of greedy algorithm is to recursively constructe a set of objectives and makes the choice that is the best at each step. It is a step-by-step recursive method for solving optimization problems. The basic idea behind greedy algorithms is to build large solutions up from smaller ones while keeping only the best immediate or local solution they find as they go along. That is why we call "Greedy". Greedy algorithm works in phases. At each phase, a decision is made that appears to be the best at the step, without regard for future consequences**.** Generally, this means that some local optimum is chosen whereas this will not lead to globe optimum at the end of the algorithm. A greedy algorithm sometime can find the overall or globally optimal solution at terminate for some optimization problems. It is because a locally optimal choice leads to a globally optimal solution. However, it does not mean that is always yield optimal solutions. In most cases, it finds less-than-optimal solutions and produces a suboptimal solution. Even more, many optimization problems cannot be solved correctly by greedy algorithms. However, the optimal solution is not always required for some optimization problems in practices. In these cases, simple and fast greedy algorithms are always good algorithms which are used to generate approximate answers, rather than using the more complicated algorithms generally required generating an exact answer.
Greedy algorithms have many advantages and thus are very attractive. The greedy algorithms employ simple strategies that are simple to implement and require minimal

amount of resources. They are fast and generally linear to quadratic and require little extra memory. They are shortsighted in their approach in the sense that they take decisions on the basis of information at hand without worrying about the effect these decisions may have in the future. They are easy to invent, easy to be implemented and most of the time quite efficient. Even when greedy algorithms do not produce the optimal solution, they often provide fast heuristics (non-optimal solution), and are often used in finding good approximations. More over, the greedy algorithm can be designed to embed into other heuristic algorithms. In such cases the greedy method is frequently the basis of other heuristic approaches. Greedy algorithms have been widely applied to solve optimization problems in communications networks, e.g., Dijkstra's algorithm for finding the shortest path; Kruskal's algorithm for finding a minimum-cost spanning tree and Prim's algorithm for finding a minimum-cost spanning tree, etc. Dynamic programming is a powerful technique, but it often leads to algorithms with higher than desired running times.

5. Survivability in optical communication networks

Optical communication networks are high-capacity telecommunications networks which use fiber as transmission media to increase transmission distance and make use of wavelength-division multiplexing (WDM) technologies to increase the network capacity. Optical communication networks can provide routing, grooming, and restoration capabilities at the wavelength level. With the explosive growth of the Internet and the emergence of new services, optical communication networks are being rapidly deployed to satisfy the high-speed transport capacity demand economically. Optical communication networks have become the primary transport networks. Recently, optical communication network employing dense wavelength division multiplexing (DWDM) technology has currently harnessed several Terabits/s bandwidths into one fiber. As the size and capacity of transport network increase rapidly, any failure in the transport network will result in a significant loss of data and service. Therefore, survivability for high-speed communication networks becomes critical. In following section, we will discuss the greedy algorithm applications in survivable optical networks for protection against multiple-link failure [1].

6. Greedy algorithm in multiple-link failure protection in survivable optical networks

In a survivable optical networks, we model the network as a graph G (N, L), where N and L are the set of nodes and the set of links respectively. A link in the graph G represents bidirectional fiber between two nodes. For each link, set the cost of the link equals to the link distance. F is the number of simultaneous link failures. In order to provide 100% guarantee for dynamic traffic, when a new connection request arrives at an ingress node, the ingress node would choose a working path (WP) and F link-disjoint backup paths (BP) between itself and the intended egress node from a pre-computed routing table. Let k denote the arrival sequence of a connection and w_k denote the bandwidth requirement of the k^{th} connection. We make use of the following notations to describe the problem and the proposed solution [1, 2].

A'_e : Set of connections whose working paths traverse link e.

A_e : Total bandwidth of all connections in A'_e.

B'_e: Set of connections whose backup paths traverse link *e*.

B_e: Backup bandwidth reserved on link *e*. Since sharing of protection bandwidth is allowed, $B_e \le \sum_{k \in B'_e} w_k$.

R_e: Residue bandwidth of link *e*; i.e., $R_e = C - A_e - B_e$ where C is the total capacity of link *e*.

S'^{b}_{a}: Set of connections whose working paths traverse link *a* and backup path traverse link *b*, i.e. $S'^{b}_{a} = A'_a \bigcap B'_b$.

S^{b}_{a}: Total bandwidth of all the connections in S'^{b}_{a}. This is the total bandwidth of all connections whose AP traverse link *a* and BP traverse link *b*.

$U'^{e}_{a,b,c,...n}$: Set of connections whose working paths transverse any of the links *a*, *b*, *c*… *n* and whose backup paths traverse link *e*. In case of all links *a, b, c … n* failure simultaneously, all connections in $U'^{e}_{a,b,c,...n}$ will be protected by link *e*.

$U^{e}_{a,b,c,...n}$:Total backup bandwidth needed on link *e* for all the connections in set $U'^{e}_{a,b,c,...n}$.

$C^{e}_{a,b,c,...n}$: Additional backup bandwidth needed on link *e* in order to use it as part of a BP for a new connection to protect against simultaneous failures of links *a*, *b*, *c*… *n*.

C_e : Additional bandwidth needed on link *e* when a new connection is established.

$W^{k,j}$: A variable that takes on the value of 1 if the j^{th} alternative route between the node pair of the k^{th} connection is selected as the AP of the connection; 0 otherwise.

$P^{k,j}$: A variable that takes on the value of 1 if the j^{th} alternative route between the node pair of the k^{th} connection is selected as the BP of the connection; 0 otherwise.

R(k): The number of alternative routers between the node pair of the k^{th} connection.

V (.): A function that returns the total bandwidth for all the connections in a set *S*, such that $V(S) = \sum_{k \in S} w_k$, where *S* is a set of connections.

Based on the above definition, we can get the following equations:

$$A_e = \sum_{k \in A'_e} w_k \tag{1}$$

$$S'^{b}_{a} = A'_a \bigcap B'_b \tag{2}$$

$$S^{b}_{a} = \sum_{k \in s^{b}_{a}} w_k \tag{3}$$

$$U'^{e}_{a,b,c,...n} = S'^{e}_{a} \bigcup S'^{e}_{b} \bigcup ... \bigcup S'^{e}_{n} \tag{4}$$

$$U^{e}_{a,b,c,...n} = \sum_{k \in U^{e}_{a,b,c,...n}} w_k \tag{5}$$

The mathematic model of the optimization problem is given as follows:

Objective: Minimize the total network bandwidth consumption:

$$Minimize \sum_{e \in L} [A_e + B_e] \tag{6}$$

where:

$$B_e = \max_{L_f \in L} (U^e_{L_1, L_2, \cdots L_n}) \tag{7}$$

Subject to:
Link capacity constraints:

$$A_e + B_e \le C \qquad \forall e \tag{8}$$

Traffic demand constraints:

$$\sum_{j=1}^{R(k)} W^{k,j} = 1; \sum_{j=1}^{R(k)} P^{k,j} = F \quad \forall k \tag{9}$$

Link-disjoint constraints:

$$W^{k,j} + P^{k,j} \le 1 \qquad \forall k, j \tag{10}$$

Integer flow constraints:

$$W^{k,j}; P^{k,j} \in \{0,1\} \qquad \forall k, j \tag{11}$$

The mathematical model described in this section is suitable for static traffic pattern. To expedite a fast on-line algorithm for dynamic traffic, we present a fast on-line Greedy heuristic algorithm. The design of the greedy algorithm is analyzed as follows.
In order to provide 100% guarantee for network connections, total backup bandwidth reserved on a link *e* should protect against random *F*-link simultaneous failure. Assume that *F*-link set is represented by $L_f = \{l_1, l_2, ..l_i..l_F \mid l_i \in L\}$. To protect against this *F*-link failure, the total backup bandwidth reserved on the link *e* is $(U^e_{l_1, l_2, ..l_i .., l_F})$. For random *F*-link failure case, the total backup bandwidth on link *e* must be enough to cater for the worst case and is therefore given by:

$$\begin{aligned} B_e &= \max_{\substack{l_i \in L \\ i=1,2,...F}} (U^e_{l_1, l_2, \cdots l_F}) \\ &= \max_{\substack{l_i \in L \\ i=1,2,...F}} \sum_{k \in S^e_{l_1} \cup S^e_{l_2} \cup ... \cup S^e_{l_F})} w_k \end{aligned} \tag{12}$$

When a new connection k_{new} (bandwidth requirement: $w_{k_{new}}$) arrives, the ingress node will choose a working path (WP) and *F* link-disjoint backup paths (BP) from a list of pre-computed routes between the source and the destination (We assume the network is *F*+1-

connected) to protect against *F* link failures. The backup bandwidth will be reserved on each link of the BPs of the new connection. The objective is to find a working path and *F* backup paths from pre-computed routing table with the minimal total additional bandwidth to carry the new connection:

$$\text{Minimize} \sum_{e \in WP \cup BP} C_e \tag{13}$$

For each link *e* of a candidate working path, an additional bandwidth $w_{k_{new}}$ is required to carry the new requested connection. For each link *e* of candidate backup paths, admission of connection k_{new} changes the sets A'_e, B'_e, $S'^{e}_{l_i}$ and $U'^{e}_{l_1,l_2,\cdots l_F}$. Since the total backup bandwidth on link *e* before the admission is B_e, to protect against the simultaneous failures of links $L_1, L_2, \ldots L_n$, the additional backup bandwidth needed on link *e* for setting up a new connection k_{new} is given by:

$$C^{e}_{L_1,L_2,\ldots L_n} = \begin{cases} 0 & \text{if } (U^{e}_{L_1,L_2,\ldots L_n} + w_{k_{new}}) < B_e \\ (U^{e}_{L_1,L_2,\ldots L_n}) + w_{k_{new}} - B_e & \text{if } B_e < (U^{e}_{L_1,L_2,\ldots L_n} + w_{k_{new}}) < R_e \\ \infty & \text{if } U^{e}_{L_1,L_2,\ldots L_n} + w_{k_{new}} - B_e > R_e \text{ or } e \in L_f \text{ or } R_e < w_{k_{new}} \end{cases} \tag{14}$$

Note that $C^{e}_{L_1,L_2,\ldots L_n} = \infty$ means that it is not feasible to set up the new connection with BP traverses link *e*. B_e is calculated according to the equation (12). The additional bandwidth on link *e* of a candidate backup path for protection against random *F* link-failure, $C_e (0 \le C_e \le w_{k_{new}})$ is given by:

$$C_e = \max_{\forall L_f \in L} C^{e}_{L_1,L_2,\ldots L_n} \tag{15}$$

Equation (12) ~ (15) gives a protection scheme for protect against random *F* (*F*>2) multiple-link failure scenarios. However, the computation time for algorithm in equation (12) is $O(L^F)$. The computation time increases exponentially with the number of simultaneous link faults number *F*. It is not suitable for on-line resource reservation for dynamic traffic. To solve this problem, we present a greedy algorithm described below to calculate B_e instead of equation (12). The detailed steps of greedy algorithm are described as follows:

- **STEP 1**: Set *j*=1. Find the relative connection set of link *e*, *S*'(*e*), as given by

$$S'(e) = \left\{ S'^{e}_{l_i} \mid l_i \in L, S'^{e}_{l_i} \neq \Phi \right\} \tag{16}$$

- **STEP 2**: Find in *S*′ (*e*) the $S'^{e}_{l_i}$ with maximum $V(S'^{e}_{l_i})$ and denote it as $S'_{\max}$.

  Calculate M (j) as given by:

$$M(j) = V(S'_{\max}) = \max_{S' \in S'(e)} V(S') \tag{17}$$

- **STEP 3**: Update S' (e) and every $S' \in S'(e)$ as follows:

$$S'(e) = S'(e) - \{S'_{\max}\} \tag{18}$$

$$S' = S' - S'_{\max}, \ \forall S' \in S'(e) \tag{19}$$

Where $A - B = A \bigcap \bar{B}$.

- **STEP 4**: Increment j. Repeat Step 2, 3 until M (1), M (2),… M (F) are found. We have then:

$$B_e = \sum_{j=1}^{F} M(j) \tag{20}$$

In equations (16) – (20), we iterate F times to get the F sequential maximal bandwidth required on link e for protect against random single link failure scenarios. After each iteration time, a connection set update operation is implemented to avoid iterate calculating a connection' bandwidth. The time complexity of the greedy algorithm (equations (16) – (20)) is O (LF).

In a centralized network, a control node will maintain the connection set, $S'^{e}_{l_i}$ $(\forall l_i, e \in L)$ information. When a dynamic connection arrives, the control node will assign an optimal working and F backup path and reserve backup bandwidth according to equations (13)-(20). The connection set, $S'^{e}_{l_i}$ $(\forall l_i, e \in L)$ will be updated. In a distributed network, each node will maintain the connection set, $S'^{e}_{l_i}$ $(\forall l_i, e \in L)$. When a dynamic connection arrives, the ingress node will assign an optimal working and F backup paths and reserve backup bandwidth according to equations (13)-(20). Then, the connection set $S'^{e}_{l_i}$ $(\forall l_i, e \in L)$ is updated and the updated information is broadcasted to all distributed nodes.

We simulate different routing and bandwidth allocation schemes on three network topologies: (a) a 8-node all connection network; (b) NJLATA network and (c) NSFNET network. The average node degrees $\bar{d}$ of the network topologies are 5, 4 and 3, respectively. We studied the following 4 algorithms: (1) greedy algorithm with alternate routing scheme (2) greedy algorithm with Fixed (Shortest path) routing scheme (3) ESPI algorithm [3, 4] and (4) NS: Shortest path with No backup bandwidth Sharing algorithm. We simulate these four algorithms with Matlab 6.2 version and 2.0GHZ CPU. In our simulation, connection request follows a Poisson process and has an exponential holding time. Connection request arrival is uniformly distributed over all node pairs and a total of 10^6 connection requests are generated. The bandwidth requirement of each demand varied randomly from 1 to 4 units. Simulation shows that our Greedy algorithm with alternate routing scheme has the least bandwidth consumption. Alternate routing is more superior in saving bandwidth than fixed routing. Greedy algorithms save more total bandwidth consumption (F=2, 12%; F=3, 17%; F=4, 22%) than the ESPI algorithm and save 32% of the total bandwidth consumption using NS algorithm. Our greedy algorithms have less blocking probability than other algorithms. Our greedy algorithms achieve superior performance both in total bandwidth consumption and blocking probability than other algorithms for protection against multiple-link failures.

7. Advances in greedy algorithms

Greedy algorithms are fast, simple straightforward and generally linear to quadratic. They employ simple strategies that are simple to implement and require minimal amount of resources. They are easy to invent, easy to implement and most of the time quite efficient. So, simple and fast greedy algorithms are always good algorithms to solve the optimization problems when the optimal solution is not required.

8. Reference

[1] Xiaofei Cheng, Xu Shao, Yixin Wang, "Multiple Link Failure Recovery in Survivable Optical Networks", Photonic Network Communications, pp. 159-164, 14 (2007), July 2007.

[2] Xiaofei Cheng, Teck Yoong Chai, Xu Shao, Yixin Wang, "Complementary Protection Under Double Link Failure for Survivable Optical Networks", IEEE GLOBECOM, OPN07-2, California, USA, 27 Nov.-1 Dec. 2006.

[3] Chunming Qiao, Dahai Xu,"Distributed Partial Information Management (DPIM) Scheme for Survivable Networks-Part I", INFOCOM 2002. Vol 1, pp. 302 –311, 2002.

[4] Murali Kodianalm,T.V.Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration," IEEE, INFOCOM 2000, pp.902~910,2000.

14

Greedy Algorithms to Determine Stable Paths and Trees in Mobile Ad hoc Networks

Natarajan Meghanathan
Jackson State University, Jackson, MS
United States of America

1. Introduction

A mobile ad hoc network (MANET) is a dynamic, resource-constrained, distributed system of independent and arbitrarily moving wireless nodes and bandwidth-constrained links. MANET nodes operate with limited battery charge and use a limited transmission range to sustain an extended lifetime. As a result, MANET routes are often multi-hop in nature and nodes forward the data for others. Based on their primary route selection principle, MANET routing protocols are classified into two categories (Meghanathan & Farago, 2004): minimum-weight based routing and stability-based routing protocols. The minimum-weight path among the set of available paths in a weighted network graph is the path with the minimum total weight summed over all its edges. The routing metrics generally targeted include: hop count, delay, energy consumption, node lifetime and etc. The stability-based routing protocols are aimed to minimize the number of route transitions and incur the least possible route discovery and maintenance overhead to the network.

A majority of the ad hoc routing protocols are minimum-weight based and are proposed to optimize one or more performance metrics in a greedy fashion without looking at the future. For example, the Dynamic Source Routing (DSR) protocol (Johnson et. al., 2001) instantaneously selects any shortest path that appears to exist and similarly the Ad hoc On-demand Distance Vector (AODV) protocol (Perkins & Royer, 1999) chooses the route that propagated the Route Request (RREQ) route discovery messages, with the lowest delay. To maintain optimality in their performance metrics, minimum-weight based routing protocols change their paths frequently and incur a huge network overhead. The stability-based routing protocols attempt to discover stable routes based on the knowledge of the past topology changes, future topology changes or a combination of both. Prominent within the relatively smaller class of stability-based routing protocols proposed in the literature include: Associativity-based Routing (ABR) (Toh, 1997), Flow-Oriented Routing Protocol (FORP) (Su et. al., 2001) and the Route-lifetime Assessment Based Routing (RABR) (Agarwal et. al., 2000) protocols. ABR selects paths based on the degree of association stability, which is basically a measure of the number of beacons exchanged between two neighbor nodes. FORP selects the route that will have the largest expiration time since the time of its discovery. The expiration time of a route is measured as the minimum of the predicted expiration time of its constituent links. RABR uses the average change in the received signal strength to predict the time when the received signal strength would fall below a critical

threshold. The stable path MANET routing protocols are distributed and on-demand in nature and thus are not guaranteed to determine the most stable routes (Meghanathan 2006d; Meghanathan 2007).

Stability is an important design criterion to be considered while developing multi-hop MANET routing protocols. The commonly used route discovery approach of flooding the route request can easily lead to congestion and also consume node battery power. Frequent route changes can also result in out-of-order data packet delivery, causing high jitter in multi-media, real-time applications. In the case of reliable data transfer applications, failure to receive an acknowledgement packet within a particular timeout interval can also trigger retransmissions at the source side. As a result, the application layer at the receiver side might be overloaded in handling out-of-order, lost and duplicate packets, leading to reduced throughput. Thus, stability is also important from quality of service (QoS) point of view too.

This chapter addresses the issue of finding the sequence of stable paths and trees, such that the number of path and tree transitions is the global minimum. In the first half of the chapter, we present an algorithm called *OptPathTrans* (Meghanathan & Farago, 2005) to determine the sequence of stable paths for a source-destination (*s-d*) communication session. Given the complete knowledge of the future topology changes, the algorithm operates on the greedy "look-ahead" principle: Whenever an *s-d* path is required at a time instant *t*, choose the longest-living *s-d* path from *t*. The sequence of long-living stable paths obtained by applying the above strategy for the duration of the *s-d* session is called the stable mobile path and it incurs the minimum number of route transitions. We quantify route stability in terms of the number of route transitions. Lower the number of route transitions, higher is the stability of the routing algorithm.

In the second half of the chapter, we show that the greedy look-ahead principle behind *OptPathTrans* is very general and can be extended to find a stable sequence of any communication structure as long as there is an underlying algorithm or heuristic to determine that particular communication structure. In this direction, we propose algorithm *OptTreeTrans* (Meghanathan, 2006c) to determine the sequence of stable multicast Steiner trees for a multicast session. The problem of determining the multicast Steiner tree is that given a weighted network graph $G = (V, E)$ where V is the set of vertices, E is the set of edges connecting these vertices and S, is a subset of set of vertices V, called the multicast group or Steiner points, we want to determine the set of edges of G that can connect all the vertices of S and they form a tree. It is very rare that greedy strategies give an optimal solution. Algorithms *OptPathTrans* and *OptTreeTrans* join the league of Dijkstra algorithm, Minimum spanning tree Kruskal and Prim algorithms (Cormen et. al., 2001) that have used greedy strategies, but yet give optimal solution. In another related work, we have also proposed an algorithm to determine the sequence of stable connected dominating sets for a network session (Meghanathan, 2006b).

The performance of algorithms *OptPathTrans* and *OptTreeTrans* have been studied using extensive simulations under two different scenarios: (1) Scenarios in which the complete knowledge of the future topology changes is available at the time of path/tree selection and (2) Scenarios in which the locations of nodes are only predicted for the near future and not exact. To simulate the second scenario, we consider a location prediction model called "Prediction with Uncertainty" that predicts the future locations of nodes at different time instants based on the current location, velocity and direction of travel of each node, even though we are not certain of the velocity and direction of travel in the future. Simulation

results illustrate that the algorithms *OptPathTrans* and *OptTreeTrans*, when run under the limited knowledge of future topology changes, yield the sequence of paths and trees such that the number of transitions is close to the minimum values obtained when run under the complete knowledge of future topology changes.

The rest of the chapter is organized as follows: In Section 2, we describe algorithm *OptPathTrans* to determine the stable mobile path, discuss its proof of correctness and run-time complexity. Section 3 illustrates the simulation results of *OptPathTrans* under the two scenarios of complete and limited knowledge of future topology changes. In Section 4, we explain algorithm *OptTreeTrans* to determine the stable mobile multicast Steiner tree, discuss its proof of correctness and run-time complexity. Section 5 illustrates the simulation results of *OptTreeTrans* under the two scenarios of complete and limited knowledge of future topology changes. In Section 6, we discuss the impact of the stability-hop count tradeoff on network resources and routing protocol performance. Section 7 concludes the chapter and discusses future work. Note that we use the terms 'path' and 'route' interchangeably throughout the chapter. They are the same.

2. Algorithm for the optimal number of path transitions

One could resort to flooding as a viable alternative at high mobility (Corson & Ephremides, 1995). But, flooding of the data packets will prohibitively increase the energy consumption and congestion at the nodes. This motivates the need for stable path routing algorithms and protocols in dynamically changing scenarios, typical to that of MANETs.

2.1 Mobile graph

A mobile graph (Farago & Syrotiuk, 2003) is defined as the sequence $G_M = G_1G_2 \ldots G_T$ of static graphs that represents the network topology changes over some time scale T. In the simplest case, the mobile graph $G_M = G_1G_2 \ldots G_T$ can be extended by a new instantaneous graph G_{T+1} to a longer sequence $G_M = G_1G_2 \ldots G_T G_{T+1}$, where G_{T+1} captures a link change (either a link comes up or goes down). But such an approach has very poor scalability. In this chapter, we sample the network topology periodically for every one second, which could, in reality, be the instants of data packet origination at the source. For simplicity, we assume that all graphs in G_M have the same vertex set (i.e., no node failures).

2.2 Mobile path

A *mobile path* (Farago & Syrotiuk, 2003), defined for a source-destination (*s*-*d*) pair, in a mobile graph $G_M = G_1G_2 \ldots G_T$ is the sequence of paths $P_M = P_1P_2 \ldots P_T$, where P_i is a static path between the same *s*-*d* pair in $G_i = (V_i, E_i)$, V_i is the set of vertices and E_i is the set of edges connecting these vertices at time instant t_i. That is, each static path P_i can be represented as the sequence of vertices $v_0v_1 \ldots v_l$, such that $v_0 = s$ and $v_l = d$ and $(v_{j-1},v_j) \in E_i$ for $j = 1,2, \ldots, l$. The timescale of t_T normally corresponds to the duration of an *s*-*d* session.

Let $w_i(P_i)$ denote the weight of a static path P_i in G_i. For additive path metrics, such as hop count and end-to-end delay, $w_i(P_i)$ is simply the sum of the link weights along the path. Thus, for a given *s*-*d* pair, if $P_i = v_0v_1 \ldots v_l$ such that $v_0 = s$ and $v_l = d$,

$$w_i(P_i) = \sum_{j=1}^{l} w_i(v_{j-1}, v_j) \tag{1}$$

For a given mobile graph $G_M = G_1G_2 \ldots G_T$ and *s-d* pair, the weight of a mobile path $P_M = P_1P_2 \ldots P_T$ is

$$w(P_M) = \sum_{i=1}^{T} w_i(P_i) + \sum_{i=1}^{T-1} C_{trans}(P_i, P_{i+1}) \quad (2)$$

where $C_{trans}(P_i, P_{i+1})$ is the transition cost incurred to change from path P_i in G_i to path P_{i+1} in G_{i+1} and is measured in the same unit used to compute $w_i(P_i)$.

2.3 Stable mobile path and minimum hop mobile path

The Stable Mobile Path for a given mobile graph and *s-d* pair is the sequence of static *s-d* paths such that the number of route transitions is as minimum as possible. A Minimum Hop Mobile Path for a given mobile graph and *s-d* pair is the sequence of minimum hop static *s-d* paths. With respect to equation (2), a Stable Mobile Path minimizes only the sum of the transition costs $\sum_{i=1}^{T-1} C_{trans}(P_i, P_{i+1})$ and a Minimum Hop Mobile Path minimizes only the term $\sum_{i=1}^{T} w_i(P_i)$, assuming unit edge weights. For additive path metrics and a constant transition cost, a dynamic programming approach to optimize the weight of a mobile path $w(P_M) = \sum_{i=1}^{T} w_i(P_i) + \sum_{i=1}^{T-1} C_{trans}(P_i, P_{i+1})$ has been proposed in (Farago & Syrotiuk, 2003).

2.4 Algorithm description

Algorithm *OptPathTrans* operates on the following greedy strategy: Whenever a path is required, select a path that will exist for the longest time. Let $G_M = G_1G_2 \ldots G_T$ be the mobile graph generated by sampling the network topology at regular instants $t_1, t_2, \ldots, t_T$ of an *s-d* session. When an *s-d* path is required at sampling time instant t_i, the strategy is to find a mobile sub graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots \cap G_j$ such that there exists at least one *s-d* path in $G(i, j)$ and no *s-d* path exists in $G(i, j+1)$. A minimum hop *s-d* path in $G(i, j)$ is selected. Such a path exists in each of the static graphs $G_i, G_{i+1}, \ldots, G_j$. If sampling instant $t_{j+1} \leq t_T$, the above procedure is repeated by finding the *s-d* path that can survive for the maximum amount of time since t_{j+1}. A sequence of such maximum lifetime static *s-d* paths over the timescale of a mobile graph G_M forms the Stabile Mobile *s-d* Path in G_M. The pseudo code of the algorithm is given in Fig. 1.

Input: $G_M = G_1G_2 \ldots G_T$, source *s*, destination *d*
Output: P_S // Stable Mobile Path
Auxiliary Variables: *i*, *j*
Initialization: $i=1$; $j=1$; $P_S = \Phi$

Begin *OptPathTrans*

1 **while** ($i \leq T$) do

2 Find a mobile graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots \cap G_j$ such that there exists at least one *s-d* path in $G(i, j)$ and {no *s-d* path exists in $G(i, j+1)$ or $j = T$}

3 $P_S = P_S \cup \{$ minimum hop *s-d* path in $G(i, j)\}$

4 $i = j + 1$

5 **end while**

6 **return** P_S

End *OptPathTrans*

Fig. 1. Pseudo code for algorithm *OptPathTrans*

2.5 Algorithm complexity and proof of correctness

In a mobile graph $G_M = G_1G_2 \ldots G_T$, the number of route transitions can be at most T. A path-finding algorithm will have to be run T times, each time on a graph of n nodes. If we use Dijkstra algorithm that has a worst-case run-time complexity of $O(n^2)$, where n is the number of nodes in the network, the worst-case run-time complexity of *OptPathTrans* is $O(n^2T)$. We use the proof by contradiction technique to prove the correctness of algorithm *OptPathTrans*. Let P_S (with m route transitions) be the mobile path generated by algorithm *OptPathTrans*. To prove m is optimal, we assume the contrary that there exists a mobile path $P_{S'}$ with m' route transitions such that $m' < m$. Let $epoch_S^1$, $epoch_S^2$, …., $epoch_S^m$ be the set of sampling time instants in each of which the mobile path P_S suffers no route transitions (refer Fig. 2). Similarly, let $epoch_{S'}^1$, $epoch_{S'}^2$, …, $epoch_{S'}^{m'}$ be the set of sampling time instants in each of which the mobile path $P_{S'}$ suffers no route transitions (refer Fig. 3).

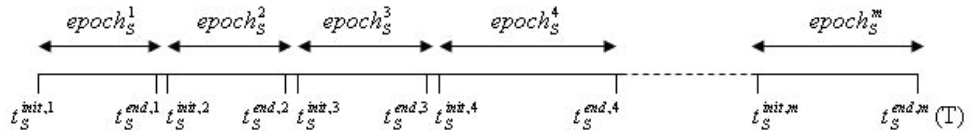


Fig. 2. Sampling Time Instants for Mobile Path P_S (Determined by *OptPathTrans*)

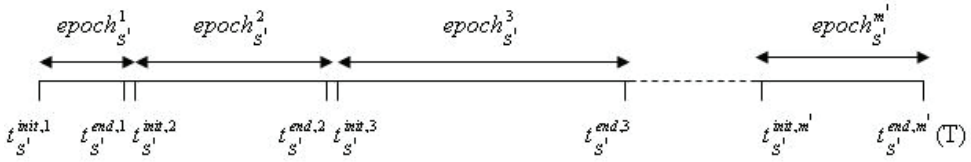


Fig. 3. Sampling Time Instants for Mobile Path $P_{S'}$ (Hypothesis for the Proof)

Let $t_S^{init,j}$ and $t_S^{end,j}$ be the initial and final sampling time instants of $epoch_s^j$ where $1 \le j \le m$. Similarly, let $t_{s'}^{init,k}$ and $t_{s'}^{end,k}$ be the initial and final sampling time instants of $epoch_{s'}^k$, where $1 \le k \le m'$. Note that $t_S^{init,1} = t_{s'}^{init,1}$ and $t_S^{end,m} = t_{s'}^{end,m'}$ to indicate that P_S and $P_{s'}$ span over the same time period, T, of the network session.

Now, since the hypothesis is $m' < m$, there should exist j, k where $1 \le j \le m$ and $1 \le k \le m'$ such that $epoch_s^j$ s $epoch_{s'}^k$, i.e., $t_{S'}^{init,k} < t_S^{init,j} < t_S^{end,j} < t_{S'}^{end,k}$ and at least one s-d path existed in $[t_{s'}^{init,k}, \ldots, t_{s'}^{end,k}]$. In other words, there should be at least one s-d path in $P_{s'}$ that has a lifetime larger than that of the lifetime of the s-d paths in P_s. But, algorithm *OptPathTrans* made a route transition at $t_S^{end,j}$ since there was no s-d path from $t_S^{init,j}$ beyond $t_S^{end,j}$. Thus, there is no common s-d path in the range $[t_S^{init,j}, \ldots, t_{s'}^{end,k}]$ and hence there is no common s-d path in the range $[t_{s'}^{init,k}, \ldots, t_{s'}^{end,k}]$. This shows that the lifetime of each of the s-d paths in $P_{s'}$ has to be smaller or equal to the lifetime of the s-d paths in P_s, implying $m' \ge m$. This is a contradiction and proves that our hypothesis $m' < m$ is not correct. Hence, the number of route transitions in P_s is optimal and P_s is the Stable Mobile Path.

2.6 Example run of algorithm *OptPathTrans*

Consider the mobile graph $G_M = G_1G_2G_3G_4G_5$ (Fig. 4.), generated by sampling the network topology for every second. Let node 1 and node 6 be the source and destination nodes respectively. The Minimum Hop Mobile 1-6 Path for the mobile graph G_M would be {{1-3-6}G_1, {1-4-6}G_2, {1-2-6}G_3, {1-3-6}G_4, {1-2-6}G_5}. As the minimum hop path in one static graph does not exist in the other, the number of route transitions incurred for the Minimum Hop Mobile Path is 5. The hop count in each of the static paths is 2 and hence the time averaged hop count would also be 2.

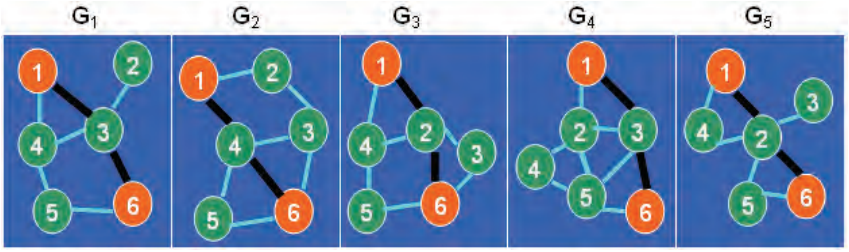


Fig. 4. Mobile Graph and Minimum Hop Mobile Path

The execution of algorithm *OptPathTrans* on the mobile graph G_M, of Fig. 4. is shown in Fig. 5. The Stable Mobile Path generated would be {{1-4-5-6}G_{123}, {1-2-5-6}G_{45}}. The number of route transitions is 2 as we have to discover a common path for static graphs G_1, G_2 and G_3 and a common path for static graphs G_4 and G_5. The hop count of each of the constituent paths of the Stable Mobile Path is 3 and hence the time averaged hop count of the Stable

Mobile Path would also be 3. Note that even though there is a 2-hop path {1-3-6} common to graphs G_1 and G_2, the algorithm ends up choosing the 3-hop path {1-4-5-6} that is common to graphs G_1, G_2 and G_3. This shows the greedy nature of algorithm *OptPathTrans*, i.e., choose the longest living path from the current time instant. To summarize, the Minimum Hop Mobile Path incurs 5 path transitions with an average hop count of 2; while the Stable Mobile Path incurs 2 path transitions with an average hop count of 3. This illustrates the tradeoff between stability and hop count which is also observed in the simulations.

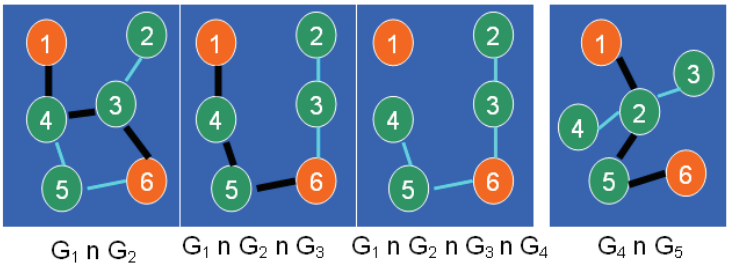


Fig. 5. Execution of Algorithm *OptPathTrans* on Mobile Graph of Fig. 4.

2.7 Prediction with uncertainty

Under the Prediction with Uncertainty model, we generate a sequence of predicted network topology changes starting from the time instant a path is required. We assume we know only the current location, direction and velocity of movement of the nodes and that a node continues to move in that direction and velocity. Whenever the node hits a network boundary, we predict it stays there, even though a node might continue to move. Thus, even though we are not sure of actual locations of the nodes in the future, we construct a sequence of predicted topology changes based on the current information. We run algorithm *OptPathTrans* on the sequence of predicted future topology changes generated starting from the time instant a path is required. We validate the generated path with respect to the actual locations of the nodes in the network. Whenever a currently used path is found to be invalid, we repeat the above procedure. The sequence of paths generated by this approach is referred to as Stable-Mobile-Path$^{\text{Uncertain-Pred}}$.

In practice, information about the current location, direction and velocity of movement could be collected as part of the Route-Request and Reply cycle in the route setup phase. After collecting the above information from each node, the source and destination nodes of a session assume that each node continues to move in its current direction of motion with the current velocity. Given the network dimensions (0… X_{max}, 0… Y_{max}), the location (x_i^t , y_i^t) of a node i at time instant t, the direction of motion Θ ($0 \le \Theta \le 360$) with reference to the positive x-axis, and the current velocity v_i^t , the location of node i at time instant $t + \delta t$, ($x_i^{t+\delta t}$, $y_i^{t+\delta t}$) would be predicted as follows:

$$x_i^{t+\delta t} = x_i^t + (v_i^t * \delta t * \cos\Theta) \qquad \text{if } 0 \le \Theta \le 90$$

$$= x_i^t - (v_i^t * \delta t * \cos(180 - \Theta)) \qquad \text{if } 90 \le \Theta \le 180$$

$$= x_i^t - (v_i^t * \delta t * \cos(\Theta - 180)) \qquad \text{if } 180 \le \Theta \le 270$$

$$= x_i^t + (v_i^t * \delta t * \cos(360 - \Theta)) \quad \text{if } 270 \le \Theta \le 360$$

$$y_i^{t+\delta t} = y_i^t + (v_i^t * \delta t * \sin\Theta) \quad \text{if } 0 \le \Theta \le 90$$

$$= y_i^t + (v_i^t * \delta t * \sin(180 - \Theta)) \quad \text{if } 90 \le \Theta \le 180$$

$$= y_i^t - (v_i^t * \delta t * \sin(\Theta - 180)) \quad \text{if } 180 \le \Theta \le 270$$

$$= y_i^t - (v_i^t * \delta t * \sin(360 - \Theta)) \quad \text{if } 270 \le \Theta \le 360$$

At any situation, when $x_i^{t+\delta t}$ is predicted to be less than 0, then $x_i^{t+\delta t}$ is set to 0.

when $x_i^{t+\delta t}$ is predicted to be greater than X_{max}, then $x_i^{t+\delta t}$ is set to X_{max}.

Similarly, when $y_i^{t+\delta t}$ is predicted to be less than 0, $y_i^{t+\delta t}$ is set to 0.

when $y_i^{t+\delta t}$ is predicted to be greater than Y_{max}, then $y_i^{t+\delta t}$ is set to Y_{max}.

When a source-destination (*s-d*) path is required at time instant *t*, we try to find the minimum hop *s-d* path in the predicted mobile sub graph $G^{pred}(t, t+\delta t) = G_t \cap G_{t+1}^{pred} \cap G_{t+2}^{pred} \cap \ldots \cap G_{t+\delta t}^{pred}$. If a minimum hop *s-d* path exists in $G^{pred}(t, t+\delta t)$, then that path is validated in the actual mobile sub graph $G^{actual}(t, t+\delta t) = G_t \cap G_{t+1} \cap G_{t+2} \cap \ldots \cap G_{t+\delta t}$ that spans time instants *t*, *t*+1, *t*+2, …, *t*+δt. If an *s-d* path exists in both $G^{pred}(t, t+\delta t)$ and $G^{actual}(t, t+\delta t)$, then that *s-d* path is used at time instants *t*, *t*+1, …, *t*+δt.

If an *s-d* path exists in $G^{pred}(t, t+\delta t)$, $G^{pred}(t, t+\delta t+1)$ and $G^{actual}(t, t+\delta t)$, but not in $G^{actual}(t, t+\delta t+1)$, the above procedure is repeated by predicting the locations of nodes starting from time instant *t*+δt+1. Similarly, if an *s-d* path exists in $G^{pred}(t, t+\delta t)$ and $G^{actual}(t, t+\delta t)$, but not in $G^{pred}(t, t+\delta t+1)$, the above procedure is repeated by predicting the locations of nodes starting from time instant *t*+δt+1. The sequence of paths obtained under this approach will be denoted as Stable-Mobile-Path$^{\text{Uncertain-Pred}}$ in order to distinguish from the Stable Mobile Path generated when future topology changes are completely known.

3. Simulation study of algorithm *OptPathTrans*

3.1 Simulation conditions

We ran our simulations with a square topology of dimensions 1000m x 1000m. The wireless transmission range of a node is 250m. The node density is varied by performing the simulations in this network with 50 (10 neighbors per node) and 150 nodes (30 neighbors per node). Note that, two nodes *a* and *b* are assumed to have a bidirectional link at time *t* if the Euclidean distance between them at time *t* (derived using the locations of the nodes from the mobility trace file) is less than or equal to the wireless transmission range of the nodes. We obtain a centralized view of the network topology by generating mobility trace files for 1000 seconds in the *ns*-2 network simulator (Bresalu et. al., 2000; Fall & Varadhan, 2001). Each data point in Fig. 6, 7, 8 and 9 is an average computed over 10 mobility trace files and 15 randomly selected *s-d* pairs from each of the mobility trace files. The starting time of each *s-d* session is uniformly randomly distributed between 1 to 20 seconds. The topology sampling interval to generate the mobile graph is 1 second.

3.2 Mobility model

We use the Random Waypoint mobility model (Betstetter et. al., 2004), one of the most widely used mobility simulating models for MANETs. According to this model, each node starts moving from an arbitrary location to a randomly selected destination with a randomly chosen speed in the range [v_{min} ... v_{max}]. Once the destination is reached, the node stays there for a pause time and then continues to move to another randomly selected destination with a different speed. We use v_{min} = 0 and pause time of a node is 0. The values of v_{max} used are 10 and 15 m/s (representing low mobility scenarios), 20 and 30 m/s (representing moderate mobility scenarios), 40 and 50 m/s (representing high mobility scenarios).

3.3 Performance metrics

The performance metrics evaluated are the number of route transitions and the time averaged hop count of the mobile path under the conditions described above. The time averaged hop count of a mobile path is the sum of the products of the number of hops per static path and the number of seconds each static path exists divided by the number of static graphs in the mobile graph. For example, if a mobile path spanning over 10 static graphs comprises of a 2-hop static path p_1, a 3-hop static path p_2, and a 2-hop static path p_3, with each existing for 2, 3 and 5 seconds respectively, then the time-averaged hop count of the mobile path would be (2*2 + 3*3 + 2*5) / 10 = 2.3.

3.4 Obtaining minimum hop mobile path

To obtain the Minimum Hop Mobile Path for a given simulation condition, we adopt the following procedure: When a minimum-hop path is required at time instant *t* and stability is not to be considered, the minimum-hop path Dijkstra algorithm is run on static graph at time instant *t*, and the minimum-hop path obtained is used as long as it exists. We repeat the above procedure until the end of the simulation time.

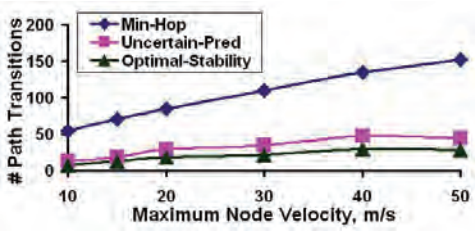


Fig. 6. Stability of Routes (50 Nodes)

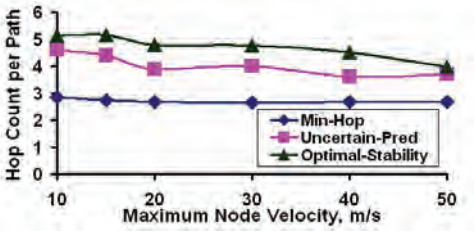


Fig. 7. Hop Count of Routes (50 Nodes)

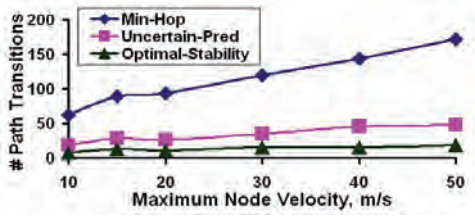


Fig. 8. Stability of Routes (150 Nodes)

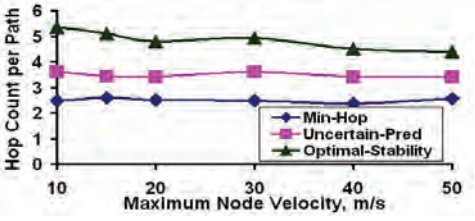


Fig. 9. Hop Count of Routes (150 Nodes)

3.5 Stability-hop count tradeoff

For all simulation conditions, the Minimum Hop Mobile Path incurs the maximum number of route transitions, while the average hop count per Minimum Hop Mobile Path is the least. On the other hand, the Stable Mobile Path incurs the minimum number of route transitions, while the average hop count per Stable Mobile Path is the maximum. The number of route transitions incurred by a Minimum Hop Mobile Path is 5 to 7 times to that of the optimal number of route transitions for a low-density network (refer Fig. 6) and 8 to 10 times to that of the optimal for a high-density network (refer Fig. 8). The average hop count per Stable Mobile Path is 1.5 to 1.8 times to that of the optimal hop count incurred in a low-density network (refer Fig. 7) and is 1.8 to 2.1 times to that of the optimal in a high-density network (refer Fig. 9). Optimality in both these metrics cannot be obtained simultaneously.

3.6 Impact of physical hop distance

The probability of a link (i.e., hop) failure increases with increase in the physical distance between the constituent nodes of the hop. We observed that the average physical distance between the constituent nodes of a hop at the time of a minimum-hop path selection is 70-80% of the transmission range of the nodes, accounting for the minimum number of intermediate nodes to span the distance between the source and destination nodes of the path. On the other hand, the average physical distance between the constituent nodes of a hop at the time of a stable path selection is only 50-55% of the transmission range of the nodes. Because of the reduced physical distance between the constituent nodes of a hop, more intermediate nodes are required to span the distance between the source and destination nodes of a stable path. Hence, the probability of failure of a hop in a stable path is far less compared to that of the probability of a failure of a hop in a minimum hop path. Also, the number of hops does not increase too much so that the probability of a path failure increases with the number of hops. Note that when we have a tie among two or more static paths that have the longest lifetime in a mobile sub graph, we choose the static path that has the minimum hop count to be part of the Stable Mobile Path.

3.7 Impact of node density

As we increase the node density, there are more neighbors per node, which increases the probability of finding a neighbor that is farther away. This helps to reduce the number of hops per path, but the probability of failure of the hop (due to the constituent nodes of the hop moving away) is also high. Thus, for a given value of v_{max}, minimum hop paths are more stable in low-density networks compared to high-density networks (compare Fig. 6 and Fig. 8). The average hop count of a Minimum Hop Mobile Path is more in a low-density network compared to that incurred in a high-density network (compare Fig. 7 and Fig. 9).

When we aim for stable *s-d* paths, we target paths that have low probability of failure due to the constituent nodes of a hop in the path moving away. With increase in node density, algorithm *OptPathTrans* gets more options in selecting the paths that can keep the source and destination connected for a longer time. In high density networks, we have a high probability of finding links whose physical distance is far less than the transmission range of the nodes. This is explored to the maximum by algorithm *OptPathTrans* and hence we observe a reduction in the number route transitions accompanied by an increase in the hop count in high-density networks compared to low-density networks.

3.8 Performance under the prediction with uncertainty model

The number of route transitions incurred by Stable-Mobile-Path$^{\text{Uncertain-pred}}$ is only at most 1.6 to 1.8 times that of the optimal for low-density networks (refer Fig. 6) and 2 to 3 times that of optimal for high-density networks (refer Fig. 8). Nevertheless, the average hop count incurred by Stable-Mobile-Path$^{\text{Uncertain-pred}}$ is 1.3–1.6 times to that incurred by Minimum-Hop-Mobile-Path (refer Fig. 7 and Fig. 9).

The mobility prediction model is practically feasible because the current location of each node, its direction and velocity can be recorded in the Route Request (RREQ) packets that get propagated from the source to destination during an on-demand route discovery. Rather than just arbitrarily choosing a minimum hop path traversed by the RREQ packet and sending a Route Reply (RREP) packet along that path, the destination node can construct a mobile sub graph by incorporating the locations of nodes in the near future, apply algorithm *OptPathTrans*, obtain the Stable-Mobile-Path$^{\text{Uncertain-Pred}}$ and send the RREP along that path.

4. Algorithm for the optimal number of multicast tree transitions

MANETs are deployed in applications such as disaster recovery, rescue missions, military operations in a battlefield, conferences, crowd control, outdoor entertainment activities, etc. One common feature among all these applications is one-to-many multicast communications among the participants. Multicasting is more advantageous than multiple unicast transmissions of the same data independently to each and every receiver, which also leads to network clogging. Hence, to support these applications in dynamic environments like MANETs, ad hoc multicast routing protocols that find a sequence of stable multicast trees are required.

4.1 Multicast steiner tree

Given a weighted graph, $G = (V, E)$, where V is the set of vertices, E is the set of edges and a subset of vertices (called the multicast group or Steiner points) $S \subseteq V$, the Steiner tree is the minimum-weight tree of G connecting all the vertices of S. In this chapter, we assume unit weight edges and that all the edges of the Steiner tree are contained in the edge set of the graph. Accordingly, we define the minimum Steiner tree as the tree with the least number of edges required to connect all the vertices in the multicast group (the set of Steiner points). Unfortunately, the problem of determining a minimum Steiner tree in an undirected graph like that of the unit disk graph is NP-complete. Efficient heuristics (e.g., Kou et. al., 1981) have been proposed in the literature to approximate a minimum Steiner tree.

4.2 Stable mobile multicast steiner tree vs minimum mobile multicast steiner tree

Aiming for the minimum Steiner tree in MANETs, results in multicast trees that are highly unstable. The multicast tree has to be frequently rediscovered, and this adds considerable overhead to the resource-constrained network. By adding a few more links and nodes to the tree, it is possible to increase its stability. We define stability of a multicast Steiner tree in terms of the number of times the tree has to change for the duration of a multicast session. Extending the greedy approach of *OptPathTrans* to multicasting, we propose an algorithm called *OptTreeTrans* to determine the minimum number of tree transitions incurred during the period of a multicast session for a multicast group comprising of a source node and a set of receiver nodes. Given the complete knowledge of future topology changes, the algorithm

operates on the following principle: Whenever a multicast tree connecting a given source node to all the members of a multicast group is required, choose the multicast tree that will keep the source connected to the multicast group members for the longest time. The above strategy is repeated over the duration of the multicast session and the sequence of stable multicast Steiner trees obtained by running this algorithm is called the Stable Mobile Multicast Steiner Tree. We use the Kou. et. al's (Kou et. al., 1981) well-known $O(|V||S|^2)$ heuristic, as the underlying heuristic to determine the longest existing multicast Steiner tree. A Minimum Mobile Multicast Steiner Tree is the sequence of approximations to the minimum Steiner tree obtained by directly using Kou's heuristic whenever required.

4.3 Heuristic to approximate minimum steiner tree

We use the Kou et. al's (Kou et. al., 1981) well-known $O(|V||S|^2)$ heuristic ($|V|$ is the number of nodes in the network graph and $|S|$ is the size of the multicast group) to approximate the minimum Steiner tree in graphs representing snapshots of the network topology. We give a brief outline of the heuristic in Fig. 10. An (s-S)-tree is defined as the multicast Steiner tree connecting a source node s to all the members of the multicast group S, which is also the set of Steiner points. Note that $s \in S$.

Input: An undirected graph $G = (V, E)$
Multicast group $S \subseteq V$
Output: A tree T_H for the set S in G

Step 1: Construct a complete undirected weighted graph $G_C = (S, E_C)$ from G and S where $\forall (v_i, v_j) \in E_C$, v_i and v_j are in S, and the weight of edge (v_i, v_j) is the length of the shortest path from v_i to v_j in G.
Step 2: Find the minimum weight spanning tree T_C in G_C (If more than one minimal spanning tree exists, pick an arbitrary one).
Step 3: Construct the sub graph G_S of G, by replacing each edge in T_C with the corresponding shortest path from G (If there is more than one shortest path between two given vertices, pick an arbitrary one).
Step 4: Find the minimal spanning tree T_S in G_S (If more than one minimal spanning tree exists, pick an arbitrary one). Note that each edge in G_S has weight 1.
Step 5: Construct the minimum Steiner tree T_H, from T_S by deleting edges in T_S, if necessary, such that all the leaves in T_H are members of S.

Fig. 10. Kou et. al's Heuristic (Kou et. al., 1981) to find an Approximate Minimum Steiner Tree

4.4 Algorithm *OptTreeTrans*

Let $G_M = G_1G_2 \ldots G_T$ be the mobile graph generated by sampling the network topology at regular instants $t_1, t_2, \ldots, t_T$ of a multicast session. When an (s-S)-tree is required at sampling time instant t_i, the strategy is to find a mobile sub graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots \cap G_j$ such that there exists at least one multicast (s-S)-tree in $G(i, j)$ and none exists in $G(i, j+1)$. A multicast (s-S)-tree in $G(i, j)$ is selected using Kou's heuristic. Such a tree exists in each of the static graphs $G_i, G_{i+1}, \ldots, G_j$. If there is a tie, the (s-S)-tree with the smallest number of constituent links is chosen. If sampling instant $t_{j+1} \leq t_T$, the above procedure is repeated by

finding the (*s*-*S*)-tree that can survive for the maximum amount of time since t_{j+1}. A sequence of such maximum lifetime multicast Steiner (*s*-*S*) trees over the timescale of G_M forms the Stable Mobile Multicast Steiner Tree in G_M. The pseudo code is given in Fig. 11.

Input: $G_M = G_1G_2 \ldots G_T$, source *s*, multicast group *S*
Output: $(s\text{-}S)_{MobileStabletree}$ // Stable-Mobile-Multicast-Steiner-Tree
Auxiliary Variables: *i*, *j*
Initialization: *i*=1; *j*=1; $(s\text{-}S)_{MobileStabletree} = \Phi$

Begin *OptTreeTrans*

1 **while** (i ≤ T) do

2 Find a mobile graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots \cap G_j$ such that there exists at least one (*s*-*S*)-tree in $G(i, j)$ and { no (*s*-*S*)-tree exists in $G(i, j+1)$ or $j = T$ }

3 $(s\text{-}S)_{MobileStabletree} = (s\text{-}S)_{MobileStabletree} \cup$ {Minimum Steiner (*s*-*S*)-tree in $G(i, j)$ }

4 $i = j + 1$

5 end while

6 **return** $(s\text{-}S)_{MobileStabletree}$

End *OptTreeTrans*

Fig. 11. Pseudo Code for Algorithm *OptTreeTrans*

4.5 Algorithm complexity and proof of correctness

In a mobile graph $G_M = G_1G_2 \ldots G_T$, the number of tree transitions can be at most *T*. The minimum Steiner tree Kou's heuristic will have to be run at most *T* times, each time on a graph of $|V|$ nodes. As Kou's heuristic is of $O(|V||S|^2)$ worst-case run-time complexity where $|S|$ is the size of the multicast group, the worst-case run-time complexity of *OptTreeTrans* is $O(|V||S|^2T)$.

Given a mobile graph $G_M = G_1G_2 \ldots G_T$, source node *s* and multicast group *S*, let the number of tree transitions in the Mobile Multicast Steiner Tree, $(s\text{-}S)_{MobileStabletree}$, generated by *OptTreeTrans* be *m*. To prove *m* is optimal, assume the contrary: there exists another Mobile Multicast Steiner Tree $(s\text{-}S)'_{MobileStabletree}$ in G_M and the number of tree transitions in $(s\text{-}S)'_{MobileStabletree}$ is $m' < m$.

Let $epoch_S^1, epoch_S^2, \ldots, epoch_S^m$ be the set of sampling time instants in each of which the Mobile Multicast Steiner Tree $(s\text{-}S)_{MobileStabletree}$ suffers no tree transitions. Let $epoch_{S'}^1, epoch_{S'}^2, \ldots, epoch_{S'}^{m'}$ be the set of sampling time instants in each of which the Mobile Multicast Steiner Tree $(s\text{-}S)'_{MobileStabletree}$ suffers no tree transitions. Let $t_{(s-S)}^{init,j}$ and $t_{(s-S)}^{end,j}$ be the

initial and final sampling time instants of $epoch_S^j$ where $1 \le j \le m$. Let $t_{(s-S)'}^{init,k}$ and $t_{(s-S)'}^{end,k}$ be the initial and final sampling time instants of $epoch_{S'}^k$ where $1 \le k \le m'$. Note that $t_S^{init,1} = t_{S'}^{init,1}$ and $t_S^{end,m} = t_{S'}^{end,m'}$ to indicate *(s-S)*$_{MobileStabletree}$ and *(s-S)'*$_{MobileStabletree}$ span over the same time period, *T*, of the network session.

Now, since we claim that $m' < m$, there should exist *j*, *k* where $1 \le j \le m$ and $1 \le k \le m'$ such that $epoch_S^j \subset epoch_{S'}^k$, i.e., $t_{S'}^{init,k} < t_S^{init,j} < t_S^{end,j} < t_{S'}^{end,k}$ and at least one (*s*-*S*)'-tree existed in $[t_{S'}^{init,k},\ldots,t_{S'}^{end,k}]$. In other words, there should be at least one (*s*-*S*)'-tree in *(s-S)'*$_{MobileStabletree}$ that has a lifetime larger than that of the lifetime of the (*s*-*S*)-trees in *(s-S)*$_{MobileStabletree}$. But, algorithm *OptTreeTrans* made a tree transition at $t_S^{end,j}$ since there was no (*s*-*S*)-tree from $t_S^{init,j}$ beyond $t_S^{end,j}$. Thus, there is no (*s*-*S*)-tree in the range $[t_S^{init,j}, \ldots, t_{S'}^{end,k}]$ and hence there is no (*s*-*S*)-tree in the range $[t_{S'}^{init,k},\ldots,t_{S'}^{end,k}]$. This shows that the lifetime of each of the (*s*-*S*)'-trees in *(s-S)'*$_{MobileStabletree}$ has to be smaller or equal to the lifetime of the (*s*-*S*)-trees in *(s-S)*$_{MobileStabletree}$, implying $m' \ge m$. This is a contradiction and proves that our hypothesis $m' < m$ is not correct. Hence, the number of tree transitions in *(s-S)*$_{MobileStabletree}$ is optimal and *(s-S)*$_{MobileStabletree}$ is the Stable Mobile Multicast Steiner Tree.

4.6 Example run of algorithm *OptTreeTrans*

Consider the mobile graph $G_M = G_1G_2G_3G_4G_5$ sampled every second (Fig. 12). Let node 1 be the source node and nodes 5 and 6 be the receivers of the multicast group. The Minimum Mobile Steiner Tree in G_M is {{1-3, 3-6, 5-6}G_1, {1-4, 4-6, 4-5}G_2, {1-2, 2-6, 5-6}G_3, {1-3, 3-6, 5-6}G_4, {1-2, 2-6, 2-5}G_5}. The edges of the constituent minimum Steiner trees in each of the static graphs are shown in dark lines. The number of tree transitions is 5 and the time averaged number of edges per Minimum Mobile Steiner Tree is 3 as there are three edges in each constituent minimum Steiner tree. The execution of algorithm *OptTreeTrans* on the mobile graph G_M is shown in Fig. 13. The Stable Mobile Steiner Tree formed is {{1-4, 4-3, 3-6, 4-5}G_{12}, {1-2, 2-3, 3-6, 2-4, 4-5}G_{345}}. The number of tree transitions is 2 and the time-averaged number of edges in the Stable Mobile Steiner Tree is (4*2 + 5*3)/5 = 4.6 as there are 4 edges in the stable Steiner tree common to graphs G_1 and G_2 and 5 edges in the stable Steiner tree common to G_3, G_4 and G_5. The simulation results also vindicate such tradeoff between the number of Steiner tree transitions and number of edges in the mobile Steiner tree.

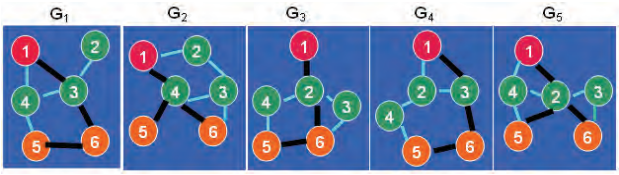


Fig. 12. Mobile Graph and Minimum-Mobile-Steiner-Tree

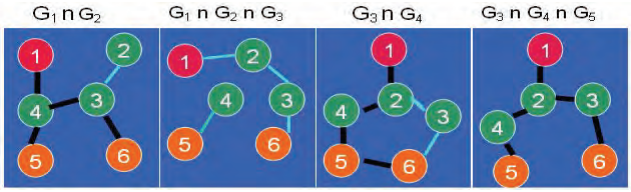


Fig. 13. Execution of Algorithm *OptTreeTrans* on Fig. 12.

5. Simulation study of algorithm *OptTreeTrans*

5.1 Simulation conditions

We ran our simulations with a square topology of dimensions 1000m x 1000m. The wireless transmission range of a node is 250m. The node density is varied by performing the simulations in this network with 50 (10 neighbors per node) and 150 nodes (30 neighbors per node). We obtain a centralized view of the network topology by generating mobility trace files for 1000 seconds in *ns*-2 (Bresalu et. al., 2000; Fall & Varadhan, 2001). Random waypoint mobility model is the mobility model used in the simulations. The range of node velocity values used are [0...10 m/s] and [0...50 m/s]. Each multicast group includes a source node and a set of receiver nodes. The multicast group size values are 2, 4, 8, 12, 18 and 24. Each data point in Fig. 14 through 21 is an average computed over 10 mobility trace files and 5 randomly selected groups for each size. The starting time of each multicast session is uniformly randomly distributed between 1 to 50 seconds and the simulation time is 1000 seconds. The topology sampling interval to generate the mobile graph is 1 second.

5.2 Minimum mobile multicast steiner tree and performance metrics

When an (*s*-*S*) Steiner tree is required at sampling time instant t_i and stability is not to be considered, then Kou's heuristic is run on static graph G_i and the (*s*-*S*) tree obtained is used as long as it exists. The procedure is repeated till the last sampling time instant t_T is reached. We refer to the sequence of multicast Steiner trees generated by the above strategy as Minimum Mobile Multicast Steiner Tree. The performance metrics evaluated are the number of tree transitions and the average number of edges in the mobile Steiner trees, which is the number of links in the constituent (*s*-*S*) Steiner trees, averaged over time.

5.3 Minimum mobile multicast steiner tree vs stable mobile multicast steiner tree

The number of multicast tree transitions increases rapidly with increase in multicast group size (refer Fig. 14, 16, 18 and 20). On the other hand, by accommodating 10-40% more edges (refer Fig. 15, 17, 19 and 21), stability of the Stable Mobile Multicast Steiner Tree is almost insensitive to multicast group size. For given value of v_{max}, the number of tree transitions incurred by the Minimum Mobile Multicast Steiner Tree in a low-density network (refer Fig. 14 and 18) is 5 (with group size of 4) to 10 (with group size of 24) times to that of the optimal. In high-density networks (refer Fig. 16 and 20), the number of tree transitions incurred by the Minimum Mobile Multicast Steiner Tree is 8 (with group size of 4) to 25 (with group size of 24) times to that of the optimal.

For a given node mobility and multicast group size, as the network density increases, algorithm *OptTreeTrans* makes use of the available nodes and links as much as possible in order to maximize the stability of the trees. For a Minimum Mobile Steiner Tree, the average

number of links in the constituent (*s*-*S*) trees is the same with increase in node density; the stability of the Minimum Mobile Steiner Trees decreases with increase in node density.

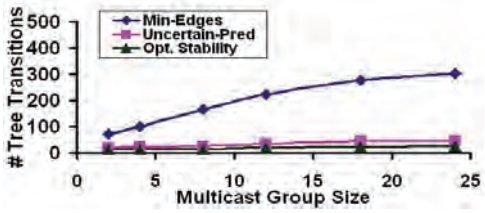


Fig. 14. Stability of Trees
(50 Nodes, v_{max} = 10 m/s)

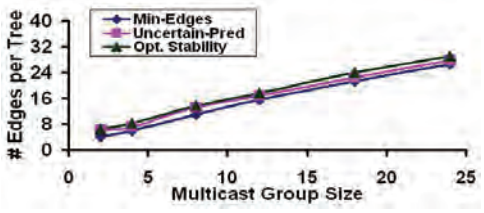


Fig. 15. Edges per Tree
(50 Nodes, v_{max} = 10 m/s)

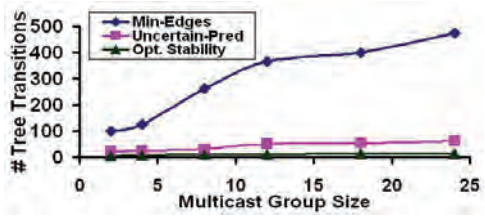


Fig. 16. Stability of Trees
(150 Nodes, v_{max} = 10 m/s)

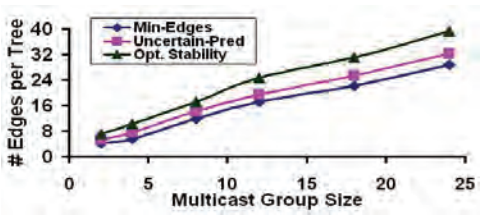


Fig. 17. Edges per Tree
(150 Nodes, v_{max} = 10 m/s)

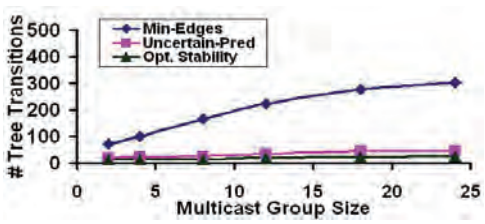


Fig. 18. Stability of Trees
(50 Nodes, v_{max} = 50 m/s)

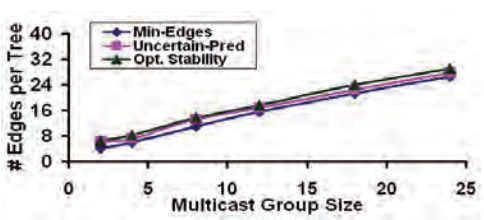


Fig. 19. Edges per Tree
(50 Nodes, v_{max} = 50 m/s)

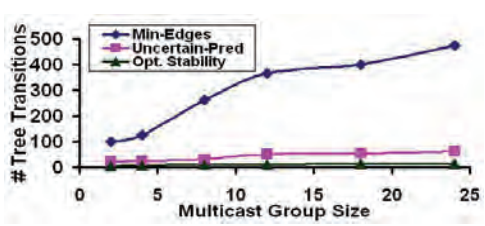


Fig. 20. Stability of Trees
(150 Nodes, v_{max} = 50 m/s)

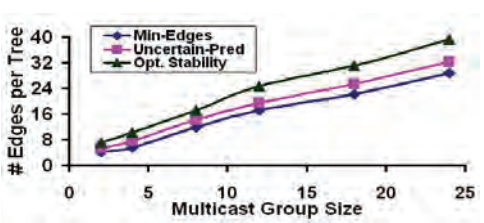


Fig. 21. Edges per Tree
(150 Nodes, v_{max} = 50 m/s)

For a given value of v_{max}, the number of tree transitions incurred by the Stable-Mobile-Multicast-Steiner-Tree$^{Uncertain\text{-}pred}$ in a low-density network (refer Fig. 14 and 18) is 1.6 (with group size of 2) to 2 (with group size of 24) times to that of the optimal, while in a high-density network (refer Fig. 16 and 20), the number of tree transitions is 3 (with group size of 2) to 4 (with group size of 24) times to that of the optimal. Thus, the stability of the constituent static trees in Stable-Mobile-Multicast-Steiner-Tree$^{Uncertain\text{-}pred}$ is not much affected by multicast group size and the number of edges (refer Fig. 15, 17, 19 and 21) is at most 40% more than that found in a Minimum Mobile Multicast Steiner Tree.

6. Impact of the stability-hop count tradeoff on network resources and routing protocol performance

We now discuss the impact of the stability-hop count tradeoff on network resources like the available node energy and the performance metrics like end-to-end delay per data packet.

6.1 Energy consumption

With increase in network density, a Stable Mobile Path incurs a reduced number of route transitions at the cost of an increased hop count; while a Minimum Hop Mobile Path incurs a reduced hop count per path at the cost of an increase in the number of route transitions. In (Meghanathan & Farago, 2006a), we analyzed the impact of these contradicting route selection policies on the overall energy consumption of a source-destination (*s*-*d*) session when using a Stable Mobile Path vis-à-vis a Minimum Hop Mobile Path for on-demand routing in MANETs. Some of the significant observations include:

- As we reduce the energy consumed per hop for data packet transfer (i.e., as we adopt reduced overhearing or no overhearing models), a Stable Mobile Path can bring significant energy savings than that obtained using a Minimum Hop Mobile Path.
- When data packets are sent continuously but at a reduced rate, we should use stable paths. If minimum hop paths are used, we may end up discovering a route to send every data packet, nullifying the energy savings obtained from reduced hop count.
- At high data packet rates (i.e., high data traffic), even a slight increase in the hop count can result in high energy consumption, especially in the presence of complete overhearing (also called as promiscuous listening). At high data traffic, energy spent in route discovery is overshadowed by the energy spent in data packet transfer.
- Route discovery is very expensive with respect to energy consumption in networks of high density compared to networks of low density. To minimize the overall energy consumption at moderate data traffic, we should use minimum-hop based routing at low network densities and stability-based routing at high network densities.

6.2 End-to-end delay per data packet

In (Meghanathan, 2008), we studied the performance of stable path routing protocols like ABR, FORP and RABR in *ns*-2 and measured the route stability and the end-to-end delay per data packet for *s*-*d* sessions running each of these three protocols. We observed a stability-hop count tradeoff within the class of stability-based routing protocols and the three protocols are ranked in the following increasing order of hop count: ABR, RABR and FORP; while in terms of the increasing order of the number of route transitions per *s*-*d* session, the ranking is: FORP, RABR and ABR. At low and moderate mobility conditions

(v_{max} <= 30 m/s), ABR routes incurred the lowest delay per packet compared to that of FORP. This could be attributed to the higher route relaying load on the nodes. Especially at high data traffic load, FORP routes incur significant delays due to MAC layer contention and queuing before transmission. RABR achieves a right balance between the route relaying load per node and the route discovery latency. RABR routes incur an end-to-end delay per packet that is close to that of ABR at low and moderate velocities and at the same time achieve stability close to that of the FORP routes. At high velocity, the buffering delay due to the route acquisition latency plays a significant role in increasing the delay of ABR routes and to a certain extent the RABR routes. Thus, at high node mobility conditions, all the three protocols incur end-to-end delay per packet that is close enough to each other.

7. Conclusions and future work

In this chapter, we described algorithms *OptPathTrans* and *OptTreeTrans* to determine respectively the sequence of stable paths (Stable Mobile Path) and multicast trees (Stable Mobile Multicast Steiner Tree) over the duration of a MANET session. Performance study of the two algorithms, when the complete knowledge of future topology changes is available at the time of path/tree selection, illustrates a distinct tradeoff between path hop count and the number of path transitions, and the number of edges in the multicast Steiner tree and the number of multicast Steiner tree transitions. It is highly impossible to simultaneously achieve optimality in the above mentioned contrasting performance metrics for paths and trees. The sequence of stable paths and trees generated by the two algorithms under the "Prediction with Uncertainty" model are highly stable compared to their minimum mobile versions. Also, the hop count, the number of edges and the number of nodes in the stable paths and trees is not as high as that observed in the stable mobile paths and trees obtained when the algorithms are run with complete knowledge of the future topology changes.

Note that the Dijkstra algorithm and the Kou et. al heuristic are merely used as a tool to find the appropriate stable communication structures. The optimal number of route and tree reconstructions does not depend on these underlying algorithms as we try to find the longest living route and tree in the mobile sub graph spanning a sequence of static graphs. But, the run-time complexity of the two algorithms depends on the underlying algorithm used to determine the Stable Mobile Path and the Stable Mobile Multicast Steiner Tree.

Future work is on the following: (i) To develop distributed versions of *OptPathTrans* and *OptTreeTrans* by extending these algorithms respectively as unicast and multicast routing protocols, (ii) To study the performance of algorithms *OptPathTrans* and *OptTreeTrans* under other MANET mobility models like Random Walk, Random Direction and Gauss-Morkov models (Camp et. al., 2002) and (iii) To develop various location-update and mobility prediction mechanisms to gather and/or distribute knowledge of future topology changes.

8. References

Agarwal, S.; Ahuja, A.; Singh, J. P. & Shorey, R. (2000). Route-Life Time Assessment Based Routing Protocol for Mobile Ad hoc Networks, *Proceedings of the IEEE International Conference on Communications*, pp. 1697-1701, ISBN: 0780362837, June 2000, New Orleans, LA, USA.

Bettstetter, C.; Hartenstein, H. & Perez-Costa, X. (2004). Stochastic Properties of the Random Way Point Mobility Model. *Wireless Networks*, Vol. 10, No. 5, (September 2004), pp. 555-567, ISSN: 10220038.

Breslau, L.; Estrin, D.; Fall, K.; Floyd, S.; Heidemann, J.; Helmy, A.; Huang, P.; McCanne, S.; Varadhan, K.; Xu, Y.; Yu, H. (2000). Advances in Network Simulation. *IEEE Computer*, Vol. 33, No. 5 (May 2000), pp. 59-67, ISSN: 00189162.

Camp, T.; Boleng, J. & Davies, V. (2002). A Survey of Mobility Models for Ad Hoc Network Research, *Wireless Communication and Mobile Computing*, Vol. 2, No. 5, (September 2002), pp. 483-502, ISSN: 15308669.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. (2001). *Introduction to Algorithms*, 2nd Edition, MIT Press, ISBN: 0262032937.

Corson, M. S. & Ephremides, A. (1995). A Distributed Routing Algorithm for Mobile Wireless Networks. *Wireless Networks*, Vol. 1, No. 1, (February 1995), pp. 61-81, ISSN: 10220038.

Fall, K. & Varadhan, K. (2001). ns notes and documentation. The VINT Project at LBL, Xerox PARC, UCB, and USC/ISI, http://www.isi.edu/nsnam/ns, August 2001.

Farago, A. & Syrotiuk, V. R. (2003). MERIT: A Scalable Approach for Protocol Assessment. *Mobile Networks and Applications*, Vol. 8, No. 5, (October 2003), pp. 567 – 577, ISSN: 1383-469X.

Johnson, D. B.; Maltz, D. A. & Broch, J. (2001). DSR: The Dynamic Source Routing Protocol for Multi-hop Wireless Ad hoc Networks, In: Ad hoc Networking, Charles E. Perkins, (Ed.), 139 – 172, Addison Wesley, ISBN: 0201309769.

Kou, L.; Markowsky, G. & Berman, L. (1981). A Fast Algorithm for Steiner Trees. *Acta Informatica*, Vol. 15, No. 2, (June 1981), pp. 141-145, ISSN: 0001-5903.

Meghanathan, N. & Farago, A. (2004). Survey and Taxonomy of 55 Unicast Routing Protocols for Mobile Ad hoc Networks, Technical Report UTD-CSC-40-04, The University of Texas at Dallas, Richardson, TX, November 2004.

Meghanathan, N. & Farago, A. (2005). An Efficient Algorithm for the Optimal Number of Route Transitions in Mobile Ad hoc Networks. *Proceedings of the 1st IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Vol. 3, pp. 41-48, ISBN: 0780391810, August 2005, Montreal, Canada.

Meghanathan, N. & Farago, A. (2006a). Comparison of Routing Strategies for Minimizing Energy Consumption in Mobile Ad Hoc Networks. *Proceedings of the 4th Asian International Mobile Computing Conference*, pp. 3-11, ISBN: 0070608342, January 2006, Kolkatta, India.

Meghanathan, N. (2006b). An Algorithm to Determine the Sequence of Stable Connected Dominating Sets in Mobile Ad Hoc Networks, *Proceedings of 2nd Advanced International Conference on Telecommunications*, ISBN: 0769525229, February 2006, Guadeloupe, French Caribbean.

Meghanathan, N. (2006c). Determining a Sequence of Stable Multicast Steiner Trees in Mobile Ad hoc Networks. *Proccedings of the 44th ACM Southeast Conference*, pp. 102-106, ISBN: 1595933158, March 2006, Melbourne, FL, USA.

Meghanathan, N. (2006d). A Simulation Study on the Stability-Oriented Routing Protocols for Mobile Ad hoc Networks. *Proceedings of the IFIP International Conference on*

Wireless and Optical Communication Networks, ISBN: 1424403405, April 2006, Bangalore, India.

Meghanathan, N. (2007). Path Stability based Ranking of Mobile Ad hoc Network Routing Protocols. *ISAST Transactions Journal on Communications and Networking*, Vol. 1, No. 1, (August 2007), pp. 66-73, ISSN: 17970989.

Meghanathan, N. (2008). Exploring the Stability-Energy Consumption-Delay-Network Lifetime Tradeoff of Mobile Ad hoc Network Routing Protocols. Journal of Networks, Vol. 3, No. 2, (February 2008), pp. 17-28, ISSN: 17962056.

Perkins, C. E. & Royer, E. M. (1999). Ad hoc On-demand Distance Vector Routing, *Proceedings of the Second Annual IEEE International Workshop on Mobile Computing Systems and Applications*, pp. 90–100, ISBN: 0769500250, February 1999, New Orleans, LA, USA.

Su, W.; Lee, S.-J. & Gerla, M. (2001). Mobility Prediction and Routing in Ad hoc Wireless Networks. *International Journal of Network Management*, Vol. 11, No. 1, (Jan-Feb. 2001), pp. 3-30, ISSN: 10991190.

Toh, C.-K. (1997). Associativity-Based Routing for Ad hoc Mobile Networks. *IEEE Personal Communications*, Vol. 4, No. 2, (March 1997), pp. 103 – 109, ISSN: 10709916.

15

Greedy Anti-Void Forwarding Strategies for Wireless Sensor Networks

Wen-Jiunn Liu and Kai-Ten Feng
Department of Communication Engineering, National Chiao Tung University
Taiwan, R.O.C.

1. Introduction

A wireless sensor network (WSN) consists of sensor nodes (SNs) with wireless communication capabilities for specific sensing tasks. Each SN maintains connectivity and exchanges messages between the decentralized nodes in the multi-hop manners. A source node can communicate with its destination via a certain number of relaying nodes, which consequently enlarges the wireless coverage of the source node. In conventional multi-hop routing algorithms, either the proactive or reactive schemes, significant amounts of routing tables and control packets are required for the construction of routing paths. Due to the limited available resources, efficient design of localized multi-hop routing protocols (Estrin et al., 1999) becomes a crucial subject within the WSNs. How to guarantee delivery of packets is considered an important issue for the localized routing algorithms. The well-known greedy forwarding (GF) algorithm (Finn, 1987) is considered a superior localized scheme with its low routing overheads, which is fit for conducting the routing task of WSNs. However, the void problem (Karp & Kung, 2000) that occurs within the GF technique will fail to guarantee the delivery of data packets.

Several routing algorithms are proposed to either resolve or reduce the void problem, which can be classified into non-graph-based and graph-based schemes. In the non-graph-based algorithms, the intuitive schemes as proposed in the research work (Stojmenovi´c & Lin, 2001) construct a two-hop neighbor table for implementing the GF algorithm. The network flooding mechanism is adopted while the void problem occurs. There also exist routing protocols that adopt the backtracking method at the occurrence of the network holes, such as GEDIR (Stojmenović & Lin, 2001), DFS (Stojmenović et al., 2000), and SPEED (He et al., 2003). The routing schemes as proposed by ARP (Giruka & Singhal, 2005) and LFR (Liu & Feng, 2006) memorize the routing path after the void problem takes place. Moreover, other routing protocols, such as PAGER (Zou & Xiong, 2005), NEAR (Arad & Shavitt, 2006), DUA (Chen et al., 2006), and YAGR (Na et al., 2007), propagate and update the information of the observed void node in order to reduce the probability of encountering the void problem. By exploiting these routing algorithms, however, the void problem can only be either (i) partially alleviated or (ii) resolved with considerable routing overheads and significant converging time.

On the other hand, there are research works on the design of graph-based routing algorithms to deal with the void problem. Several routing schemes as surveyed in the literature (Frey & Stojmenović, 2006) adopt the planar graph (West, 2000) as their network

topologies, such as GPSR (Karp & Kung, 2000), GFG (Bose et al., 2001), Compass Routing II (Kranakis at al., 1999), GOAFR+ (Kuhn at al., 2003), GOAFR++ (Kuhn at al., 2003), and GPVFR (Leong at al., 2005). Nevertheless, the usage of the planar graphs has significant pitfalls due to the removal of communication links leading to the sparse network link distribution; while the adoption of the unit disk graph (UDG) for modeling the underlying network is suggested. A representative UDG-based greedy routing scheme, i.e. the BOUNDHOLE algorithm (Fang at al., 2004), forwards the packets around the network holes by identifying the locations of the holes. However, the delivery of packets cannot be guaranteed in the BOUNDHOLE scheme even if a route exists from the source to the destination node.

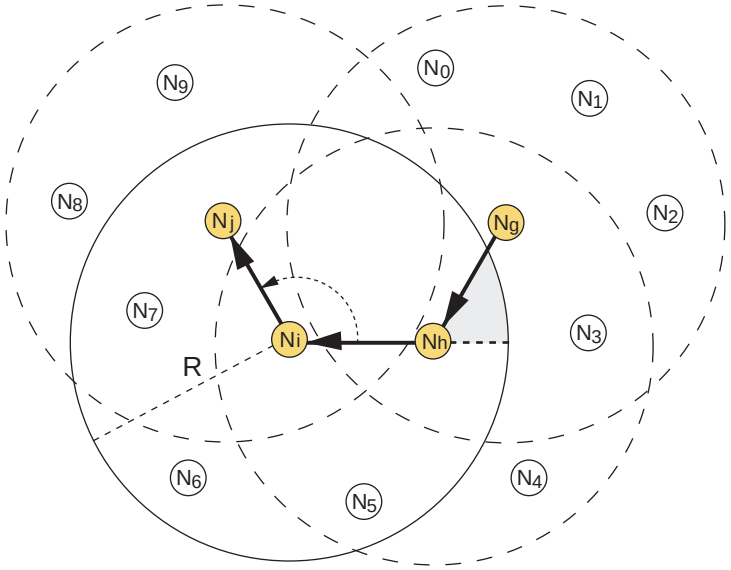


Fig. 1. The forbidden region and the minimal sweeping angle criterion of the BOUNDHOLE algorithm: The node N_i determines the next-hop node of the packets based on the previous two hops N_h and N_g. The forbidden region is defined as the area bounded by (i) the two backward-extended edges of E_{gh} and E_{hi} and (ii) the transmission range border, i.e. the grey region. The node N_j is selected as the next-hop node of N_i since it has the minimal sweeping angle from the previous hop N_h.

In the beginning, the principle of the BOUNDHOLE routing algorithm is briefly described. As shown in Fig. 1, the node N_i is conducting the routing tasks of the packets based on the previous two hops N_h and N_g. The BOUNDHOLE algorithm adopts the forbidden region and the minimal sweeping angle criterion within its formulation. The forbidden region is defined as the area bounded by (i) the backward-extended edges of E_{gh} and E_{hi} and (ii) the transmission range border, i.e. the grey region as in Fig. 1. All nodes in the forbidden region are not considered as the next-hop of N_i. The criterion of the minimal sweeping angle from the previous hop is utilized in the determination of the next-hops within the BOUNDHOLE algorithm. For example, as shown in Fig. 1, the node N_j, which is not in the forbidden region, has the minimal sweeping angle from the previous node N_h. The node N_j is therefore selected as the next-hop node of N_i.

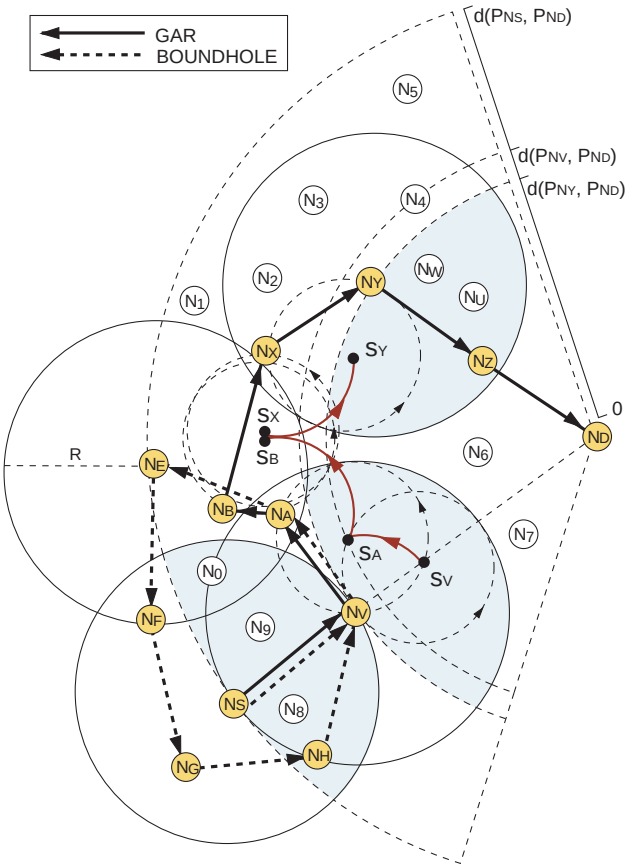


Fig. 2. The example routing paths constructed by using the GAR and the BOUNDHOLE algorithms under the existence of the void problem: (N_S, N_D) is the transmission pair and N_V is the void node. N_X is within the transmission range of N_B; while it is out of the range of N_A and N_E. The GAR protocol utilizes the RUT scheme (with red solid arcs denoted as the trajectory of the SPs); while the minimal angle criterion is employed by the BOUNDHOLE algorithm. The resulting paths obtained from these two schemes are {N_S, N_V, N_A, N_B, N_X, N_Y, N_Z, N_D} using the GAR protocol and {N_S, N_V, N_A, N_E, N_F, N_G, N_H, N_V} by adopting the BOUNDHOLE algorithm, which is observed to be undeliverable. The blue-shaded region associated with each SN is utilized to determine if the SN is a void node or not.

For the comparison purposes, the BOUNDHOLE algorithm is further investigated via an illustrative example. As shown in Fig. 2, the nodes (N_S, N_D) are considered the transmission pair; while N_V represents the node that the void problem occurs. In this example, it is assumed that the node N_X is located within the transmission range of N_B; while it is considered out of the transmission ranges of nodes N_A and N_E. Based on the minimal sweeping angle criterion within the BOUNDHOLE algorithm, N_A will choose N_E as its next hopping node since the counter-clockwise sweeping from N_V to N_E (hinged at N_A) is smaller comparing with that from N_V to N_B. Therefore, the resulting path by adopting the

BOUNDHOLE scheme becomes {N_S, N_V, N_A, N_E, N_F, N_G, N_H, N_V}. It is observed that the undeliverable routing path from the source node N_S is constructed even with un-partitioned network topology. Moreover, two cases of edge intersections within the BOUNDHOLE algorithm result in high routing overhead in order to identify the network holes.

In this book chapter, the greedy anti-void routing (GAR) protocol is proposed to resolve the void problem by exploiting the boundary finding technique under the UDG-based network topology. The proposed rolling-ball UDG boundary traversal (RUT) scheme is also employed to completely guarantee the delivery of packets from the source to the destination nodes. Moreover, the hop count reduction (HCR) and the intersection navigation (IN) mechanisms are incorporated within the GAR protocol (denoted as the GAR-E algorithm) to further improve the routing efficiency and the communication overhead. The proofs of correctness for the GAR scheme are also given in this book chapter. Comparing with the existing anti-void routing algorithms, the simulation results show that the proposed GAR-based protocols can provide better routing efficiency with guaranteed packet delivery.

The remainder of this book chapter is organized as follows. Section 2 describes the network model and the problem statement. The proposed GAR protocol is explained in Section 3; while Section 4 exploits the two enhanced mechanisms, i.e. the hop count reduction (HCR) and the intersection navigation (IN) schemes. The performance of the GAR-based protocols is evaluated and compared in Section 5. Section 6 draws the conclusions.

2. Network model and problem statement

Considering a set of SNs $\mathbf{N} = \{N_i \mid \forall i\}$ within a two-dimensional Euclidean plane, the locations of the set **N**, which can be acquired by their own positioning systems, are represented by the set $\mathbf{P} = \{P_{N_i} \mid P_{N_i} = (x_{N_i}, y_{N_i}), \forall i\}$. It is assumed that all the SNs are homogeneous and equipped with omni-directional antennas. The set of closed disks defining the transmission ranges of **N** is denoted as $\overline{\mathbf{D}} = \{\overline{D}(P_{N_i}, R) \mid \forall i\}$, where $\overline{D}(P_{N_i}, R) = \{x \mid \|x - P_{N_i}\| \le R, \forall x \in \mathbf{R}^2\}$. It is noted that $\mathbf{R}^2$ presents the two-dimensional real vector space and P_{N_i} is the center of the closed disk with R denoted as the radius of the transmission range for each N_i. Therefore, the underlying network model for the WSNs can be represented by a unit disk graph (UDG) as G(**P**, **E**) with the edge set $\mathrm{E} = \{E_{ij} \mid E_{ij} = (P_{N_i}, P_{N_j}), P_{N_i} \in \overline{D}(P_{N_j}, R), \forall i \neq j\}$. The edge E_{ij} indicates the unidirectional link from P_{N_i} to P_{N_j} whenever the position P_{N_i} is within the closed disk region $\overline{D}(P_{N_j}, R)$. Moreover, the one-hop neighbor table for each N_i is defined as

$$\mathbf{T}_{N_i} = \{[ID_{N_k}, P_{N_k}] \mid P_{N_k} \in \overline{D}(P_{N_i}, R), \forall k \neq i\} \quad (1)$$

where ID_{N_k} represents the designated identification number for N_k. In the greedy forwarding (GF) algorithm, it is assumed that the source node N_S is aware of the location of the destination node N_D. If N_S wants to transmit packets to N_D, it will choose the next hopping node from its $\mathbf{T}_{N_S}$ which (i) has the shortest Euclidean distance to N_D among all the SNs in $\mathbf{T}_{N_S}$ and (ii) is located closer to N_D compared to the distance between N_S and N_D (e.g. N_V as in Fig. 2). The same procedure will be performed by the intermediate nodes (such as N_V) until N_D is reached. However, the GF algorithm will be inclined to fail due to the

occurrences of voids even though some routing paths exist from N_S to N_D. The void problem is defined as follows.

Problem 1 (Void Problem). The greedy forwarding (GF) algorithm is exploited for packet delivery from N_S to N_D. The void problem occurs while there exists a void node (N_V) in the network such that

$$\{P_{N_k} \mid d(P_{N_k}, P_{N_D}) < d(P_{N_V}, P_{N_D}), \forall P_{N_k} \in \mathbf{T}_{N_V}\} = \phi \tag{2}$$

where $d(x, y)$ represents the Euclidean distance between x and y. $\mathbf{T}_{N_V}$ is the neighbor table of N_V.

3. Proposed Greedy Anti-void Routing (GAR) protocol

The objective of the GAR protocol is to resolve the void problem such that the packet delivery from N_S to N_D can be guaranteed. Before diving into the detail formulation of the proposed GAR algorithm, an introductory example is described in order to facilitate the understanding of the GAR protocol. As shown in Fig. 2, the data packets initiated from the source node N_S to the destination node N_D will arrive in N_V based on the GF algorithm. The void problem occurs as N_V receives the packets, which leads to the adoption of the RUT scheme as the forwarding strategy of the GAR protocol. A circle is formed by centering at s_V with its radius being equal to half of the transmission range $R/2$. The circle is hinged at N_V and starts to conduct counterclockwise rolling until an SN has been encountered by the boundary of the circle, i.e. N_A as in Fig. 2. Consequently, the data packets in N_V will be forwarded to the encountered node N_A.

Subsequently, a new equal-sized circle will be formed, which is centered at s_A and hinged at node N_A. The counter-clockwise rolling procedure will be proceeded in order to select the next hopping node, i.e. N_B in this case. Similarly, the same process will be performed by other intermediate nodes (such as N_B and N_X) until the node N_Y is reached, which is considered to have a smaller distance to N_D than that of N_V to N_D. The conventional GF scheme will be resumed at N_Y for delivering data packets to the destination node N_D. As a consequence, the resulting path by adopting the GAR protocol becomes {N_S, N_V, N_A, N_B, N_X, N_Y, N_Z, N_D}. In the following subsections, the formal description of the RUT scheme will be described in Subsection 3.1; while the detail of the GAR algorithm is explained in Subsection 3.2. The proofs of correctness of the GAR protocol are given in Subsection 3.3.

3.1 Rolling-ball UDG boundary Traversal (RUT) scheme

The RUT scheme is adopted to solve the boundary finding problem. The definition of boundary and the problem statement are described as follows.

Definition 1 (Boundary). If there exists a set $\mathbf{B} \subseteq \mathbf{N}$ such that (i) the nodes in **B** form a simple unidirectional ring and (ii) the nodes located on and inside the ring are disconnected with those outside of the ring, **B** is denoted as the boundary set and the unidirectional ring is called a boundary.

Problem 2 (Boundary Finding Problem). Given a UDG G(**P**, **E**) and the one-hop neighbor tables $\mathbf{T} = \{\mathbf{T}_{N_i} \mid \forall N_i \in \mathbf{N}\}$, how can a boundary be obtained by exploiting the distributed computing techniques?

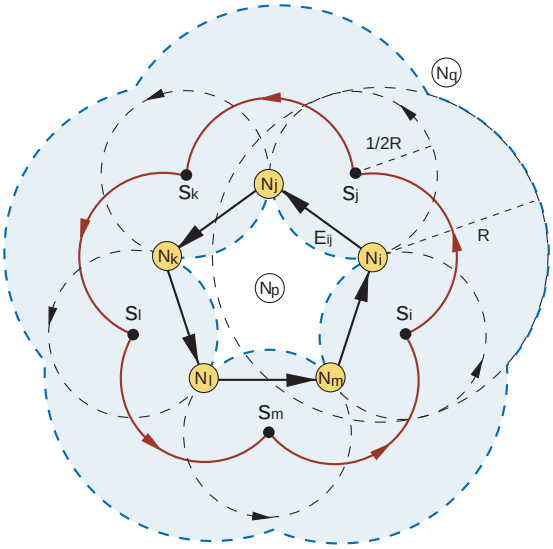


Fig. 3. The rolling-ball UDG boundary traversal (RUT) scheme: Given s_i and N_i, the RUT scheme rotates the rolling ball $RB_{N_i}(s_i, R/2)$ counter-clockwise and constructs the simple closed curve (i.e. the flower-like red solid curve). The boundary set **B** = {N_i, N_j, N_k, N_l, N_m} is established as a simple unidirectional ring by using the RUT scheme.

There are three phases within the RUT scheme, including the initialization, the boundary traversal, and the termination phases.

3.1.1 Initialization phase

No algorithm can be executed without the algorithm-specific trigger event. The trigger event within the RUT scheme is called the starting point (SP). The RUT scheme can be initialized from any SP, which is defined as follows.

Definition 2 (Rolling Ball). Given $N_i \in \mathbf{N}$, a rolling ball $RB_{N_i}(s_i, R/2)$ is defined by (i) a rolling circle hinged at P_{N_i} with its center point at $s_i \in \mathbf{R}^2$ and the radius equal to *R/2*; and (ii) there does not exist any $N_k \in \mathbf{N}$ located inside the rolling ball as $\{RB_{\tilde{N}_i}(s_i, R/2) \cap \mathbf{N}\} = \phi$, where $RB_{\tilde{N}_i}(s_i, R/2)$ denotes the open disk within the rolling ball.

Definition 3 (Starting Point). The starting point of N_i within the RUT scheme is defined as the center point $s_i \in \mathbf{R}^2$ of $RB_{N_i}(s_i, R/2)$.

As shown in Fig. 3, each node N_i can verify if there exists an SP since the rolling ball $RB_{N_i}(s_i, R/2)$ is bounded by the transmission range of N_i. According to Definition 3, the SPs should be located on the circle centered at P_{N_i} with a radius of *R/2*. As will be proven in Lemmas 1 and 2, all the SPs will result in the red solid flower-shaped arcs as in Fig. 3. It is noticed that there should always exist an SP while the void problem occurs within the network, which will be explained in Subsection 3.2. At this initial phase, the location s_i can be selected as the SP for the RUT scheme.

3.1.2 Boundary traversal phase

Given s_i as the SP associated with its $RB_{N_i}(s_i, R/2)$ hinged at N_i, either the counter-clockwise or clockwise rolling direction can be utilized. As shown in Fig. 3, $RB_{N_i}(s_i, R/2)$ is rolled counter-clockwise until the next SN is reached (i.e. N_j in Fig. 3). The unidirectional edge E_{ij} = (P_{N_i}, P_{N_j}) can therefore be constructed. A new SP and the corresponding rolling ball hinged at N_j, i.e. s_j and $RB_{N_j}(s_j, R/2)$, will be assigned and consequently the same procedure can be conducted continuously.

3.1.3 Termination phase

The termination condition for the RUT scheme happens while the first unidirectional edge is revisited. As shown in Fig. 3, the RUT scheme will be terminated if the edge E_{ij} is visited again after the edges E_{ij} , E_{jk}, E_{kl}, E_{lm}, and E_{mi} are traversed. The boundary set initiated from N_i can therefore be obtained as **B** = {N_i, N_j, N_k, N_l, N_m}.

3.2 Detail description of proposed GAR protocol

As shown in Fig. 2, the packets are intended to be delivered from N_S to N_D. N_S will select N_V as the next hopping node by adopting the GF algorithm. However, the void problem prohibits N_V to continue utilizing the same GF algorithm for packet forwarding. The RUT scheme is therefore employed by assigning an SP (i.e. s_V) associated with the rolling ball $RB_{N_V}(s_V, R/2)$ hinged at N_V. As illustrated in Fig. 2, s_V can be chosen to locate on the connecting line between N_V and N_D with *R/2* away from N_V. It is noticed that there always exists an SP for the void node (N_V) since there is not supposed to have any SN located within the blue-shaded region (as in Fig. 2), which is large enough to satisfy the requirements as in Definitions 2 and 3. The RUT scheme is utilized until N_Y is reached (after traversing N_A, N_B, and N_X). Since $d(P_{N_Y}, P_{N_D}) < d(P_{N_V}, P_{N_D})$, the GF algorithm is resumed at N_Y and the next hopping node will be selected as N_Z. The route from N_S to N_D can therefore be constructed for packet delivery. Moreover, if there does not exist a node N_Y such that $d(P_{N_Y}, P_{N_D}) < d(P_{N_V}, P_{N_D})$ within the boundary traversal phase, the RUT scheme will be terminated after revisiting the edge E_{VA}. The result indicates that there does not exist a routing path between N_S and N_D.

3.3 Proof of correctness

In this subsection, the correctness of the RUT scheme is proven in order to solve Problem 2; while the GAR protocol is also proven for resolving the void problem (i.e. Problem 1) in order to guarantee packet delivery.

Fact 1. A simple closed curve is formed by traversing a point on the border of a closed filled two-dimensional geometry.

Lemma 1. All the SPs within the RUT scheme form the border of the resulting shape by overlapping the closed disks $\overline{D}(P_{N_i}, R/2)$ for all $N_i \in \mathbf{N}$, and vice versa.

Proof: Based on Definitions 2 and 3, the set of SPs can be obtained as **S** = $\mathbf{R_1} \cap \mathbf{R_2}$ = $\{s_i \mid \|s_i - P_{N_i}\| = R/2, \exists N_i \in \mathbf{N}, s_i \in \mathbf{R}^2\} \cap \{s_j \mid \|s_j - P_{N_j}\| \geq R/2, \forall N_j \in \mathbf{N}, s_j \in \mathbf{R}^2\}$ by adopting the (i) and (ii) rules within Definition 2. On the other hand, the border of the resulting shape from the overlapped closed disks $\overline{D}(P_{N_i}, R/2)$ for all $N_i \in \mathbf{N}$ can be denoted as

$\mathbf{\Omega} = \mathbf{Q}_1 - \mathbf{Q}_2 = \bigcup_{N_i \in \mathbf{N}} C(P_{N_i}, R/2) - \bigcup_{N_i \in \mathbf{N}} D(P_{N_i}, R/2)$, where $C(P_{N_i}, R/2)$ and $D(P_{N_i}, R/2)$ represent the circle and the open disk centered at P_{N_i} with a radius of *R/2* respectively. It is obvious to notice that $\mathbf{R_1} = \mathbf{Q_1}$ and $\mathbf{R_2} = \mathbf{Q'_2}$, which result in $\mathbf{S} = \mathbf{\Omega}$. It completes the proof.

Lemma 2. A simple closed curve is formed by the trajectory of the SPs.

Proof: Based on Lemma 1, the trajectory of the SPs forms the border of the overlapped closed disks $\overline{D}(P_{N_i}, R/2)$ for all $N_i \in \mathbf{N}$. Moreover, the border of a closed filled two-dimensional geometry is a simple closed curve according to Fact 1. Therefore, a simple closed curve is constructed by the trajectory of the SPs, e.g. the solid flower-shaped closed curve as in Fig. 3. It completes the proof.

Theorem 1. The boundary finding problem (Problem 2) is resolved by the RUT scheme.

Proof: Based on Lemma 2, the RUT scheme can draw a simple closed curve by rotating the rolling balls $RB_{N_i}(s_i, R/2)$ hinged at P_{N_i} for all $N_i \in \mathbf{N}$. The closed curve can be divided into arc segments $S(s_i, s_j)$, where s_i is the starting SP associated with N_i; and s_j is the anchor point while rotating the $RB_{N_i}(s_i, R/2)$ hinged at P_{N_j}. The arc segments $S(s_i, s_j)$ can be mapped into the unidirectional edges $E_{ij} = (P_{N_i}, P_{N_j})$ for all $N_i, N_j \in \mathbf{U}$, where $\mathbf{U} \subseteq \mathbf{N}$. Due to the one-to-one mapping between $S(s_i, s_j)$ and E_{ij}, a simple unidirectional ring is constructed by E_{ij} for all $N_i, N_j \in \mathbf{U}$. According to the RUT scheme, there does not exist any $N_i \in \mathbf{N}$ within the area traversed by the rolling balls, i.e. inside the light blue region as in Fig. 3. For all $N_p \in \mathbf{N}$ located inside the simple unidirectional ring, the smallest distance from N_p to N_q, which is located outside of the ring, is greater than the SN's transmission range *R*. Therefore, there does not exist any $N_p \in \mathbf{N}$ inside the simple unidirectional ring that can communicate with $N_q \in \mathbf{N}$ located outside of the ring. Based on Definition 1, the set **U** is identical to the boundary set, i.e. **U** = **B**. It completes the proof.

Theorem 2. The void problem (Problem 1) is solved by the GAR protocol with guaranteed packet delivery.

Proof: With the existence of the void problem occurring at the void node N_V, the RUT scheme is utilized by initiating an SP (s_V) with the rolling ball $RB_{N_V}(s_V, R/2)$ hinged at N_V. The RUT scheme within the GAR protocol will conduct boundary (i.e. the set **B**) traversal under the condition that $d(P_{N_i}, P_{N_D}) \geq d(P_{N_V}, P_{N_D})$ for all $N_i \in \mathbf{B}$. If the boundary within the underlying network is completely travelled based on Theorem 1, it indicates that the SNs inside the boundary (e.g. N_V) are not capable of communicating with those located outside of the boundary (e.g. N_D). The result shows that there does not exist a route from the void node (N_V) to the destination node (N_D), i.e. the existence of network partition. On the other hand, if there exists a node N_Y such that $d(P_{N_Y}, P_{N_D}) < d(P_{N_V}, P_{N_D})$ (as shown in Fig. 2), the GF algorithm will be adopted within the GAR protocol to conduct data delivery toward the destination node N_D. Therefore, the GAR protocol solves the void problem with guaranteed packet delivery, which completes the proof.

4. Enhanced mechanisms for proposed GAR protocol

In order to enhance the routing efficiency of the proposed GAR protocol, two mechanisms are proposed in this section, i.e. the hop count reduction (HCR) and the intersection navigation (IN) schemes. These two mechanisms are described as follows.

4.1 Hop Count Reduction (HCR) mechanism

Based on the rolling ball traversal within the RUT scheme, the selected next-hop nodes may not be optimal by considering the minimal hop count criterion. Excessive routing delay associated with power consumption can occur if additional hopping nodes are traversed by adopting the RUT scheme. As shown in Fig. 4, the void node N_V starts the RUT scheme by selecting N_1 as its next hop node with the counter-clockwise rolling direction; while N_2 and N_3 are continuously chosen as the next hopping nodes. Considering the case that N_3 is located within the same transmission range of N_1, it is apparently to observe that the packets can directly be transmitted from N_1 to N_3. Excessive communication waste can be preserved without conducting the rerouting process to N_2. Moreover, the boundary set **B** forms a simple unidirectional ring based on Theorem 1, which indicates that the next-hop SN of a node can be uniquely determined if its previous hopping SN is already specified. For instance (as in Fig. 4), if N_V is the previous node of N_1, N_1's next hopping node N_2 is uniquely determined, i.e. the transmission sequences of every three nodes (e.g. $\{N_V \rightarrow N_1 \rightarrow N_2\}$ or $\{N_1 \rightarrow N_2 \rightarrow N_3\}$) can be uniquely defined.

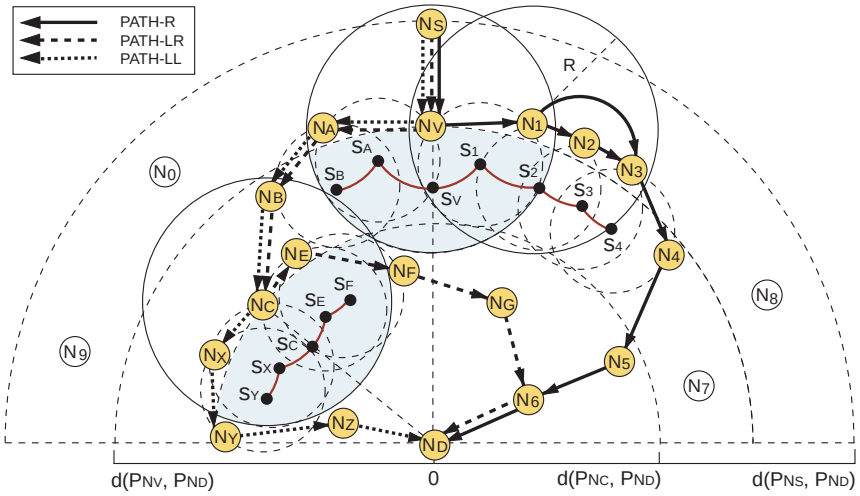


Fig. 4. The hop count reduction (HCR) and the intersection navigation (IN) mechanisms: (N_S, N_D) is the transmission pair, and N_V and N_C are the void nodes. The HCR mechanism: The SN N_1 can create a short cut to its neighbor N_3 by listening to the packet forwarding since the path $\{N_1 \rightarrow N_2 \rightarrow N_3\}$ is uniquely determined. The IN mechanism: Counter-clockwise and clockwise rolling directions (denoted as the symbols of R and L) can be adopted in the RUT scheme. By flooding the navigation map (NAV_MAP) control packets, the shortest path can be acquired as PATH-R $\{N_S, N_V, N_1, N_3, N_4, N_5, N_6, N_D\}$ with 7 hops. Consequently, all packets will contain the sequence {R} to choose the counter-clockwise rolling direction at the first void node N_V. In the case that the path were selected as PATH-LR, the sequence would change to {LR} by choosing the clockwise rolling direction at the first void node N_V and counter-clockwise at the second void node N_C.

According to the concept as stated above, the hop count reduction (HCR) mechanism is to acquire the information of the next few hops of neighbors under the RUT scheme by

listening to the same forwarded packet. It is also worthwhile to notice that the listening process does not incur additional transmission of control packets. As shown in Fig. 4, N_1 chooses N_2 as its next-hop node for packet forwarding; while N_2 selects N_3 as the next hopping node in the same manner. Under the broadcast nature, N_1 will listen to the same packets in the forwarding process from N_2 to N_3. By adopting the HCR mechanism, N_1 will therefore select N_3 as its next hopping node instead of choosing N_2 while adopting the original RUT scheme. Consequently, N_1 will initiate its packet forwarding process to N_3 directly by informing the RUT scheme that the rerouting via N_2 can be skipped.

4.2 Intersection Navigation (IN) mechanism

The intersection navigation (IN) mechanism is utilized to determine the rolling direction in the RUT scheme while the void problem occurs. It is noticed that the selection of rolling direction (i.e. either counter-clockwise or clockwise) does not influence the correctness of the proposed RUT scheme to solve Problem 2 as in Theorem 1. However, the routing efficiency may be severely degraded if a comparably longer routing path is selected at the occurrence of a void node. The primary benefit of the IN scheme is to choose a feasible rolling direction while a void node is encountered. Consequently, smaller rerouting hop counts (HC) and packet transmission delay can be achieved.

Based on the transmission pair (N_S, N_D) as shown in Fig. 4, N_V and N_C become the void nodes within the network topology. There exist three potential paths from N_S to N_D by adopting the RUT scheme, i.e. PATH-R, PATH-LR, and PATH-LL. The suffixes R, LR, and LL represent the sequences of the adopted rolling direction at each encountered void node, where the symbol R is denoted as counter-clockwise rolling direction and L represents clockwise direction. It is noted that the suffix with two symbols indicates that two void nodes are encountered within the path. The entire node traversal for each path is as follows: PATH-R = {N_S, N_V, N_1, N_3, N_4, N_5, N_6, N_D}, PATH-LR = {N_S, N_V, N_A, N_B, N_C, N_E, N_F, N_G, N_6, N_D}, PATH-LL = {N_S, N_V, N_A, N_B, N_C, N_X, N_Y, N_Z, N_D}. Different HCs are observed with each path as HC(PATH-R) = 7, HC(PATH-LR) = 9, and HC(PATH-LL) = 8.

The main objective of the IN scheme is to monitor the number of HC such that the path with the shortest HC can be selected, i.e. PATH-R in this case. A navigation map control packet (NAV_MAP) defined in the IN scheme is utilized to indicate the rolling direction while the void node is encountered. For example, two NAV_MAP packets are initiated after N_V is encountered, where NAV_MAP = {R} is delivered via the counter-clockwise direction to N_D and NAV_MAP = {L} is carried with the clockwise direction. It is noticed that the HC associated with each navigation path is also recorded within the NAV_MAP packets. As the second void node N_C is observed, the control message NAV_MAP = {L} is transformed into two different navigation packets (i.e. NAV_MAP = {LR} and NAV_MAP = {LL}), which traverse the two different rolling directions toward N_D. As a result, the destination node N_D will receive several NAV_MAP packets at different time instants associated with the on-going transmission of the data packets. The NAV_MAP packet with the shortest HC value (i.e. NAV_MAP = {R} in this case) will be selected as the targeting path. Therefore, the control packet with NAV_MAP = {R} will be traversed from N_D back to the N_S in order to notify the source node N_S with the shortest path for packet transmission. After acquiring the NAV_MAP information, N_S will conduct its remaining packet delivery based on the corresponding rolling direction. Considerable routing efficiency can be preserved as a shorter routing path is selected by adopting the IN mechanism.

5. Performance evaluation

The performance of the proposed GAR algorithm is evaluated and compared with the existing localized schemes (i.e. the GF and the BOUNDHOLE algorithms) via simulations. Furthermore, the GAR protocol with the enhanced mechanisms (i.e. the HCR and the IN schemes) is also implemented, which is denoted as the GAR-E algorithm. The simulations are conducted in the NS-2 network simulator (Heidemann at al., 2001) with wireless extension, using the IEEE 802.11 DCF as the MAC protocol. The parameters utilized in the simulations are listed as shown in Table 1.

Parameter Type	**Parameter Value**
Grid Area	1000 x 800 m²
Void Block	500 x 800 m²
Simulation Time	150 sec
Transmission Range	250 m
Traffic Type	Constant Bit Rate (CBR)
Data Rate	12 Kbps
Size of Data Packets	512 Bytes
Number of Nodes	41, 51, 61, 71, 81

Table 1. Simulation Parameters

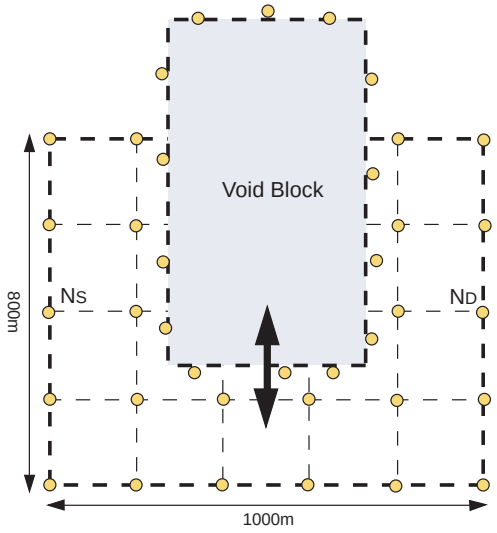


Fig. 5. The simulation scenario: The transmission pair (N_S, N_D) is located at the center of the left and right boundaries of the grid topology. Moreover, there exists a void block with SNs located around the peripheral of the block; while none of the SNs is situated inside the block. The void block is randomly moved in the vertical direction in order to simulate the existence of a void problem within the network.

The simulation scenario is described as follows. As shown in Fig. 5, the grid topology with the existence of a void block is considered in the simulation. It is noted that there are SNs located around the peripheral of the void block; while none of the SNs is situated inside the block. The source and destination nodes N_S and N_D are located at the center of the left and right boundaries as shown in Fig. 5. The data packets are transmitted from N_S to N_D with the void block that is randomly moved with vertical direction in order to simulate the existence of a void problem within the network. It is noted that network partition between N_S and N_D is not considered to exist in the simulation. One hundred simulation runs are conducted for each randomly moved void block case. The following five metrics are utilized in the simulations for performance comparison:

1. **Packet Arrival Rate**: The ratio of the number of received data packets to the number of total data packets sent by the source.
2. **Average End-to-End Delay**: The average time elapsed for delivering a data packet within a successful transmission.
3. **Path Efficiency:** The ratio of the number of total hop counts within the entire routing path over the number of hop counts for the shortest path.
4. **Communication Overhead:** The average number of transmitted control bytes per second, including both the data packet header and the control packets.
5. **Energy Consumption:** The energy consumption for the entire network, including transmission energy consumption for both the data and control packets under the bit rate of 11 Mbps and the transmitting power of 15 dBm for each SN.

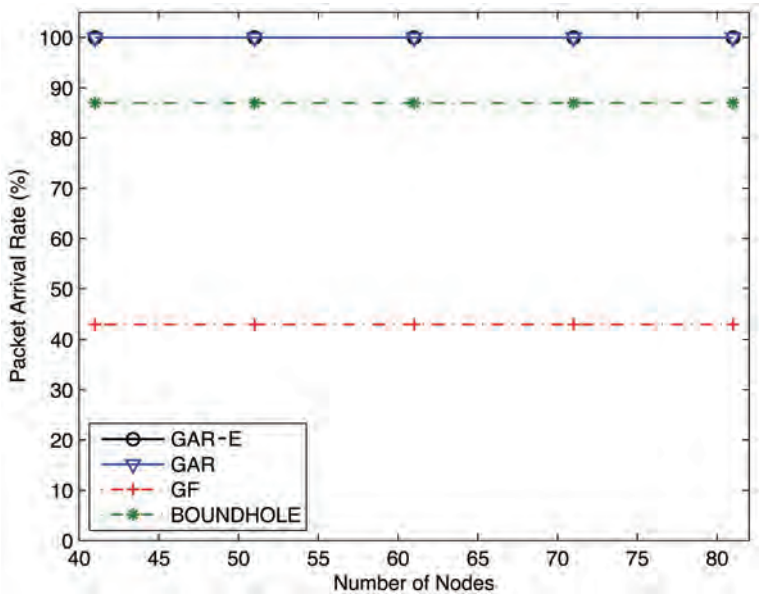


Fig. 6. Packet Arrival Rate (%) vs. Number of Nodes

Figs. 6 to 10 show the performance comparison between these four algorithms under different number of nodes within the UDG network. As can be seen from Fig. 6, the packet arrival rates obtained from these four algorithms are independent to the number of nodes

within the network, which is attributed to the design nature of these four schemes. In both of the proposed GAR and GAR-E protocols, 100% of packet arrival rate is guaranteed under different number of nodes. These results are consistent with the protocol design that is proven to ensure 100% of packet arrival rate as long as the network is not partitioned between N_S and N_D. It can also be observed in Fig. 6 that the BOUNDHOLE algorithm can achieve around 88% of packet arrival rate due to the occurrence of routing loop; while the GF scheme can only attain around 45% since the void problem is not considered within its protocol design.

Fig. 7 shows the average end-to-end delay for successful packet delivery by adopting these four algorithms. The conventional GF protocol possesses the smallest end-to-end delay due to its negligence of the void problem, which leads to less than 50% of packet arrival rate as shown in Fig. 6. On the other hand, the BOUNDHOLE algorithm results in the largest end-to-end delay owing to its potential rerouting and looping under certain cases, e.g. as in Fig. 2. The proposed GAR and GAR-E protocols can achieve comparably less routing delay comparing with the BOUNDHOLE scheme, i.e. around 15 to 25 ms less in end-to-end delay. Moreover, the GAR-E algorithm can provide additional 8 to 15 ms less delay comparing with the original GAR protocol due to the enhanced HCR and IN mechanisms. It is also noteworthy to observe the *M*-shape curves resulted within these four schemes. The primary reason can be attributed to the different hop counts between the source/destination pair generated by the GF algorithm. It is noted that the GAR, GAR-E, and BOUNDHOLE schemes implement the GF algorithm without the occurrence of the void problem. The hop counts under the cases of the five different numbers of SNs are computed as 5, 7, 6, 6, and 5. It can be apparently translated into the *M*-shape curves of the end-to-end delay performance as shown in Fig. 7.

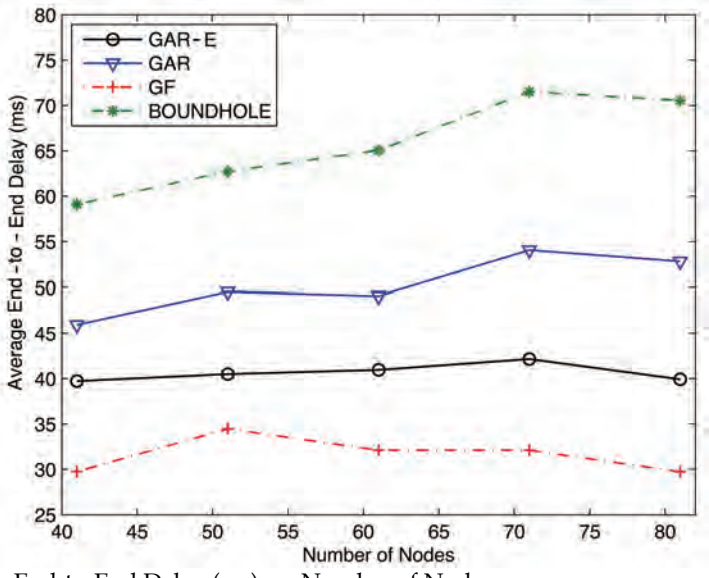


Fig. 7. Average End-to-End Delay (ms) vs. Number of Nodes

As shown in Fig. 8, the path efficiency acquired from these four schemes follows the similar trend as that from the average end-to-end delay. Due to the greedy nature and the negligence of the void problem, the path efficiency of the conventional GF scheme can achieve almost one in the simulations, i.e. the total number of hop counts is almost equal to that of the shortest path. The proposed GAR algorithm possesses the path efficiency of around 1.3 to 1.5. Furthermore, the GAR-E protocol further enhances the path efficiency to around the value of 1.1, which greatly outperforms the BOUNDHOLE schemes.

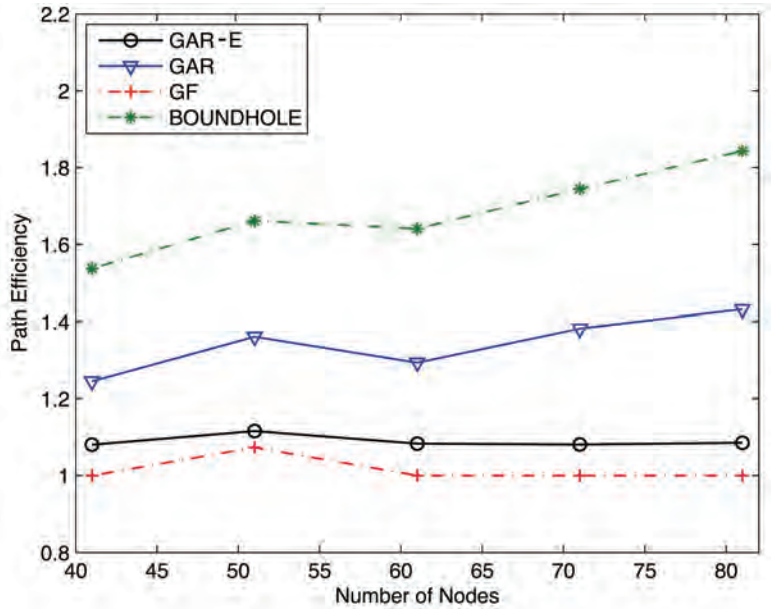


Fig. 8. Path Efficiency vs. Number of Nodes

Fig. 9 shows the communication overheads resulting from these four schemes, which are observed to increase as the increment of the number of nodes. The reason is attributed to the excessive control packets that are required for obtaining the neighbor's locations while the number of nodes is augmented. It is noted that the GF algorithm possesses the smallest communication overheads owing to its ignorance of the void problem. The BOUNDHOLE algorithm results in the largest communication overhead among all the schemes due to its usage of excessive header bytes for preventing the routing loops. It is noticed that even though the GAR-E scheme requires additional NAV_MAP control packets for achieving the IN mechanism, the total required communication overhead is smaller than that from the GAR protocol due to its comparably smaller rerouting number of hop counts.

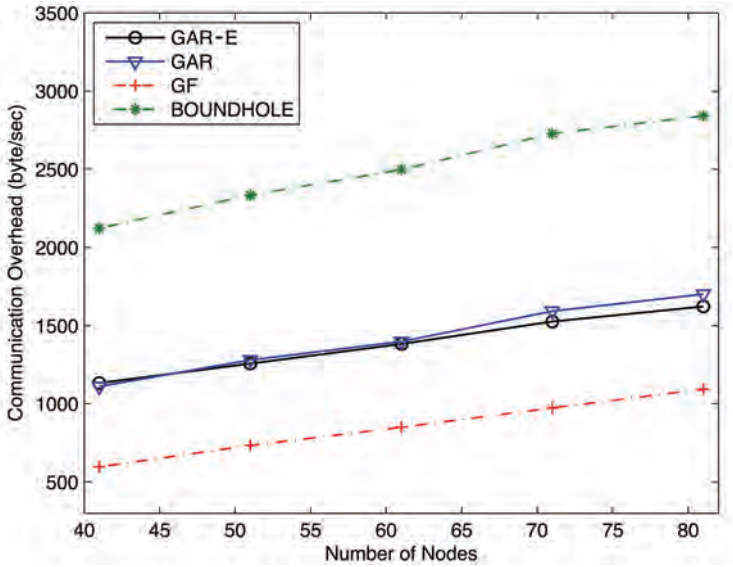


Fig. 9. Communication Overhead (byte/sec) vs. Number of Nodes

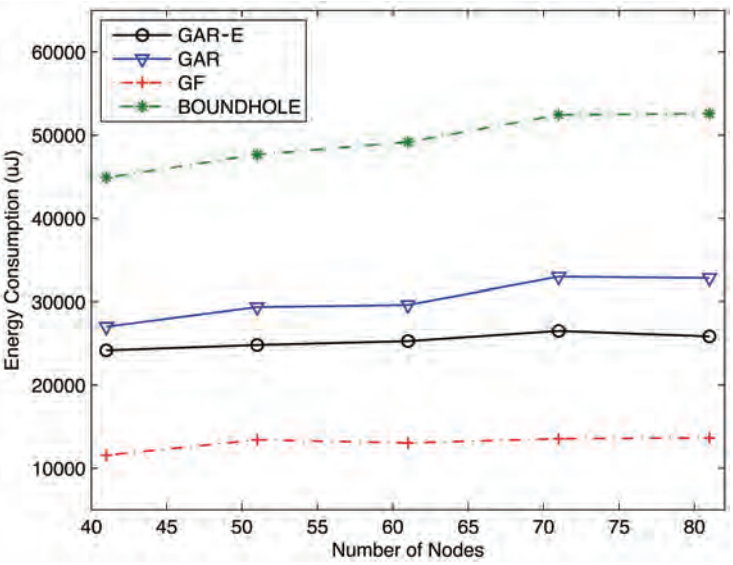


Fig. 10. Energy Consumption (uJ) vs. Number of Nodes

The comparison for energy consumption between these algorithms is presented in Fig. 10. Similar performance trend can be observed between the energy consumption and the communication overhead as shown in Fig. 9. Except for the reference GF protocol, the proposed GAR and GAR-E algorithms can effectively reduce the energy consumption in comparison with the baseline BOUNDHOLE scheme. The merits of the proposed GAR and GAR-E algorithms are observed and validated via the simulation results.

6. Conclusion

In this book chapter, a greedy anti-void routing (GAR) protocol is proposed to completely resolve the void problem incurred by the conventional greedy forwarding algorithm. The rolling-ball UDG boundary traversal (RUT) scheme is adopted within the GAR protocol to solve the boundary finding problem, which results in guaranteed delivery of data packets. The correctness of the RUT scheme and the GAR algorithm are properly proven. The GAR protocol with two delay-reducing schemes, the hop count reduction (HCR) and the intersection navigation (IN) mechanisms, is proposed as the enhanced GAR (GAR-E) algorithm that inherits the merit of guaranteed delivery. The performance of both the GAR and GAR-E protocols is evaluated via simulations and is compared with existing localized routing algorithms. The simulation study shows that the proposed GAR and GAR-E algorithms can guarantee the delivery of data packets; while the GAR-E scheme further improves the routing efficiency and the communication overhead. Feasible routing performance can therefore be achieved.

7. Acknowledgments

This work was in part funded by the MOE ATU Program 95W803C, NSC 96-2221-E-009-016, MOEA 96-EC-17-A-01-S1-048, the MediaTek research center at National Chiao Tung University, and the Universal Scientific Industrial (USI) Co., Taiwan.

8. References

Arad, N. & Shavitt, Y. (2006). Minimizing Recovery State in Geographic Ad-Hoc Routing, *Proceedings of ACM Int. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc'06)*, pp. 13-24, May 2006

Bose, P.; Morin, P.; Stojmenovi´c, I. & Urrutia, J. (2001). Routing with Guaranteed Delivery in Ad Hoc Wireless Networks, *ACM/Kluwer Wireless Networks*, vol. 7, no. 6, Nov. 2001, pp. 609-616

Chen, S.; FAN, G. & Cui, J.H. (2006). Avoid "Void" in Geographic Routing for Data Aggregation in Sensor Networks, *Int. Journal of Ad Hoc and Ubiquitous Computing (IJAHUC)*, vol. 1, no. 4, 2006, pp. 169-178

Estrin, D.; Govindan, R.; Heidemann, J. & Kumar, S. (1999). Next Century Challenges: Scalable Coordination in Sensor Networks, *Proceedings of ACM/IEEE Int. Conf. Mobile Computing and Networking (MobiCom'99)*, pp. 263-270, Aug. 1999

Fang, Q.; Gao, J. & Guibas, L. (2004). Locating and Bypassing Routing Holes in Sensor Networks, *Proceedings of IEEE Int. Conf. Computer Communications (INFOCOM'04)*, pp. 2458-2468, Mar. 2004

Finn, G.G. (1987). Routing and Addressing Problems in Large Metropolitan-Scale Internetworks, *Info. Sci. Inst. (ISI), Tech. Rep. ISI/RR-87-180*, Mar. 1987

Frey, H. & Stojmenović, I. (2006). On Delivery Guarantees of Face and Combined Greedy Face Routing in Ad Hoc and Sensor Networks, *Proceedings of ACM/IEEE Int. Conf. Mobile Computing and Networking (MobiCom'06)*, pp. 390-401, Sept. 2006

Giruka, V.C. & Singhal, M. (2005). Angular Routing Protocol for Mobile Ad Hoc Networks, *Proceedings of IEEE Int. Conf. Distributed Computing Systems Workshops (ICDCSW'05)*, pp. 551-557, Jun. 2005

He, T.; Stanković, J.A.; Lu, C. & Abdelzaher, T. (2003). SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks, *Proceedings of Int. Conf. Distributed Computing Systems (ICDCS'03)*, pp. 46-55, May 2003

Heidemann, J.; Bulusu, N.; Elson, J.; Intanagonwiwak, C.; Lan, K.; Xu, Y.; Ye, W.; Estrin, D. & Govindan, R. (2001). Effects of Detail in Wireless Network Simulation, *Proceedings of SCS Multiconference on Distributed Simulation*, pp. 3-11, Jan. 2001

Karp, B. & Kung, H.T. (2000). GPSR: Greedy Perimeter Stateless Routing for Wireless Networks, *Proceedings of ACM/IEEE Int. Conf. Mobile Computing and Networking (MobiCom'00),* pp. 243-254, Aug. 2000

Kranakis, E.; Singh, H. & Urrutia, J. (1999). Compass Routing on Geometric Networks, *Proceedings of Canadian Conf. Computational Geometry (CCCG'99)*, pp. 51-54, Aug. 1999

Kuhn, F.; Wattenhofer, R.; Zhang, Y. & Zollinger, A. (2003). Geometric Ad-hoc Routing: Of Theory and Practice, *Proceedings of ACM Symp. Principles of Distributed Computing (PODC)*, pp. 63-72, Jul. 2003

Kuhn, F.; Wattenhofer, R. & Zollinger, A. (2003). Worst-case Optimal and Average-case Efficient Geometric Ad-hoc Routing, *Proceedings of ACM Int. Symp. Mobile Computing and Networking (MobiHoc'03)*, pp. 267-278, Jun. 2003

Leong, B.; Mitra, S. & Liskov, B. (2005). Path Vector Face Routing: Geographic Routing with Local Face Information, *Proceedings of IEEE Int. Conf. Network Protocols (ICNP'05)*, pp. 147-158, Nov. 2005

Liu, W.J. & Feng, K.T. (2006). Largest Forwarding Region Routing Protocol for Mobile Ad Hoc Networks, *Proceedings of IEEE Global Communications Conference (GLOBECOM'06)*, pp. 1-5, Nov. 2006

Na, J.; Soroker, D. & Kim, C.K. (2007). Greedy Geographic Routing Using Dynamic Potential Field for Wireless Ad Hoc Networks, *IEEE Commun. Lett.*, vol. 11, no. 3, Mar. 2007, pp. 243-245

Stojmenović, I. & Lin, X. (2001). Loop-Free Hybrid Single-Path/Flooding Routing Algorithms with Guaranteed Delivery for Wireless Networks, *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, Oct. 2001, pp. 1023-1032

Stojmenović, I.; Russell, M. & Vukojevic, B. (2000). Depth First Search and Location Based Localized Routing and QoS Routing in Wireless Networks, *Proceedings of IEEE Int. Conf. Parallel Processing (ICPP'00)*, pp. 173-180, Aug. 2000

West, D.B. (2000). *Introduction to Graph Theory,* 2nd ed., Prentice Hall

Zou, L.; Lu, M. & Xiong, Z. (2005). A Distributed Algorithm for the Dead End Problem of Location Based Routing in Sensor Networks, *IEEE Trans. Veh. Technol.*, vol. 54, no. 4, Jul. 2005, pp. 1509-1522

16

Greedy Like Algorithms for the Traveling Salesman and Multidimensional Assignment Problems

Gregory Gutin and Daniel Karapetyan
Royal Holloway, University of London
United Kingdom

1. Introduction

Majority of chapters of this book show usefulness of greedy like algorithms for solving various combinatorial optimization problems. The aim of this chapter is to warn the reader that not always a greedy like approach is a good option and, in certain cases, it is a very bad option being sometimes among the worst possible options. Our message is not a discouragement from using greedy like algorithms altogether; we think that for every combinatorial optimization problem of importance, researchers and practitioners should simply investigate the appropriateness of greedy like algorithms and the existence of better alternatives to them (considering both quality of solution and running time). In many cases, especially when the running time must be very short, the conclusion may still be that the most practical of known approaches is a greedy like algorithm.

The Traveling Salesman and Multidimensional Assignment Problems are optimization problems for which greedy like approaches are usually not very successful. We demonstrate this by providing both theoretical and experimental results on greedy like algorithms as well as on some other algorithms that produce (in theory and/or in experiments) much better results without spending significantly more time.

There are some general theoretical results that indicate that there are, in fact, many combinatorial optimization problems for which greedy like algorithms are not the best option even among fast construction heuristics, see, e.g., [3, 5, 17]. We will not consider these general results in order to avoid most mathematical details that are not necessary for understanding the results of this chapter. For this reason we will not give proofs here apart from two simple proofs: that of Theorem 8 which shows that some instances on which the greedy algorithm fails are not exotic in a sense and that of Theorem 11 since Theorem 11 is a new result.

It is not a trivial question whether a certain algorithm is greedy like or not. In the next section we define an independence system and give the classic definition of the greedy algorithm for such a system. We extend this definition to so-called greedy type algorithms that include such well-known algorithms as the Prim's algorithm for the minimum spanning tree problem and the nearest neighbor algorithm for the traveling salesman problem. We use the term 'greedy like' in an informal way and we include in this class simple and fast construction heuristics that seem to us to be of greedy nature.

Unfortunately, no formal definition exists for the wide family of greedy like algorithms and one can understand the difficulty to formally classify such algorithms by, for example, considering local search algorithms which find the best solution in each neighborhood they search. Intuitively, it is clear that such local search algorithms are not greedy yet their every search is greedy in a sense.
In the next section, we give most of terminology and notation used in this chapter. Several results on theoretical performance of greedy like algorithms for the Traveling Salesman and Multidimensional Assignment Problems are discussed in Sections 3 and 4, respectively. Experimental results on greedy like algorithms for the Traveling Salesman and Multidimensional Assignment Problems are given and analyzed in Sections 5 and 6, respectively.

2. Terminology and notation

The *Asymmetric Traveling Salesman Problem (ATSP)* is the problem of computing a minimum weight tour (Hamilton directed cycle) passing through every vertex in a weighted complete digraph K_n^* on n vertices. The *Symmetric TSP (STSP)* is the same problem, but on a weighted complete undirected graph K_n. When a certain fact holds for both ATSP and STSP, we will simply speak of *TSP*. We often assume that the vertices of K_n^* and K_n are 1, 2, …, n and often refer to the weight $w(ij)$ of an edge ij of K_n^* (or K_n) as the *distance* from i to j. TSP has a large number of applications, see, e.g., the two recent books [1, 13] on TSP.
The Multidimensional Assignment Problem (MAP) (abbreviated s-AP in the case of s dimensions) is a well-known optimization problem with a host of applications (see, e.g., [2, 6, 7] for 'classic' applications and [4, 25] for recent applications in solving systems of polynomial equations and centralized multisensor multitarget tracking). In fact, several applications described in [4, 6, 25] naturally require the use of s-AP for values of s larger than 3.
For a fixed $s \geq 2$, the s-AP is stated as follows. Let $X_1 = X_2 = \ldots = X_s = \{1, 2, \ldots, n\}$. We will consider only vectors that belong to the Cartesian product $X = X_1 \times X_2 \times \ldots \times X_s$. Each vector $e \in X$ is assigned a non-negative integral weight $w(e)$. For a vector $e \in X$, the component e_j denotes its jth coordinate, i.e., $e_j \in X_j$. A collection A of $t \leq n$ vectors $e^1, e^2, \ldots, e^t$ is a *(feasible) partial assignment* if $e_j^i \neq e_j^k$ holds for each $i \neq k$ and $j \in \{1, 2, \ldots, s\}$. The *weight* of a partial assignment A is $w(A) = \sum_{i=1}^{t} w(e_i)$. An *assignment* (or *full assignment*) is a partial assignment with n vectors. The objective of s-AP is to find an assignment of minimum weight.
Let $\mathcal{P}$ be a combinatorial optimization problem and let $\mathcal{H}$ be a heuristic for $\mathcal{P}$. The *domination number* $\mathrm{domn}(\mathcal{H}, \mathcal{I})$ of $\mathcal{H}$ *for an instance* $\mathcal{I}$ of $\mathcal{P}$ is the number of solutions of $\mathcal{I}$ that are not better than the solution s produced by $\mathcal{H}$ including s itself. For example, consider an instance $\mathcal{T}$ of the STSP on 5 vertices. Suppose that the weights of tours in $\mathcal{T}$ are 2, 5, 5, 6, 6, 9, 9, 11, 11, 12, 12, 15 (every instance of STSP on 5 vertices has 12 tours) and suppose that the greedy algorithm computes the tour $\mathcal{T}$ of weight 6. Then $\mathrm{domn}(\texttt{greedy}, \mathcal{T}) = 9$. In general, if $\mathrm{domn}(\mathcal{H}, \mathcal{I})$ equals the number of solutions in $\mathcal{I}$, then $\mathcal{H}$ finds an optimal solution for $\mathcal{I}$. If $\mathrm{domn}(\mathcal{H}, \mathcal{I}) = 1$, then the solution found by $\mathcal{H}$ for $\mathcal{I}$ is the unique worst possible one. The *domination number* $\mathrm{domn}(\mathcal{H}, n)$ of $\mathcal{H}$ is the minimum of $\mathrm{domn}(\mathcal{H}, \mathcal{I})$ over all instances $\mathcal{I}$ of size n.
An *independence system* is a pair consisting of a finite set E and a family $\mathcal{F}$ of subsets (called *independent sets*) of E such that (I1) and (I2) are satisfied.

(I1) the empty set is in $\mathcal{F}$;

(I2) If $X \in \mathcal{F}$ and Y is a subset of X, then $Y \in \mathcal{F}$.

All maximal sets of $\mathcal{F}$ are called *bases* (or, *feasible solutions*).

Many combinatorial optimization problems can be formulated as follows. We are given an independence system $(E,\mathcal{F})$, a set $W \subseteq \mathbb{Z}_+$ and a weight function w that assigns a weight $w(e) \in W$ to every element of E ($\mathbb{Z}_+$ is the set of non-negative integers). The weight $w(S)$ of $S \in \mathcal{F}$ is defined as the sum of the weights of the elements of S. It is required to find a base $B \in \mathcal{F}$ of minimum weight. We will consider only such problems and call them the $(E,\mathcal{F},W)$-*optimization problems*.

ATSP is an $(E,\mathcal{F},\mathbb{Z}_+)$-optimization problem, where E is the set of arcs of the complete digraph K_n^* and $\mathcal{F} = \{B \subseteq H : H \in \mathcal{H}\}$, where $\mathcal{H}$ is the set of Hamilton directed cycles of K_n^*. MAP is also an $(E,\mathcal{F},\mathbb{Z}_+)$-optimization problem, where E is the set of all vectors and $\mathcal{F}$ is the set of all partial assignments.

If $S \in \mathcal{F}$, then let $I(S) = \{x : S \cup \{x\} \in \mathcal{F}\} \setminus S$. This means that $I(S)$ consists of those elements from $E \setminus S$, which can be added to S, in order to have an independent set of size $|S| + 1$. Note that by (I2) $I(S) \neq 0$ for every independent set S which is not a base.

The *Greedy Algorithm* (`Greedy`) tries to construct a minimum weight base as follows: it starts from an empty set X, and at every step it takes the current set X and adds to it a minimum weight element $e \in I(X)$, the algorithm stops when a base is built. We assume that the greedy algorithm may choose any element among equally weighted elements in $I(X)$. Thus, when we say that the greedy algorithm *may construct* a base B, we mean that B is built provided the appropriate choices between elements of the same weight are made.

Greedy type algorithms were introduced in [14]. They include the nearest neighbor algorithm for TSP and are defined as follows. A *greedy type* algorithm $\mathcal{H}$ is similar to the greedy algorithm: start with the partial solution $X = 0$; and then repeatedly add to X an element of minimum weight in $I_{\mathcal{H}}(X)$ (ties are broken arbitrarily) until X is a base of $\mathcal{F}$, where $I_{\mathcal{H}}(X)$ is a subset of $I(X)$ that does not depend on the cost function c, but only on the independence system $(E,\mathcal{F})$ and the set X. Moreover, $I_{\mathcal{H}}(X)$ is non-empty if $I(X) \neq 0$, a condition that guarantees that $\mathcal{H}$ always outputs a base.

3. Theoretical performance of greedy like algorithms for TSP

The main practical message of this and the next section is that one should be careful while using the classical greedy algorithm and its variations in combinatorial optimization: there are many instances of combinatorial optimization problems for which such algorithms will produce the unique worst possible solution. Moreover, this is true for several well-known optimization problems and the corresponding instances are not exotic, in a sense. This means that not always the paradigm of greedy optimization provides any meaningful optimization at all.

The first results of the kind mentioned in the previous paragraph were obtained in [19]:

Theorem 1. *For each $n \geq 2$ there is an instance of ATSP for which the Greedy Algorithm finds the unique worst possible tour.*

Gutin, Yeo and Zverovitch [19] proved Theorem 1 also for the *Nearest Neighbor* (`NN`) algorithm: start from an arbitrary vertex i_1 and go to a vertex $i_2 \neq i_1$ with shortest distance

from i_1; when in a vertex i_k, $k < n$, go to a vertex i_{k+1} with shortest distance from i_k among vertices not in the set $\{i_1, i_2, \ldots, i_{k-1}\}$. The proof for `NN` is correct for both ATSP and STSP. The proof of Theorem 1 itself (for `Greedy`) cannot be extended to STSP, but Theorem 1 holds also for STSP [18].

The `Greedy` and `NN` algorithms are special cases of greedy type algorithms and Bendall and Margot [5] proved the following result that generalizes all the results above.

Theorem 2. *Let $\mathcal{A}$ be a TSP greedy type algorithm. For each $n \geq 3$ there is an instance of TSP for which $\mathcal{A}$ finds the unique worst possible tour.*

At this stage the reader may ask the following natural question: 'Perhaps, it is true that every TSP heuristic has the domination number equal 1?' The answer is negative. In fact, there are many TSP heuristics (see, e.g., [15, 23]) which, for every instance $\mathcal{T}$ of TSP with n vertices ($n \geq 3$), produce a tour that is no longer than the average length of a tour in $\mathcal{T}$. Among these heuristics there are several fast construction heuristics. Thus, we can apply the following theorem to many TSP heuristics (we formulate this theorem for ATSP, but an almost identical result by Rublineckii [26] is known for STSP, see [13]):

Theorem 3. *Let $\mathcal{H}$ be an ATSP heuristics that, for every instance $\mathcal{T}$ of ATSP on $n \geq 2$ vertices, produces a tour that is no longer than the average length of a tour in $\mathcal{T}$. Then the domination number of $\mathcal{H}$ is at least $(n-2)!$ for each $n \neq 6$.*

This theorem was first proved by Sarvanov [27] for odd values of n and by Gutin and Yeo [15] for even values of n.

Sometimes, we are interested in TSP with only restricted range of weights. The following two results for this variation of TSP were obtained by Bang-Jensen, Gutin and Yeo [3].

Theorem 4. *Consider STSP as an $(E,\mathcal{H},W)$-optimization problem.*

a. *If $n \geq 4$ and $|W| \leq \left\lfloor \frac{n-1}{2} \right\rfloor$, then the greedy algorithm never produces the unique worst possible base (i.e., tour).*
b. *If $n \geq 3$, $r \geq n-1$ and $W = \{1, 2, \ldots, r\}$, then there exists a weight function $w : E \rightarrow \{1, 2, \ldots, r\}$ such that the greedy algorithm may produce the unique worst possible base (i.e., tour).*

Theorem 5. *Consider ATSP as an $(E,\mathcal{H},W)$-optimization problems. Let $n \geq 3$.*

a. *If $|W| \leq \left\lfloor \frac{n-1}{2} \right\rfloor$, then the greedy algorithm never produces the unique worst possible base (i.e., tour).*
b. *For every $r \geq \left\lceil \frac{n+1}{2} \right\rceil$ there exists a weight function $w : E(K_n^*) \rightarrow \{1, 2, \ldots, r\}$ such that the greedy algorithm may produce the unique worst possible base (i.e., tour).*

Notice that the above-mentioned theorems can be proved as corollaries of general results that hold for many $(E,\mathcal{H},W)$-optimization problems, see, e.g., [3, 5, 16].

Another ATSP greedy like heuristic, max-regret-fc (fc abbreviates First Coordinate), was first introduced by Ghosh et al. [8]. Extensive computational experiments in [8] demonstrated a clear superiority of max-regret-fc over the greedy algorithm and several other construction heuristics from [9]. Therefore, the result of Theorem 6 obtained by Gutin, Goldengorin and Huang [11] was somewhat unexpected.

Let Q be a collection of disjoint directed paths in K_n^* and let $V = V(K_n^*) = \{1, 2, \ldots, n\}$. An arc $a = ij$ is a *feasible addition* to Q if $Q \cup \{a\}$ is either a collection of disjoint paths or a tour in K_n^*.

Consider the following two ATSP heuristics: max-regret-fc and max-regret.

The heuristic *max-regret-fc* proceeds as follows. Set $W = T = 0$. While $W \neq V$ do the following: For each $i \in V \setminus W$, compute two lightest arcs ij and ik that are feasible additions to T, and

compute the difference $\Delta_i = |w(ij) - w(ik)|$. For $i \in V \setminus W$ with maximum Δ_i choose the lightest arc ij, which is a feasible addition to T and add ij to M and i to W.
The heuristic *max-regret* proceeds as follows. Set $W^+ = W^- = T = 0$. While $W^+ \neq V$ do the following: For each $i \in V \setminus W^+$, compute two lightest arcs ij and ik that are feasible additions to T, and compute the difference $\Delta_i^+ = |w(ij) - w(ik)|$; for each $i \in V \setminus W^-$, compute two lightest arcs ji and ki that are feasible additions to T, and compute the difference $\Delta_i^- = |w(ji) - w(ki)|$. Compute $i' \in V \setminus W^+$ with maximum $\Delta_{i'}^+$ and $i'' \in V \setminus W^-$ with maximum $\Delta_{i''}^-$. If $\Delta_{i'}^+ \geq \Delta_{i''}^-$ choose the lightest arc $i'j'$, which is a feasible addition to T and add $i'j'$ to M, i' to W^+ and j' to W^-. Otherwise, choose the lightest arc $j''i''$, which is a feasible addition to T and add $j''i''$ to M, i'' to W^- and j'' to W^+.
Notice that in max-regret-fc, if $|V \setminus W| = 1$ we set $\Delta_i = 0$. A similar remark applies to max-regret.
Theorem 6. *The domination number of both max-regret-fc and max-regret equals* 1 *for each* $n \geq 2$.

4. Theoretical performance of greedy like algorithms for MAP

In this section, we will first prove that the greedy algorithm for s-AP is of domination number 1. The proof shows that the greedy algorithm fails on instances that cannot be called 'exotic' in the sense that they do not have very large weights. For our proof we need the following definitions and lemma.
A vector h is *backward* if $\min\{h_i : 2 \leq i \leq s\} < h_1$; a vector h is *horizontal* if $h_1 = h_2 = \ldots = h_s$. A vector is *forward* if it is not horizontal or backward.
Lemma 7. *Let F be an assignment of s-AP* ($s \geq 2$). *Either all vectors of F are horizontal or F contains a backward vector.*
Proof: Let $F = \{f^1, f^2, \ldots, f^n\}$, where $f_1^i = i$ for each $1 \leq i \leq n$. Assume that not every vector of F is horizontal. We show that F has a backward vector. Suppose it is not true. Then F has a forward vector f^i. Thus, there is a subscript j such that $f_j^i > i$. By the pigeonhole principle, there exists a superscript $k > i$ such that $f_j^k \leq i$, i.e., f^k is backward; a contradiction. □
Theorem 8. *For each* $s \geq 2$, $n \geq 2$, *there exists an instance of s-AP for which* `Greedy` *will find the unique worst possible assignment.*
Proof: Consider some $M > n$ and let $E = \{e^1, e^2, \ldots, e^n\}$, where $e^i = (i, i, \ldots i)$ for every $1 \leq i \leq n$. We define the required instance $\mathcal{I}$ as follows: $w(e^i) = iM$ for each $1 \leq i \leq n$ and, for each $f \notin E$, $w(f) = \min\{f^i : 1 \leq i \leq s\} \cdot M + 1$.
Observe that `Greedy` will construct E. Let $F = \{f^1, f^2, \ldots, f^n\}$ be any other assignment, where $f_1^i = i$ for each $1 \leq i \leq n$. By Lemma 7, F has a backward vector f^k. Notice that

$$w(f^k) \leq (k-1)M + 1 \tag{1}$$

By the definition of the weights and (1),

$$\begin{aligned} w(F) &= \sum_{i=1}^{n} w(f^i) = \sum_{i \neq k} w(f^i) + w(f^k) \\ &\leq \sum_{i \neq k} (iM + 1) + (k-1)M + 1 \end{aligned}$$

$$= \sum_{i=1}^{n} iM + n - M$$
$$< \sum_{i=1}^{n} iM = w(E).$$

□

One can consider various greedy type algorithms for s-AP. One natural algorithm of this kind proceeds as follows: At the ith iteration, $i = 1, 2, \dots, n$ choose a vector e^i of minimum weight such that $e^i_1 = i$ and $\{e^1, e^2, \dots, e^i\}$ is a partial assignment. We call this algorithm the *First Coordinate Fixing* (FCF) heuristic. A simple modification of the proof of the first half of Theorem 10 in [5] shows the following:

Theorem 9. *For every $n \geq 1$, $s \geq 2$, there is an instance of s-AP and for every greedy type algorithm $\mathcal{H}$ for s-AP, there is an instance I of s-AP for which $\mathcal{H}$ finds the unique worst possible assignment.*

In Section 3, we considered the max-regret-fc and max-regret heuristics. In fact, max-regret was first introduced for 3-AP by Balas and Saltzman [2]. The s-AP heuristic *max-regret* proceeds as follows. We start from the empty partial assignment A and, at every iteration, we consider a partial assignment A. Set $V_d = \{1, 2, \dots, n\}$ for each $1 \leq d \leq s$. For each dimension d and each value $v \in V_d$ consider every vector $e \in X'$ such that $e_d = v$, where $X' \subset X$ is a set of feasible additional vectors, i.e., $A \cup \{e\}$ is a feasible partial assignment if $e \in X'$. If $X' \neq 0$, find two vectors $e^1_{\min}$ and $e^2_{\min}$ in the considered subset $Y_{d,v} = \{e \in X' : e_d = v\}$ such that $e^1_{\min} = \operatorname{argmin}_{e \in Y_{d,v}} w(e)$, and $e^2_{\min} = \operatorname{argmin}_{e \in Y_{d,v} \setminus \{e^1_{\min}\}} w(e)$. (If $|Y_{d;v}| = 1$, set $e^2_{\min} = e^1_{\min}$.) Select the pair (d, v) that corresponds to the maximum difference $w(e^2_{\min}) - w(e^1_{\min})$ and add the vector $e^1_{\min}$ for the selected (d, v) to the solution A. In computational experiments, Balas and Saltzman [2] compared the greedy algorithm with max-regret and concluded that max-regret is superior to the greedy algorithm with respect to the quality of solutions. However, after conducting wider computational experiments, Robertson [25] came to a different conclusion: the greedy algorithm and max-regret are of similar quality for 3-AP. Gutin, Goldengorin and Huang [11] share the conclusion of Robertson: both greedy algorithm and max-regret are of domination number 1 for s-AP for each $s \geq 3$. Moreover, there exists a common family of s-AP instances for which both heuristics find the unique worst assignment [11] (for each $s \geq 3$).

Similarly to TSP, we may obtain MAP heuristics of factorial domination number if we consider not-worth-than-average heuristics. This follows from the next theorem:

Theorem 10. *[11] Let $\mathcal{H}$ be a heuristic that for each instance of s-AP constructs an assignment of weight at most the average weight of an assignment. Then the domination number of $\mathcal{H}$ is at least $((n-1)!)^{s-1}$.*

Using Theorem 10, it is proved in [11] that the following heuristic is of domination number at least $((n-1)!)^{s-1}$. The *Recursive Opt Matching* (ROM) heuristic proceeds as follows. Initialize the solution by trivial vectors: $e^i = (i, i, \dots, i)$, $i = 1, 2, \dots, n$. On each jth iteration of the heuristic, $j = 1, 2, \dots, s-1$, calculate an $n \times n$ matrix $M_{i,v} = \sum_{e \in Y(j,i,v)} w(e)$, where $Y(j, i, v)$ is a set of all vectors $e \in X$ such that the first j coordinates of the vector e are equal to the first j coordinates of the vector e^i and the $(j+1)$th coordinate of e is v: $Y(j, i, v) = \{e \in X : e^k = e^i_k, 1 \leq k \leq j, \text{ and } e_{j+1} = v\}$. Let permutation π be a solution of the 2-AP for the matrix M. We set $e^i_{j+1} = \pi(i)$ for each $1 \leq i \leq n$.

The *Multi-Dimensionwise Variation* (MDV) heuristic is introduced in [12] as a local search heuristic for MAP. MDV starts from the trivial solution $e^i = (i, i, \dots, i)$, $i = 1, 2, \dots, n$. On each step it selects a nonempty set of distinct dimensions, $F \subsetneq \{1, 2, ..., s\}$. The corresponding dimensions are fixed, while the others are varied, and an $n \times n$ matrix $M_{i,j} = w(v^{i,j})$ is produced, where

$$v_d^{i,j} = \begin{cases} e_d^i & \text{if } d \in F \\ e_d^j & \text{if } d \notin F \end{cases} \qquad \text{for } d = 1, 2, \dots, s.$$

Let permutation ρ be a solution of the corresponding 2-AP. If ρ is not an identity permutation, the heuristic changes the s-AP assignment in the following way:

$$e_d^i = \begin{cases} e_d^i & \text{if } d \in F \\ e_d^{\rho(i)} & \text{if } d \notin F \end{cases} \qquad \text{for } i = 1, 2, \dots, n \text{ and } d = 1, 2, \dots, s.$$

There are $2^s - 2$ distinct sets of the fixed dimensions F, but a half of them may be omitted since there is no difference whether to fix the selected dimensions and to vary the others, or to vary the selected dimensions and to fix the others. So, every iteration of the heuristic tries $2^{s-1} - 1$ distinct sets F. If no improvement was obtained during an iteration, the algorithm terminates. We have the following:

Theorem 11. *The domination number of* MDV *equals* $(2^{s-1} - 1)(n! - 1) + 1$.

Proof: Let a vector $e^i = (i, i, \dots, i) \in X$ for every $i = 1, 2, \dots, n$ and let vectors $e^{(i,j,F)} \in X$, $1 \le i \ne j \le n$, be defined as $e_k^{(i,j,F)} \in \{i, j\}$ for every $k = 1, 2, \dots, s$ and $e_k^{(i,j,F)} = i$ if and only if $k \in F$. We assign the weights as follows: $w(e^i) = 1$ for every $i = 1, 2, \dots, n$, $w(e^{(i,j,F)}) = 2$ for each of the $2^{s-1} - 1$ sets F and $1 \le i \ne j \le n$ and $w(e) = 0$ for every vector $e \in X$ that has at least three coordinates of different value. Let F_0 be the first set F chosen by MDV. Observe that for F_0 MDV outputs the trivial assignment $e^1, \dots, e^n$, which is the best among $n!$ assignments.

For every other F MDV outputs the trivial assignment which is better than $n!$–1 assignments. Further iteration will output the trivial assignment as well. Thus, we conclude that the trivial assignment is the best among at most $(2^{s-1} - 1)(n! - 1)+1$ assignments considered by MDV and, hence, the domination number of MDV is at most $(2^{s-1} - 1)(n! - 1) + 1$.

Now consider the last iteration of MDV. No improvement is made, and, thus, a solution with which we started the iteration will not change during the iteration. By permuting the elements of $X_2, X_3, \dots, X_s$ (recall that $X = X_1 \times X_2 \times \dots \times X_s$), if needed, we may assume, without loss of generality, that the solution at the start of the last permutation is the trivial assignment. Since $e^{(i,j,F')} \ne e^{(i,j,F'')}$ provided $F' \ne F''$ and $F' \ne \{1, 2, \dots, s\} \setminus F''$, as above, we can see that the trivial assignment is the best among exactly $(2^{s-1} - 1)(n! - 1)+1$ distinct assignments of the last iteration. Thus, $(2^{s-1} - 1)(n! - 1) + 1$ is a lower bound on the domination number of MDV. Since this lower bound is also an upper bound on the domination number, we are done. □

This theorem and the result just after Theorem 10 show that ROM is of larger domination number than MDV for every fixed $s \ge 3$ for every n large enough. This is in contrast with the experimental results reported in Section 6, where the solutions obtained by MDV are almost

always better than those produced by ROM. There is no contradiction in the two comparisons as they measure different sides of the quality of the two heuristics: the worst case behavior vs. the performance on some particular families of MAP instances.

5. Empirical evaluation of greedy like algorithms for TSP

We considered three ATSP heuristics in our experiments: Greedy, NN, and Patch Cycles (Patch) [22].

The Greedy heuristic is implemented as follows. Construct an array of all arcs x_iy_i, $x_i \neq y_i$, $1 \le i \le n(n-1)$, and sort this array by the arc weight: $w(x_iy_i) \le w(x_{i+1}y_{i+1})$ for every $1 \le i < n(n-1)$. Let $prev(i)$ be the vertex preceding the vertex i in the tour, and let $next(i)$ be the vertex succeeding the vertex i in the tour. Initialize $prev(i) = next(i) = 0$ for every $1 \le i \le n$. While the solution is incomplete, it consists of several separate components. For a vertex i let $id(i)$ be the identifier of the vertex i component. Initialize $id(i) = i$ for every $1 \le i \le n$. On every kth step of the heuristic try to add the arc x_ky_k to the current solution, i.e., check whether $next(x_k) = 0$ and $prev(y_k) = 0$ and $id(x_k) \neq id(y_k)$. If all the conditions are met, set $next(x_k) = y_k$, $prev(y_k) = x_k$, and $id(i) = id(y_k)$ for every $i \in \{ j : id(j) = id(x_k)\}$. When $n - 1$ arcs are added to the solution, the algorithm closes the cycle and stops.

The details on the NN heuristic are available in Section 3.

The Patch heuristic proceeds as follows. Let π be a solution of the assignment problem (AP) for the distance matrix of ATSP. Construct vertex-disjointed cycles c^i based on the AP solution π such that $c^i_{j+1} = \pi(c^i_j)$ for every $1 \le j < s_i$ and $c^i_1 = \pi(c^i_{s_i})$, where s_i is the number of vertices in the ith cycle. Let m be the number of cycles, such that $\sum_{i=1}^{m} s_i = n$. If $m = 1$, the cycle c^1 is the optimal solution of ATSP and no further actions are required. Otherwise select two longest cycles (i.e., the cycles with the maximum values of s_i) and patch them by removing edges x_1x_2 from the first of them and y_1y_2 from the second one such that the value $w(x_1y_2)+w(y_1x_2) - w(x_1x_2) - w(y_1y_2)$ is minimized. Repeat this procedure until there is just one cycle, that is considered as a solution.

All the heuristics in this section and in Section 6 are implemented in Visual C++. The evaluation platform is based on AMD Athlon 64 X2 3.0 GHz processor.

The experiment results are reported in Tables 1 and 2. Table 1 includes the results for randomly generated instances of nine classes (for details see [13]). Ten instances of size 100, ten instances of size 316, three instances of size 1000, and one instance of size 3162 are considered for every instance class. The solution quality is presented by percent above the Held-Karp (HK) lower bound [20, 21].

Table 2 includes the results for several real-world ATSP instances from TSPLIB [24] and some other sources [20]. The solution quality is presented by percent above the best known solutions.

One can see that Patch clearly outperforms both Greedy and NN with respect to the solution quality, and the NN solutions are usually better than the Greedy ones (though Greedy slightly outperform NN on average with respect to the solution quality for the real-world instances). NN is much faster than both Greedy and Patch, while Patch is faster than Greedy for small instances and slower for the large ones. Johnson et al. [20] showed that, along with Patch, there are some other ATSP heuristics that are relatively fast and normally produce solutions that are much better than those obtained by Greedy and NN. (Some ATSP heuristics of good quality are also studied in [8].) Thus, it appears that Greedy

should never be used in practice and NN is of interest only if a very fast heuristic is required.

6. Empirical evaluation of greedy like algorithms for MAP

In this chapter we consider four MAP heuristics: Greedy, First Coordinate Fixing (FCF), Recursive Opt Matching (ROM), and Multi-Dimensionwise Variation (MDV).

Greedy proceeds as follows. Let $A = \emptyset$ be a partial assignment and B an array of vectors. While $|A| < n$, i.e., A is not a full assignment, the following steps are repeated. Scan the weight matrix to fill array B with k vectors corresponding to k minimal weights and sort B in non-decreasing order. For each vector $e \in B$, starting from the lightest, check whether $A \cup \{e\}$ is a feasible partial assignment and, if so, add e to A. Note, that during the second and further cycles we scan not the whole weight matrix but only a subset $X' \subset X$ of the vectors that can be included into the partial assignment A with the feasibility preservation: $A \cup \{x\}$ is a partial assignment for any $x \in X'$. The size of the array B is calculated as $k = \min\{128, |X'|\}$.

The details on FCF, ROM and MDV heuristics are available in Section 4. As in [12], the number of MDV iterations was artificially restricted to 10.

The testbed includes three instance families: Random, Composite, and GP, discussed in [12].

In *Random Instance Family* (Random) the weight assigned to a vector is a random uniformly distributed integer value in the interval $[a, b - 1]$. We set $a = 1$ and $b = 101$. It is proved (see [12]) that the optimal solutions of large Random instances are very likely to be of weight an, so we assume in our experiments that the optimal solutions of the considered Random instances are exactly n.

The *Composite Instance Family* (Composite) is a family of semi-random instances. They were introduced by Crama and Spieksma for 3-AP as a problem T [7]. We extend this family for s-AP.

Let $d^1, d^2, \ldots, d^s$ be $n \times n$ matrices of non-negative uniformly distributed random integers in the interval $[a, b - 1]$. Let us consider a graph $G(X_1 \cup X_2 \cup \ldots \cup X_s, (X_1 \times X_2) \cup (X_2 \times X_3) \cup \ldots \cup (X_{s-1} \times X_s) \cup (X_1 \times X_s))$, where the weight of an edge $(i, j) \in X_k \times X_{k+1}$ is $d^k_{i,j}$ for $1 \le k < s$ and the weight of an edge $(i, j) \in X_1 \times X_s$ is $d^s_{i,j}$. In this interpretation of s-AP, the objective is to find a set of n vertex-disjoint s-cycles $C \subset X_1 \times X_2 \times \ldots \times X_s$ such that the total weight of all edges covered by the cycles C is minimized.

In other words, $w(e) = d^1_{e_1,e_2} + d^2_{e_2,e_3} + \ldots + d^{s-1}_{e_{s-1},e_s} + d^s_{e_1,e_s}$.

The *GP Instance Family* (GP) contains pseudo-random instances with predefined optimal solutions. GP instances are generated by an algorithm given by Grundel and Pardalos [10]. The generator is naturally designed for s-AP for arbitrary large values of s and n. The GP generator is relatively slow and, thus, is was impossible to experiment with large GP instances.

The results of the experiments are reported in Tables 3, 4, and 5. One can see that Greedy is significantly slower than the FCF heuristic, while its solution quality is not significantly better than the FCF's one. ROM outperforms or have very close results to Greedy with respect to both the solution quality and the running times. MDV clearly outperforms all other heuristics with respect to the solution quality, and it is the fastest algorithm for

`Random` and `Composite` instances. For `GP` instances `FCF` and `ROM` are faster than `MDV`. Based on the experimental data, `MDV` is definitely the overall winner.

7. References

[1] D.L. Applegate, R.E. Bixby, V. Chvátal and W.J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, 2006.

[2] E. Balas, and M.J. Saltzman, An algorithm for the three-index assignment problem, *Operations Research* 39 (1991), 150–161.

[3] J. Bang-Jensen, G. Gutin and A. Yeo, When the greedy algorithm fails, *Discerete Optimization* 1 (2004), 121–127.

[4] H. Bekker, E.P. Braad and B. Goldengorin, Using bipartite and multidimentional matchings to select roots of a system of polynomial equations. In Proc. ICCSA'05, Lecture Notes in Computer Science 3483 (2005), 397–406.

[5] G. Bendall and F. Margot, Greedy Type Resistance of Combinatorial Problems, *Discrete Optimization* 3 (2006), 288–298.

[6] R.E. Burkard and E. C¸ ela, Linear assignment problems and extensions, in Handbook of Combinatorial Optimization, Kluwer, Dordrecht, 1999, (Z. Du and P. Pardalos, eds.), 75–149.

[7] Y. Crama and F.C.R. Spieksma, Approximation algorithms for threedimensional assignment problems with triangle inequalities, *Europ. J. Operational Res.* 60 (1992), 273–279.

[8] D. Ghosh, B. Goldengorin, G. Gutin and G. J¨ager, Tolerance-based greedy algorithms for the traveling salesman problem, *Communications in DQM* 10 (2007), 52–70.

[9] F. Glover, G. Gutin, A. Yeo and A. Zverovich, Construction heuristics for the asymmetric TSP, *European Journal of Operational Research* 129 (2001), 555–568.

[10] D.A. Grundel and P. M. Pardalos, Test problem generator for the multidimensional assignment problem, *Comput. Optim. Appl.*, 30(2):133146, 2005.

[11] G. Gutin, B. Goldengorin, and J. Huang, 'Worst Case Analysis of Max-Regret, Greedy and Other Heuristics for Multidimensional Assignment and Traveling Salesman Problems', *Lect. Notes Computer Sci.*, 4368 (2006), 214–225.

[12] G. Gutin and D. Karapetyan, Local Search Heuristics For The Multidimensional Assignment Problem, Preprint arXiv:0806.3258v2.

[13] G. Gutin and A.P. Punnen (eds.), *The Traveling Salesman Problem and its Variations* , Kluwer, 2002 and Springer-Verlag, 2007.

[14] G. Gutin, A. Vainshtein and A. Yeo, When greedy-type algorithms fail, unpublished manuscript, 2002.

[15] G. Gutin and A. Yeo, Polynomial approximation algorithms for the TSP and the QAP with a factorial domination number, *Discrete Appl. Math.* 119 (2002), 107–116.

[16] G. Gutin and A. Yeo, Anti-matroids, *Oper. Res. Lett.* 30 (2002), 97–99.

[17] G. Gutin and A. Yeo, Domination Analysis of Combinatorial Optimization Algorithms and Problems. *Graph Theory, Combinatorics and Algorithms: Interdisciplinary Applications* (M.C. Golumbic and I.B.-A. Hartman, eds.), Springer-Verlag, 2005.

[18] G. Gutin and A. Yeo, The Greedy Algorithm for the Symmetric TSP. *Algorithmic Oper. Res.* 2 (2007), 33–36.

[19] G. Gutin, A. Yeo and A. Zverovitch, Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for the TSP, *Discrete Appl. Math.* 117 (2002), 81–86.

[20] D.S. Johnson, G. Gutin, L.A. McGeoch, A. Yeo, X. Zhang and A. Zverovitch, Experimental Analysis of Heuristics for ATSP, Chapter 10 in [13].

[21] D.S. Johnson and L.A. McGeoch, Experimental Analysis of Heuristics for STSP, Chapter 9 in [13].

[22] R.M. Karp, A patching algorithm for the non-symmetric traveling salesman problem, *SIAM J. Comput.*, 8:561573, 1979.

[23] A.P. Punnen, F. Margot and S.N. Kabadi, TSP heuristics: domination analysis and complexity, *Algorithmica* 35 (2003), 111–127.

[24] G. Reinelt, TSPLIB—A traveling salesman problem library, ORSA J. Comput. 3 (1991), 376-384, http://www.crpc.rice.edu/softlib/ tsplib/.

[25] A.J. Robertson, A set of greedy randomized adaptive local search procedure implementations for the multidimentional assignment problem. *Computational Optimization and Applications* 19 (2001), 145–164.

[26] V.I. Rublineckii, Estimates of the Accuracy of Procedures in the Traveling Salesman Problem, *Numerical Mathematics and Computer Technology* no. 4 (1979), 18–23 [in Russian].

[27] V.I. Sarvanov, The mean value of the functional of the assignment problem, *Vestsi Akad. Navuk BSSR Ser. Fiz. -Mat. Navuk* no. 2 (1976), 111–114 [in Russian].

Family	n	Percents above HK			Running times, ms		
		Greedy	NN	Patch	Greedy	NN	Patch
amat	100	176.4	203.7	11.1	2.03	0.04	0.28
	316	246.1	231.3	6.5	25.97	0.45	2.91
	1000	283.4	337.4	2.7	319.18	4.38	50.34
	3162	344.1	397.3	1.9	3364.30	44.60	816.12
coin	100	26.9	27.7	16.0	1.67	0.05	0.16
	316	28.9	25.5	18.6	18.37	0.54	1.41
	1000	26.9	25.2	17.4	208.20	4.30	34.18
	3162	26.8	25.6	19.0	2382.02	46.26	431.35
crane	100	30.1	43.0	9.3	2.12	0.05	0.38
	316	45.3	60.0	13.6	26.03	0.46	8.92
	1000	66.1	81.0	31.1	338.89	5.10	156.46
	3162	105.9	113.2	58.4	3357.79	43.86	2366.64
disk	100	99.2	90.2	6.7	1.98	0.04	0.21
	316	148.7	107.7	2.2	26.37	0.40	5.72
	1000	296.1	117.3	0.9	321.94	4.35	205.25
	3162	566.5	162.0	0.3	3692.79	44.31	6935.62
rtilt	100	318.5	28.5	17.9	2.57	0.04	0.70
	316	651.6	28.2	17.9	26.59	0.41	10.06
	1000	1219.4	27.9	19.5	391.19	5.46	259.75
	3162	2260.9	24.2	21.3	3537.03	46.79	4477.49
shop	100	48.1	16.7	1.2	2.03	0.04	1.37
	316	55.9	14.8	0.6	20.90	0.43	26.89
	1000	60.4	13.5	0.4	209.73	4.61	939.74
	3162	63.8	11.9	0.2	2236.86	44.69	26662.96
stilt	100	103.8	32.6	25.3	2.08	0.05	0.88
	316	141.4	29.1	23.4	26.09	0.39	13.82
	1000	166.3	27.2	25.2	315.08	5.03	156.53
	3162	172.0	24.1	23.4	3428.08	44.22	2094.31
super	100	3.9	8.0	1.8	0.65	0.04	0.17
	316	4.1	8.8	2.7	6.70	0.94	2.55
	1000	4.5	9.9	4.2	73.66	4.75	70.06
	3162	4.9	10.1	6.1	815.01	61.75	959.90
tmat	100	26.3	38.0	0.9	2.11	0.04	0.21
	316	25.7	37.1	0.7	24.77	0.43	4.11
	1000	24.7	37.8	0.2	220.47	4.23	75.23
	3162	25.2	35.4	0.0	2041.57	45.73	1266.14
Avg.	100	92.6	54.3	10.0	1.92	0.04	0.48
	316	149.8	60.3	9.6	22.42	0.50	8.49
	1000	238.7	75.2	11.3	266.48	4.69	216.39
	3162	396.7	89.3	14.5	2761.72	46.91	5112.28

Table 1. ATSP heuristics experiment results for randomly generated instances.

		Percents above BK			Running times, ms		
Instance	n	Greedy	NN	Patch	Greedy	NN	Patch
atex8	600	32.2	20.8	35.1	0.066	0.001	0.006
big702	702	17.9	35.6	0.0	0.090	0.002	0.028
code198	198	∞	∞	14.0	0.002	0.000	0.001
code253	253	59.0	∞	3.2	0.003	0.000	0.001
dc112	112	1.3	1.6	0.2	0.001	0.000	0.001
dc126	126	2.9	4.0	0.8	0.004	0.000	0.001
dc134	134	0.9	1.7	0.0	0.002	0.000	0.001
dc176	176	2.2	2.7	0.7	0.003	0.000	0.001
dc188	188	1.4	2.3	0.1	0.003	0.000	0.001
dc563	563	1.6	2.6	0.3	0.042	0.002	0.036
dc849	849	0.3	0.7	0.0	0.050	0.003	0.143
dc895	895	1.3	2.2	0.2	0.115	0.004	0.062
dc932	932	0.9	1.4	0.2	0.105	0.003	0.061
ftv100	101	60.6	44.1	1.9	0.002	0.000	0.000
ftv110	111	38.1	47.2	2.5	0.002	0.000	0.000
ftv120	121	45.3	38.8	6.6	0.002	0.000	0.000
ftv130	131	40.2	32.9	3.8	0.002	0.000	0.000
ftv140	141	38.1	42.2	3.5	0.003	0.000	0.000
ftv150	151	43.2	36.4	2.6	0.003	0.000	0.000
ftv160	161	50.4	51.5	1.7	0.003	0.000	0.000
ftv170	171	42.8	42.4	2.0	0.004	0.000	0.001
kro124p	100	21.0	31.1	13.8	0.002	0.000	0.000
rbg323	323	7.9	30.8	0.0	0.011	0.000	0.007
rbg358	358	8.0	55.8	0.0	0.012	0.000	0.009
rbg403	403	0.8	43.4	0.0	0.014	0.001	0.010
rbg443	443	0.7	44.2	0.0	0.021	0.001	0.012
Avg.		20.8	25.7	3.6	0.022	0.001	0.015

Table 2. ATSP heuristics experiment results for real-world instances. Here ∞ stands for '> 10^5' and BK for 'best known.'

		Percents above optimal				Running times, ms			
s	n	Greedy	FCF	ROM	MDV	Greedy	FCF	ROM	MDV
3	100	101.0	101.0	67.3	12.9	16.8	3.3	8.3	8.0
3	200	42.0	50.5	15.5	0.1	112.9	29.2	69.8	29.6
3	300	27.2	37.6	3.8	0.0	324.0	91.4	221.8	51.3
4	50	105.2	181.0	128.2	34.2	72.9	18.4	49.4	4.9
4	80	74.0	64.9	76.4	8.0	426.8	114.0	316.1	20.0
5	40	159.3	139.5	168.8	29.5	1072.4	232.2	803.5	8.7
6	20	277.5	302.5	327.0	41.0	693.8	156.1	509.1	3.8
7	12	420.8	425.0	475.0	57.5	412.5	84.6	294.5	2.7
8	8	548.8	550.0	723.8	62.5	207.2	44.3	152.2	2.2
Avg.		195.1	205.8	220.6	27.3	371.02	85.95	269.41	14.57

Table 3. MAP heuristics experiment results for `Random` instances.

		Percents above best known				Running times, ms			
s	n	Greedy	FCF	ROM	MDV	Greedy	FCF	ROM	MDV
3	100	21.3	38.1	15.9	0.0	15.9	3.6	8.9	12.0
3	200	19.3	36.9	18.6	0.0	158.8	29.2	67.2	65.8
3	300	8.5	22.8	14.6	0.0	644.3	92.5	221.7	160.3
4	50	32.3	39.8	12.4	0.0	83.3	19.1	50.0	7.9
4	80	23.8	36.7	7.8	0.0	588.1	118.8	312.7	27.9
5	40	28.9	42.5	4.3	0.0	1350.9	241.6	932.5	11.4
6	20	35.3	43.4	3.8	0.0	812.9	142.1	563.6	5.5
7	12	36.0	43.8	5.7	0.0	504.1	85.0	294.8	4.0
8	8	28.5	34.9	7.3	0.0	233.9	45.4	146.5	3.1
Avg.		26.0	37.7	10.1	0.0	488.03	86.36	288.65	33.11

Table 4. MAP heuristics experiment results for `Composite` instances.

		Percents above optimal				Running times, ms			
s	n	Greedy	FCF	ROM	MDV	Greedy	FCF	ROM	MDV
3	50	9.9	11.7	12.2	6.8	4.4	0.6	1.5	3.5
3	80	7.6	8.2	11.6	4.5	22.0	2.0	4.9	13.3
3	100	5.6	7.5	9.8	3.7	52.4	4.0	9.7	28.8
4	20	11.7	12.2	5.3	6.8	7.0	0.7	1.6	0.9
4	30	8.8	9.2	2.2	4.9	49.1	2.8	7.0	2.1
5	8	28.7	25.9	17.6	8.8	1.2	0.2	0.3	0.4
5	12	12.8	17.1	9.2	4.8	9.1	1.2	2.6	0.6
6	8	24.6	20.1	13.6	3.3	6.5	0.9	2.6	0.7
7	5	27.0	27.3	19.5	5.1	1.7	0.4	0.8	0.6
8	4	21.4	30.2	28.1	5.2	1.9	0.3	0.8	0.7
Avg.		15.8	16.9	12.9	5.4	15.5	1.3	3.2	5.2

Table 5. MAP heuristics experiment results for `GP` instances.

17

Greedy Methods in Plume Detection, Localization and Tracking

Huimin Chen
University of New Orleans, Department of Electrical Engineering
2000 Lakeshore Drive, New Orleans, LA 70148,
USA

1. Introduction

Greedy method, as an efficient computing tool, can be applied to various combinatorial or nonlinear optimization problems where finding the global optimum is difficult, if not computationally infeasible. A greedy algorithm has the nature of making the locally optimal choice at each stage and then solving the subproblems that arise later. It iteratively makes one greedy choice after another, reducing each given problem into a smaller one. In other words, a greedy algorithm never reconsiders its choices. Clearly, greedy method often fails to find the globally optimal solution. However, a greedy algorithm can be proven to yield the global optimum for a given class of problems such as Kruskal's algorithm and Prim's algorithm for finding minimum spanning tree, Dijkstra's algorithm for finding single-source shortest path, and the algorithm for finding optimum Huffman tree [5]. Even for some optimization problems proven to be NP hard, a greedy algorithm may generate near optimal solution with high probability if one exploits the problem structure properly. In this chapter, we focus on the optimization problems arising from plume detection, localization and tracking and provide convincing argument on the usefulness of greedy algorithms.

Detection, identification, localization, tracking and prediction of chemical, biological or nuclear propagation is crucial to battlefield surveillance and homeland security. In addition, post-accident management for public protection relies critically on detecting and tracing dangerous gas leakages promptly. The determination of source origins and release rates is useful for the forecast of gas concentration in the atmosphere and for the management staff to prioritize off-site evacuation plans. A lot of research has been focused on detecting and localizing single or multiple plume sources with autonomous vehicles [11] or sensor networks such as [22] for a vapor-emitting source, [2] for a nuclear source, and [14, 15] for a chemical source. In [12] the plume detection and localization problem is formulated as abrupt change detection using sparse sensor measurements. The development of a large scale testbed has been reported in [8] for plume detection, identification and tracking. In [3] dense sensor coverage has been used for radioactive source detection while [26] showed that using three error-free intensity sensors, one can identify the plume origin to any desired accuracy with high probability. Although this approach offers an effective solution with linear complexity of the hypothesis space, a major limitation is that the continuous time dynamic model of plume propagation has to be in the product form.

When the sensing devices can not provide accurate plume concentration readings, plume tracking relies heavily on the sensor coverage instead of the physics-based propagation model. In this case, hidden Markov model (HMM) offers a flexible tool to model the uncertainty of plume propagation motion in the air. It has been applied to plume mapping in [11] and chemical detection in [23]. The main issue of HMM resides in the time varying state transition probabilities which are not readily available from the physics based plume propagation equation. A viable approach is to use the generalized HMM with fuzzy measure and fuzzy integral [20]. The resulting plume localization problem becomes finding the most likely source sequence based on a fuzzy HMM. Existing algorithms of Viterbi type [20, 21] can be very inefficient when the size of the hidden state is large. Recently, [19] showed that the average complexity of finding the maximum likelihood sequence can be much lower than that using Viterbi algorithm for an HMM in the high SNR regime. Motivated by the theoretical result in [19], we propose a decoding algorithm of greedy type to obtain a candidate source path and search only for state sequences within a constrained Hamming distance from the candidate plume path. Our method is applicable to a general class of fuzzy measures and fuzzy integrals being used in fuzzy HMM. We compare the localization error using our algorithm with that using fuzzy Viterbi algorithm in a plume localization scenario with randomly deployed sensors. Simulation results indicate that the proposed greedy algorithm is much faster than fuzzy Viterbi algorithm for plume tracing over a long observation sequence when the localization error probability is small.

When the sensing devices provide fairly accurate concentration readings of the sources, one would expect that plume localization and release sequence estimation can be solved jointly. However, despite the abundant literature in plume detection [3, 23, 24] and localization [11, 30, 31], limited efforts have been made toward solving the joint problem of source localization and parameter estimation. The main reason is that even finding linear parameters related to the source release rate is an ill-posed problem and one has to impose certain regularization technique to avoid potential overfit. To solve the plume identification and parameter estimation jointly, we adopt the least squares technique based on the solution to the advection-diffusion equation [16, 17] and impose l_p-regularization for $0 \le p \le 1$ [4, 7] to characterize the sparsity of the unknown source release rate signal. We also discuss its advantage over the popularly used l_2-regularization. The accuracy of source parameter estimation is examined for the cases where both the number of sources and the corresponding locations are unknown. Since the resulting optimization problem is nonlinear and involves both discrete and continuous variables, we apply a greedy approach to identify and localize one source at a time. It is very efficient and can be interpreted as greedy basis pursuit [13].

The rest of the chapter is organized as follows. Section 2 formulates the plume localization problem using multiple binary detection sensors as maximum likelihood decoding over fuzzy HMM. Greedy algorithm is applied to maximum likelihood sequence estimation where the complexity comes from the fine resolution of the quantized surveillance area. Section 3 introduces the joint plume localization and source parameter estimation problem. Greedy algorithm is applied to source identification where the computational complexity mainly comes from the aggregation of unknown number of sources. Section 4 presents a concluding summary and discusses when one can expect good performance using greedy method.

2. Sequence estimation using fuzzy hidden Markov model

This section focuses on the maximum likelihood sequence estimation where the problem lends itself with a combinatorial structure similar to a decoding problem. We start with continuous time plume propagation model in Section 2.1 and then discuss the discrete time Markov approximation of the plume source as well as the sensor measurement model in Section 2.2. Section 2.3 presents the sequence estimation problem over a fuzzy hidden Markov model (HMM) using Viterbi and greedy heuristic algorithm. Section 2.4 provides simulation study on tracing a single plume source with unknown source location and initial releasing time.

2.1 Gaussian puff plume propagation model

It is challenging to accurately model the spatial and temporal distribution of a contaminant released into an environment due to the inherent randomness of the wind velocities in turbulent flow. Here we adopt a continuous time plume propagation model of instantaneous release type given in [28]. A plume consisting of particles or gases has the concentration c satisfying the following continuity equation

$$\frac{\partial c}{\partial t} + \frac{\partial}{\partial x_j}(u_j c) = D\frac{\partial^2 c}{\partial x_j^2} + R(c,T) + S(\mathbf{x},t)$$

where u_j is the j-th component of the wind velocity; D is the molecular diffusion coefficient; R is the rate of particle generation depending on the temperature T; S is the rate of aggregation of particles at location $\mathbf{x}$ and time t. In a perfectly known wind field where one knows the wind velocities at all locations, there will not be any turbulent diffusion. However, due to the randomness of the wind velocities, one can only expect that the mean concentration to satisfy the atmospheric diffusion equation

$$\frac{\partial \bar{c}}{\partial t} + \bar{u}_j\frac{\partial \bar{c}}{\partial x_j} = K_{jj}\frac{\partial^2 \bar{c}}{\partial x_j^2} + S(\mathbf{x},t)$$

where $\bar{u}_j$ is the j-th component of mean wind velocity and K_{jj} is the eddy diffusivity assuming molecular diffusion is negligible relative to turbulent diffusion. Assuming $S(\mathbf{x}, t) = 0$ (instantaneous release) without any boundary condition, one can obtain the closed form solution to the above partial differential equation, which in the two dimensional case is

$$\bar{c}(x,y,t) = \frac{c_0}{4\pi t\sqrt{K_x K_y}} e^{-\frac{(x-\bar{u}_x t)^2}{4K_x t} - \frac{(y-\bar{u}_y t)^2}{4K_y t}}$$

where x and y are the axis of the Cartesian coordinate system centered at the plume origin. In practice, one may not have the knowledge of the mean wind velocity at any location and it can also be time varying. In order to accommodate the uncertainty due to the aggregation of the plume release and the wind turbulence, in the sequel, we consider a dynamic model with both time and spatial transition following a Markov chain.

2.2 Approximate plume propagation dynamics and measurement model

We assume that the search region is partitioned into N cells indicating the possible origins of the plume source. The centroid of cell i is denoted by (q_{xi}, q_{yi}) for $i = 1, \ldots, N$. Sensors are

homogeneous and randomly deployed in the search region. Time is discretized by the sensing interval Δt where chemical intensity is measured in the neighborhood of each sensor's location. If the intensity is above a predetermined threshold, then a sensor will declare its detection of chemical plume. Thus at any time instant k, a binary sequence $\mathbf{y}_k$ is obtained from M sensors located at (r_{xi}, r_{yi}) indicating a possible chemical detection for $i = 1, \ldots, M$. We assume that the flow velocity $(v_{xi}(k), v_{yi}(k))$ is also recorded by sensor i at time k $(\forall i, k)$. Denote by $x(j)_k$ the hidden state of cell j at time k taking binary values indicating whether it contains detectable chemicals. Denote by $\mathbf{x}_k = [x(1)_k\, x(2)_k \ldots x(N)_k]'$ the plume map at time k. Denote by $\mathbf{y}_{1:K} = [\mathbf{y}_1 \ldots \mathbf{y}_K]$ the detection sequence up to time K and, accordingly, $\mathbf{x}_{1:K} = [\mathbf{x}_1 \ldots \mathbf{x}_K]$ the possible plume sequence up to time K. The plume mapping problem can be written as finding the most likely plume sequence

$$\hat{\mathbf{x}}_{1:K} = \arg \max_{\mathbf{x}_{1:K} \in \{0,1\}^{N K}} p(\mathbf{y}_{1:K} | \mathbf{x}_{1:K}) \tag{1}$$

where a statistical model between the state and observation sequence is assumed and the maximum likelihood (ML) criterion is used. From the ML estimate of the state sequence, one can identify the origin of the plume and its initial releasing time. Note that using the above formulation, one can also estimate the origins of multiple plumes at unknown and possibly different releasing times. The major difficulty lies in the availability of state and measurement model at any given time.

2.3 Fuzzy hidden Markov model and maximum likelihood decoding

2.3.1 Hidden Markov plume model

Two methods are popularly used in modeling plume propagation: numerical solution to the advection-dispersion equation and random simulation [9]. In this section, we use random simulation to generate realistic plume propagation sequence when evaluating the state sequence estimation algorithm. In a hidden Markov plume model, the state sequence $\{\mathbf{x}_k\}$ is assumed to be a Markov chain. At any time k, a cell i has a probability p_b originating a new plume release if it contains no plume at k - 1. A cell i has a probability p_c releasing the same amount of plume at time k if it contains a plume source at k -1. A cell j has detectable plume at time k coming from the source in cell i at time k - 1 with probability $p_d(i)$ depending on the source intensity Q and minimal detectable intensity C. Without the presence of wind, we have $p_d(i) = 0$ for Gaussian plume if

$$r_{ij}^2 = (q_{xi} - q_{xj})^2 + (q_{yi} - q_{yj})^2 > 4D\Delta t \log(Q/C) \tag{2}$$

where D is the diffusion coefficient. With known flow velocity at cell i, the detectable region can be modified accordingly. Thus we denote by $A(k)$ the state transition probability matrix of size $2^N \times 2^N$ with element $a_{mn}(k)$ indicating $p(\mathbf{x}_k = n \,|\, \mathbf{x}_{k-1} = m)$ where m and n represent two binary sequences of length N. For each sensor, the probability of false alarm is assumed to be very low when there is no detectable plume in its neighborhood. The detection probability depends on the distance d between the sensor location and the centroid of the nearest cell which contains detectable plume. The following crude model is assumed.

$$P_D = 1 - e^{-\alpha/d} \tag{3}$$

where α is chosen such that the detection probability at the edge of the cell is 1-C/Q. Thus we have specified the observation model B of size $2^N \times 2^M$ for M sensors with element b_{nl} indicating the probability $p(\mathbf{y}_k = l \,|\, \mathbf{x}_k = n)$. The hidden Markov plume model is represented by the parameter vector $\Lambda = [\pi, \{A(k)\}_{k=1}^K, B]$ where π is the initial probability of the state. Note that the hidden Markov plume model is nonstationary since the state transition matrix is time varying. The model parameter Λ is difficult to learn from experiments since it requires large training sets with various wind conditions.

2.3.2 Fuzzy hidden Markov model

Fuzzy hidden Markov model (FHMM) is a natural extension of the classical hidden Markov model with fuzzy measure and fuzzy integral. The theoretical framework was first proposed in [20] and applied to handwritten word recognition in [21]. Here we briefly highlight the key components in FHMM and its advantage over a nonstationary hidden Markov plume model.

FHMM replaces the probability measure used in the classical HMM with the fuzzy measure. A fuzzy measure μ on the state space X is a mapping from subset of X onto the unit interval μ: $2^X \rightarrow [0, 1]$ such that $\mu(\phi) = 0$, $\mu(X) = 1$, and if $E \subset F$, then $\mu(E) \leq \mu(F)$. To combine the evidences from different sensor measurements, the concept of fuzzy integral is introduced to replace the classical probabilistic inference. For a discrete set $X = \{x_1, \ldots, x_n\}$, the Choquet integral of a function h with respect to a fuzzy measure μ is computed as follows.

$$\int_X h(x)d\mu = \sum_{i=1}^{n} [h(x_i) - h(x_{i-1})]\mu_i^n \tag{4}$$

where $h(x_0) = 0$, $h(x_1) \leq h(x_2) \leq \ldots \leq h(x_n)$ and

$$\mu_i^j = \mu(\{x_i, x_{i+1}, \ldots, x_j\}), \quad \text{for } i \leq j. \tag{5}$$

A conditional fuzzy measure on Y given X is a fuzzy measure $\nu(\cdot \,|\, x)$ on Y for any given $x \in X$. For $E \subset Y$, the ν-induced fuzzy measure is computed by

$$\mu_Y(E) = \int_X \nu_Y(E|x)d\mu(x). \tag{6}$$

With the above tools, a fuzzy hidden Markov model can be parameterized by $\bar{\lambda} = [\bar{\pi}, \bar{A}, \bar{B}]$ where $\bar{\pi}$ is the initial fuzzy density of the state; $\bar{A}$ is the state transition matrix parameterized by fuzzy densities; $\bar{B}$ is measurement matrix parameterized by fuzzy densities. Note that the fuzzy state transition matrix is no longer time varying. This simplifies the learning of model parameters significantly. On the other hand, FHMM preserves the non-stationary nature of plume propagation and the nonstationary behavior is achieved naturally by the nonlinear aggregation of sensor measurements using fuzzy integral [20].

2.3.3 Viterbi algorithm for most likely sequence estimation

For an HMM with parameter Λ, finding the most likely state sequence $\hat{\mathbf{x}}_{1:K}$ given the observation $\mathbf{y}_{1:K}$ is often called maximum likelihood (ML) decoding. Viterbi algorithm guarantees obtaining the ML sequence with the following procedure [25].

$$\delta_1(i) = \pi(i) b_{ik_1},$$

$$\delta_t(i) = \max_j [\delta_{i-1}(j) a_{ji}(t) b_{ik_t}],$$

$$\phi_t(i) = \arg\max_j [\delta_{t-1}(j) a_{ji}(t)],$$

$$i_K = \arg\max_i \delta_K(i),\ i_t = \phi_{t+1}(i_{t+1}),\ t = K-1, ..., 1.$$

For a fuzzy hidden Markov model, the most likely state sequence given the observation sequence can also be defined with properly chosen fuzzy measure and fuzzy integral. The resulting optimization problem can be written as

$$\hat{\mathbf{x}}_{1:K} = \arg \max_{\mathbf{x}_{1:K} \in \{0,1\}^{NK}} \bar{p}(\mathbf{y}_{1:K} | \mathbf{x}_{1:K}) \tag{7}$$

where $\bar{p}(\cdot)$ is the extension of likelihood function in (13) with the conditional fuzzy measure. Note that the fuzzy likelihood function can be decomposed as follows.

$$\bar{p}(\mathbf{y}_{1:K} | \mathbf{x}_{1:K}) = \bar{\pi} \prod_{i=1}^{K} \bar{p}(\mathbf{x}_i | \mathbf{x}_{i-1}) \bar{p}(\mathbf{y}_i | \mathbf{x}_i) \tag{8}$$

Thus the decoding algorithm of Viterbi type can also be applied to FHMM. Specifically, the fuzzy Viterbi decoding procedure is as follows.

$$\bar{\delta}_1(i) = \bar{\pi}(i) \bar{b}_{ik_1}, \tag{9}$$

$$\bar{\delta}_t(i) = \max_j [\bar{\delta}_{i-1}(j) \bar{a}_{ji} \rho_{ji}(t) \bar{b}_{ik_t}], \tag{10}$$

$$\bar{\phi}_t(i) = \arg\max_j [\bar{\delta}_{t-1}(j) \bar{a}_{ji} \rho_{ji}(t)], \tag{11}$$

where

$$\rho_{ji}(t) = \frac{\mu_t^K(j,i) - \mu_{t+1}^K(j,i)}{\sum_j \bar{a}_{ji} (\mu_{t-1}^K(j,i) - \mu_t^K(j,i))} \tag{12}$$

if Choquet integral is used with respect to a fuzzy measure μ [20]. It is a time varying and nonlinear function of the fuzzy state transition parameter $\bar{a}_{ji}$, which characterizes the nonstationary nature of plume propagation using only time invariant parameter set $\bar{\Lambda}$. The resulting state sequence estimate is still given by

$$i_K = \arg\max_i \bar{\delta}_K(i),\ i_t = \bar{\phi}_{t+1}(i_{t+1}),\ t = K-1, ..., 1.$$

Note that the most likely sequence estimation algorithm of Viterbi type guarantees finding the optimal solution and it has the worst case complexity of $O(K2^N)$. Clearly, Viterbi algorithm is much more efficient than the exhaustive search method for general decoding problem given by (13) or its fuzzy extension (7), which has the complexity of $O(2^{NK})$.

2.3.4 Greedy heuristic sequence estimation algorithm

Viterbi algorithm (VA) has a complexity linear in the length of observation sequence but exponential in the number of cells. Throughout the past three decades, many attempts have been made to reduce the complexity of VA by searching only a selected number of paths in obtaining $\phi_t(i)$ (or $\bar{\phi}_t(i)$) with various criteria. However, unlike the original VA, there is no guarantee that the best state sequence obtained by any of those algorithms is indeed the optimal one. Recently, [19] proved the existence of efficient and exact maximum likelihood decoding method with the complexity polynomial in N under high SNR regime. Unfortunately, the decoding error probability goes to zero only when the SNR goes to infinity. In plume localization problem, the high SNR assumption is usually valid for the m-th bit of the state variable when a sensor n is in cell m measuring its chemical concentration intensity. Thus we propose a greedy heuristic decoding algorithm applicable to both HMM and FHMM following the general constructive approach proposed in [19].

The algorithm contains three steps.

1. Obtain a feasible solution $\hat{\mathbf{x}}_{1:K}$ satisfying $\hat{x}(m)_k = y(n)_k$, $\forall k$, n where sensor n in cell m has a plume detection.
2. Test the optimality: If the solution satisfies

$$-\log \bar{p}(\mathbf{y}_{1:K}|\hat{\mathbf{x}}_{1:K}) < L_0, \tag{13}$$

   then declare that $\hat{\mathbf{x}}_{1:K}$ is the most likely sequence and stop.
3. If the optimality test fails, then search the subset of the VA paths with Hamming distance no greater than L from $\hat{\mathbf{x}}_{1:K}$.

The first step is crucial and may save significant amount of computational time if the solution is near optimal. It has been suggested in [19] to use the decision feedback method for obtaining a candidate solution. Its complexity scales in $O(KN^2)$. Intuitively, in the high SNR regime, we can assume that the false alarm probability of each sensor is very small, therefore, the plume map at a later time can be directly used to estimate the plume map at an earlier time where few sensor detections are made. Our decision feedback algorithm is similar to that of [19] but runs reversely in time as follows.

$$\hat{\mathbf{x}}_K = \arg\max_{\mathbf{x}_K} \bar{p}(\mathbf{y}_K|\mathbf{x}_K),$$

$$\hat{\mathbf{x}}_t = \arg\max_{\mathbf{x}_t} \bar{p}(\mathbf{y}_t|\mathbf{x}_t, \hat{\mathbf{x}}_{t+1:K}), \quad t = K-1, ..., 1.$$

The threshold L_0 used in step 2 depends on the presumed plume path and SNR to be determined after having sensor detections. Note that if $P_{FA} \to 0$ and $P_D \to 1$, then the candidate solution will pass the test with high probability for arbitrarily chosen L_0. The search constraint L used in step 3 is chosen to be compatible with $(N - M)$ for large K.

2.3.5 Performance analysis

The proposed greedy decoding algorithm has the worst case complexity of $O(K2^L)$. If the SNR is large enough, then the average complexity of the algorithm is $O(KN^2)$ [19]. Next we show that the accuracy of our greedy heuristic algorithm has no essential loss compared with the optimal decoding algorithm, i.e., maximum likelihood decoder of Viterbi type, in the high SNR regime.

Theorem: Assume that the plume localization error $P_e \to 0$ as $K \to \infty$. There exist L_0 and L such that the greedy heuristic algorithm yields the error no larger than $O(P_e)$ for large enough K.

Proof sketch: For any state sequence $\mathbf{x}_{1:K}$ and the corresponding observation $\mathbf{y}_{1:K}$, define

$$d_{\min} = \min_{\mathbf{x}_{1:K}} \min_{\mathbf{e}_{1:K} \neq \mathbf{x}_{1:K}} \log \frac{\bar{p}(\mathbf{y}_{1:K}|\mathbf{x}_{1:K})}{\bar{p}(\mathbf{y}_{1:K}|\mathbf{e}_{1:K})} \tag{14}$$

It has been shown in [19] that $P(d_{\min} > 0) = 1$ when the SNR is large enough. If we choose

$$L_0 \leq d_{\min} - \max_{\mathbf{x}_{1:K}} \max_{\mathbf{e}_{1:K} \neq \mathbf{x}_{1:K}} \log \bar{p}(\mathbf{y}_{1:K}|\mathbf{e}_{1:K}) \tag{15}$$

then the test (13) is optimal in the sense that it only allows the most likely sequence estimate to pass asymptotically. Note that the test can be nontrivial if one chooses $L_0 = d_{\min}$ due to the fact that any mismatched fuzzy likelihood function should satisfy $\log \bar{p}(\mathbf{y}_{1:K}|\mathbf{e}_{1:K}) < -d_{\min}$ with probability one as $P_e \to 0$.

The actual decoding error depends on the model parameter $\bar{\Lambda}$. By invoking Fano's inequality [6], we have $P_e \geq O\left(\frac{H(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})}{NK}\right)$ for any algorithm asymptotically. Under high SNR assumption, the entropy rate of the observation sequence satisfies

$$h_{\mathbf{y}} = \lim_{K \to \infty} H(\mathbf{y}_{1:K})/K > 0 \tag{16}$$

When K is large, the feasible solution $\hat{\mathbf{x}}_{1:K}$ will satisfy the following condition: $\forall m$ containing a sensor, $\hat{x}(m)_k = x(m)_k$ with probability one. If we choose $L = N - M$, then the best solution $\hat{\mathbf{x}}^{\text{opt}}_{1:K}$ within the subset of the VA paths with Hamming distance no greater than L from $\hat{\mathbf{x}}_{1:K}$ has an error probability

$$P(\hat{\mathbf{x}}^{\text{opt}}_{1:K} \neq \mathbf{x}_{1:K}) \leq \sum_{k=1}^{K} \sum_{j \neq m} P(\hat{x}(j)_k \neq x(j)_k|\mathbf{y}_{1:K}) \tag{17}$$

Since $P(\hat{x}(j)_k \neq x(j)_k)$ decays exponentially with rate $h_{\mathbf{y}}$ [19], we have

$$\begin{aligned} P(\hat{\mathbf{x}}^{\text{opt}}_{1:K} \neq \mathbf{x}_{1:K}) &\leq O\left(\frac{H(\mathbf{x}_{1:K}|\mathbf{y}_{1:K}, \hat{\mathbf{x}}^{\text{opt}}_{1:K})}{NK}\right) \\ &\leq O\left(\frac{H(\mathbf{x}_{1:K}|\mathbf{y}_{1:K})}{NK}\right) \end{aligned}$$

for large K where the second inequality follows by the fact that conditioning reduces the entropy. In practice, the conditional entropy $H(\mathbf{x}_{1:K} \mid \mathbf{y}_{1:K})$ has to be estimated using the posterior distribution π_n. For the decoding problem over an HMM, π_n is obtained recursively using Bayes rule.

$$\pi_n = \frac{B(\mathbf{y}_n)A(n)'\pi_{n-1}}{\sum_{l=1}^{2^M} B(\mathbf{y}_n = l)A(n)'\pi_{n-1}}, \ \pi_0 = \pi. \tag{18}$$

For the decoding problem over an FHMM, the above equation should be replaced by the fuzzy intersection in the numerator and fuzzy integral in the denominator. In both cases, the resulting posterior distribution is helpful to design K for the desired decoding accuracy.

2.4 Simulation study

We simulate a plume source as independent particles following random walks which satisfy the advection and diffusion constraints. The model is reasonably accurate and the plume path generation is usually much faster than solving the advection-dispersion equation directly [18]. As an illustration, assuming Δt = 1s, the plume source is at (0, 0) with release rate 100 particles per second and duration of 8s. There are 20 sensors randomly deployed in a 1000×1000m^2 field with sensing range of 50m for each sensor and at least 10 particles in its sensing area for a plume detection at any time. With wind velocity given by $(v_x(k), v_y(k))$ = (8, 5)m/s, longitudinal and transversal dispersivity α_L = 0.8, α_T = 0.2 and diffusion coefficient D = 0.9, one realization of the Gaussian plume at 100s is shown in Fig. 1 with two sensor detections.

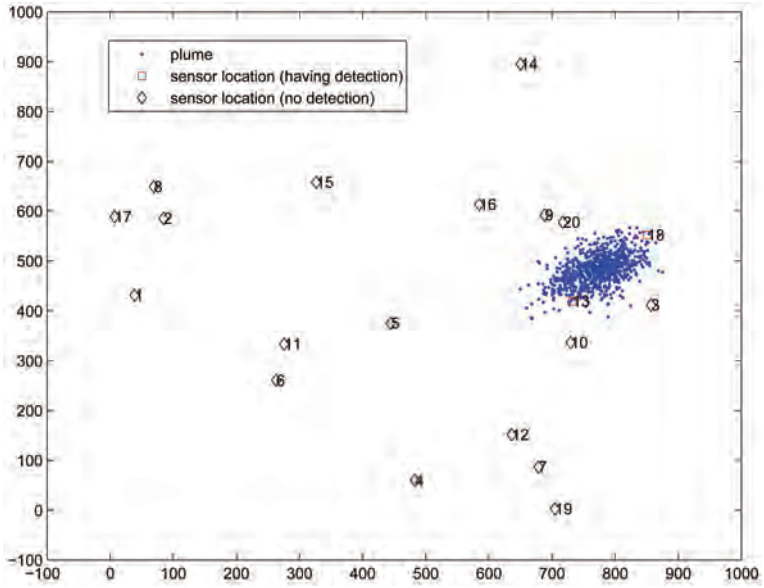


Fig. 1. One realization of plume propagation at K = 100.

We partition the region into 100 cells of the same square shape. It is assumed that initially there is no plume source in the sensing field. All 20 sensors are assumed to be synchronized and provide binary detections to a centralized data processor for plume mapping. The FHMM assumes p_b = 0.005 and p_c = 0.8. The plume source always starts at the bottom left cell at 4s with a constant release rate. Viterbi algorithm maintains all feasible solutions in its trellis graph while greedy heuristic algorithm only keeps the solutions within a Hamming distance of 8 to the initial candidate. We compare the probability of finding the correct cell and initial releasing time of the plume source after K time steps. The plume localization error probabilities are shown in Fig. 2 based on 5000 Monte Carlo runs for each K. We can

see that greedy heuristic algorithm has the localization error close to that using Viterbi algorithm and the performance gap decreases as K increases. Using Matlab to compile both algorithms on a Pentium 4 PC with 2.80GHz CPU, we found that the average time to find the best state sequence using greedy heuristic algorithm is 0.05s when K = 100 while Viterbi algorithm takes 5.3s in average to obtain the most likely sequence estimate. Thus the proposed greedy algorithm achieves the plume localization accuracy close to that using Viterbi algorithm while the computational time is orders faster than that using Viterbi algorithm.

Our approach can also be used to estimate the total mass of plume release. However, there is no guarantee on its accuracy even for instantaneous release of a single plume due to the nature of binary sensor detection. To estimate the release rate sequence of a plume source, denser sensor coverage or more accurate plume concentration intensity measurement is needed. This problem will be addressed in the next section.

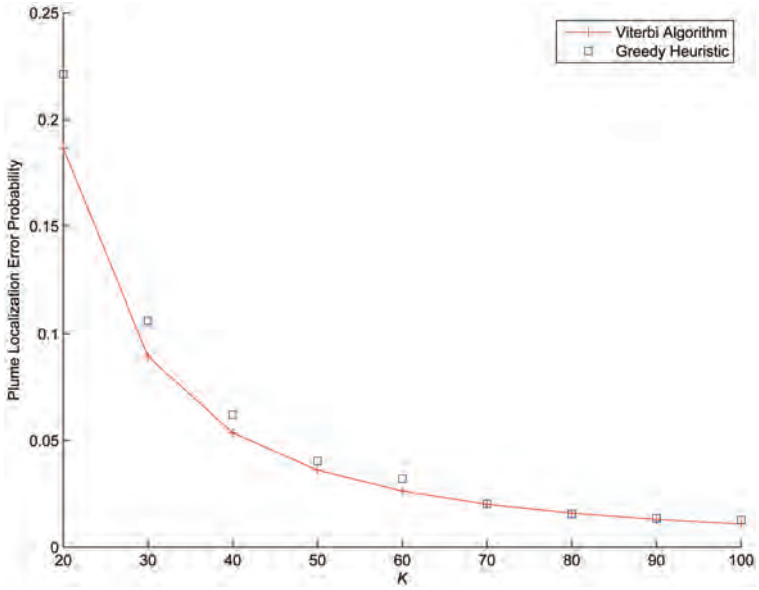


Fig. 2. Comparison of plume localization error probability with various observation length K.

3. Parameter estimation and model selection for gaussian plume sources

This section deals with joint plume localization and release sequence estimation when the number of plume sources is unknown. We start with the plume source aggregation and sensor measurement model in Section 3.1. Section 3.2 presents the regularized least squares solution to the parameter estimation problem. Section 3.3 discusses the implementation of the joint model selection (on the number of sources) and parameter estimation using greedy algorithm and the choice of regularization parameter. Section 3.4 compares our approach with alternative regularization methods. Section 3.5 provides realistic source release scenarios to assess the performance of the proposed algorithm.

3.1 Plume aggregation and sensor measurement model

We assume that the wind field in the search area can be accurately modeled and sensors can collect fairly accurate concentration readings in their neighboring areas. A Cartesian coordinate system is used with x-axis oriented towards the mean wind direction, y-axis in the cross-wind direction and z-axis in the vertical direction. If the source of a pollutant is located at (x_0, y_0, z_0) with release rate $q(t)$, then at time t, the concentration of the pollutant at some down-stream location $(x, y, 0)$ can be written as [16]

$$C(x, y, 0, t) = \int_0^t K(t, \tau) q(\tau) d\tau \tag{19}$$

where the kernel $K(t, \tau)$ is

$$K(t,\tau) = \frac{\exp\left[-\frac{[x-x_0-u(t-\tau)]^2}{4K_x(t-\tau)} - \frac{(y-y_0)^2}{4K_y(t-\tau)} - \frac{z_0^2}{4K_z(t-\tau)}\right]}{8\pi^{\frac{3}{2}}(K_x K_y K_z)^{\frac{1}{2}}(t-\tau)^{\frac{3}{2}}} \tag{20}$$

with mean wind speed u and diffusion coefficients K_x, K_y, K_z along x, y and z directions, respectively.

We assume that there are J sensors deployed at fixed locations where sensor j is located at $(x_j, y_j, 0)$ and collects N concentration readings $\mathbf{c}_j = \{C(x_j, y_j, 0, t_n)\}_{n=1}^{N}$. Denote by $\mathbf{c} = \{\mathbf{c}_j\}_{j=1}^{J}$ the collection of all sensor readings. Denote by $\mathbf{q} = \{q(\tau_i)\}_{i=1}^{M}$ the discretized source release sequence where $q(\tau_i)$ is the release rate at time τ_i. Ideally, we have the following observation equation

$$\mathbf{c} = A(\mathbf{p})\mathbf{q} \tag{21}$$

where $\mathbf{p} = (x_0, y_0, z_0)$ denotes the unknown source location. Note that for measurement $c_j(x_j, y_j, 0, t_n)$, the corresponding element $a_{(jn,k)}$ in $A(\mathbf{p})$ is given by

$$a_{(jn,k)} = K(t_n, \tau_k)\beta_{nk} \tag{22}$$

where β_{nk} is a quadrature weight [16, 17]. The estimation of source location $\mathbf{p}$ and release rate $\mathbf{q}$ can be formulated as the least squares problem given by

$$(\mathbf{p}^*, \mathbf{q}^*) = \arg\min_{\mathbf{p},\mathbf{q}} ||\mathbf{c} - A(\mathbf{p})\mathbf{q}||_{l_2}^2. \tag{23}$$

Note that this formulation is valid only for a single source.

To extend the estimation problem to include multiple sources, we assume that the concentration readings are the results of aggregation from multiple source releases. Assume that there are s sources with unknown locations $\{\mathbf{p}(i)\}_{i=1}^{s}$ and release rate sequences $\{\mathbf{q}(i)\}_{i=1}^{s}$. Then we have the following ideal observation equation

$$\mathbf{c} = \sum_{i=1}^{s} A(\mathbf{p}(i))\mathbf{q}(i). \tag{24}$$

The source parameter estimation problem becomes identifying the number of sources, the corresponding origins and the release sequences jointly using only the concentration readings from multiple sensors.

3.2 Regularized least squares

In the single source case, the matrix A in (23) can be analyzed for various source locations. By ranking the singular values of A, the authors of [16] found that the discrete time least squares problem (23) is in general ill-posed and suggested to use the Tikhonov regularization to ensure certain smoothness of $\mathbf{q}$. However, for a source with an instantaneous release, the sequence $\mathbf{q}$ may have only a single spike, which violates the smoothness assumption. Nevertheless, for multiple sources with instantaneous releases, we will observe the aggregated sparse signal with an unknown number of spikes. In fact, the sparsity assumption is crucial for the identification of multiple sources with instantaneous releases at different times. To encourage the sparsity of the release rate sequence estimate, we propose to use l_p-regularized least squares as the objective function, i.e.,

$$J(\mathbf{p},\mathbf{q}) = \|\mathbf{c} - A(\mathbf{p})\mathbf{q}\|_{\ell_2}^2 + \lambda\|\mathbf{q}\|_{\ell_p} \tag{25}$$

where the regularization parameters p controls the sparsity of the solution $\mathbf{q}$ and λ makes the tradeoff between the goodness-of-fit to the observations and the complexity of the model. Note that $p = 1$ is popularly used in compressed sensing [10] due to its numerical reliability. In fact, for any given $\mathbf{p}$, minimizing (25) becomes a convex program if one chooses $p = 1$. However, our l_1-regularized least squares problem still requires non-convex optimization without knowing the source location $\mathbf{p}$. In addition, when choosing the regularization term with $0 < p < 1$, one favors a more sparse solution than that using l_1-regularization [7]. This might be helpful when one has prior knowledge about the type of release of plume sources. In this case, the regularization term $\|\cdot\|_{\ell_p}^p$ is not a norm, but $d(x, y) = \|x - y\|_{\ell_p}^p$ is still a metric.

When the concentration readings are the aggregation of individual release, we have to identify the number of sources and find their locations. In this case, we are facing a model selection problem, where model s corresponds to s sources with unknown locations $\{\mathbf{p}(i)\}_{i=1}^s$ and release rate sequences $\{\mathbf{q}(i)\}_{i=1}^s$. Assuming that different sources have different instantaneous release times so that there is no identifiability issue among the models, we can choose model s that minimizes a modified version of (25)

$$J(s,\mathbf{p}(s),\mathbf{q}(s)) = \left\|\mathbf{c} - \sum_{i=1}^{s} A(\mathbf{p}(i))\mathbf{q}(i)\right\|_{\ell_2}^2 + \mathrm{pen}(\mathbf{p}(s),\mathbf{q}(s)) \tag{26}$$

where

$$\mathrm{pen}(\mathbf{p}(s),\mathbf{q}(s)) = \lambda\sum_{i=1}^{s}\|\mathbf{q}(i)\|_{\ell_p} + 3s\log N. \tag{27}$$

Note that the first term in the penalty encourages the sparsity of each identified source release sequence and the second term is for model complexity of the source location parameter based on the Bayesian information criterion [27]. The second term is necessary because one does not want to treat one source with two instantaneous releases (sparsity of 2) as two different sources with instantaneous releases at different time instances (sparsity of 1). In practice, when the locations of two sources are close, they could be identified as a

single source with aggregated release rate sequence. This seems to be acceptable when the locations of multiple sources are within the arrange of the localization accuracy obtained by minimizing (26).

3.3 Model selection and parameter estimation with a greedy algorithm

Finding the optimal solution to (26) requires solving a high dimensional nonlinear optimization problem for any fixed regularization parameters p and λ. In practice, the number of sources is usually small and a strong source can have the dominant effect on the sensor readings. Thus it would be meaningful to identify and localize one source at a time by treating the impact from the remaining possible sources as additive noise. In this case, assuming the source location is given, one can obtain the sparse solution of the release rate sequence by solving the following optimization problem.

$$\min_{\mathbf{q}} ||\mathbf{c} - A\mathbf{q}||_{\ell_2}^2 + \lambda ||\mathbf{q}||_{\ell_p}. \tag{28}$$

When $p = 1$, the problem becomes a convex program and is highly related to LASSO [29]. Once we obtain the release rate of the source, we can refine the estimate of source location by solving the regular nonlinear least squares problem given by

$$\min_{\mathbf{p}} ||\mathbf{c} - A(\mathbf{p})\mathbf{q}||_{\ell_2}^2. \tag{29}$$

Note that for the newly estimated source location, the sparsity (non-zero locations) of the solution to (28) may change. We can iteratively update the release rate and source location estimate until the residual is comparable to the noise level of sensor readings.

We can extend the above procedure to deal with unknown number of sources. We apply a greedy heuristic algorithm that iteratively refines the estimate of signal sparsity and the noise level to determine the appropriate regularization parameter. The algorithm is greedy in the sense of extracting one plume source at a time, from the strongest one to the weakest (based on the penalty term in the model selection criterion). It simultaneously determines the number of sources, the corresponding locations and release rate sequences by the following steps.

1. Set $\boldsymbol{s} = 1$.
2. **Initialization:** Set $\boldsymbol{k} = 0$, $\mathbf{q}(\boldsymbol{s})_k = \mathbf{0}$ with an initial guess of source location $\mathbf{p}(\boldsymbol{s})_k$.
3. **Refining the estimate:** Use Newton-Ralphson update

$$\mathbf{q}(s)_{k+1} = \mathbf{q}(s)_k + A(\mathbf{p}(s)_k)^T [\mathbf{c} - A(\mathbf{p}(s)_k)\mathbf{q}(s)_k] \tag{30}$$

   to refine the estimated source release sequence.
4. **Choosing regularization parameter:** Compute the median of the residual $|\mathbf{c} - A(\mathbf{p}(s)_k)\mathbf{q}(s)_{k+1}|$ and choose λ proportional to the estimated noise level.
5. **Denoising by soft thresholding:** Compute the sparse approximation of $\mathbf{q}(s)_{k+1}$ by $\mathbf{q}(s)^*_{k+1} = T(\mathbf{q}(s)_{k+1})$ where

$$T(x) = \text{sign}(x) \max\{|x| - \lambda, 0\}. \tag{31}$$

6. **Source localization:** Solve the nonlinear least squares problem

$$\mathbf{p}(s)_{k+1} = \arg\min_{\mathbf{p}(s)_k} ||\mathbf{c} - A(\mathbf{p}(s)_k)\mathbf{q}(s)^*_{k+1}||^2_{\ell_2}. \tag{32}$$

7. **Model selection:** Set $k = k+1$ and iterate until $\mathbf{q}(s)$ converges to a sparse solution $\mathbf{q}(s)^*$ or a predetermined maximum number of iterations $k_{\max}$ has reached. Subtract out the identified source from sensor readings.

$$\mathbf{c} \leftarrow \mathbf{c} - A(\mathbf{p}(s)^*)\mathbf{q}(s)^*, \quad s = s+1 \tag{33}$$

   Repeat steps 2-6 until

$$J(s-1, \mathbf{p}(s-1)^*, \mathbf{q}(s-1)^*) < J(s, \mathbf{p}(s)^*, \mathbf{q}(s)^*). \tag{34}$$

8. Declare the number of sources (s - 1), the corresponding locations $\mathbf{p}(s$ - 1)* and release rate sequences $\mathbf{q}(s$ - 1)*.

For any given λ and $\mathbf{p}(s)^*$, the above iterative procedure converges to the optimal solution of (28) for p = 1 [1]. We used the median estimator of the residue to obtain the noise level which is robust against outliers. It is less sensitive to possible model mismatch than using the mean of the residue when we initially assume that there is a single (strong) source which results in the concentration readings while treating other (weak) sources as noise. Note that the dimension of $\mathbf{p}(s)$ only depends on the model order, i.e., the number of sources, which is usually much lower than the dimension of release sequences $\mathbf{q}(s)$. Thus solving the nonlinear least squares problem (32) is less computationally demanding than solving (26) directly.

When $0 < p < 1$, (28) becomes non-convex program and any iterative procedure may be trapped at a local minimum. Another issue is that (24) may become underdetermined when A has rank deficiency. In such a case, the sparse solution to the following constrained optimization problem

$$\min_{\mathbf{q}} ||\mathbf{q}||_{\ell_p}, \ \text{s.t.} \ \mathbf{c} = A(\mathbf{p})\mathbf{q}$$

is still meaningful. To encourage more sparsity of the release rate sequence with smaller p and solve the above constrained optimization problem directly, we apply iterative reweighted least squares (IRLS) update [7] and replace the soft thresholding step by

$$\mathbf{q}(s)^{(n)}_{k+1} = W^{(n)} A(\mathbf{p}(s)_k)^T [A(\mathbf{p}(s)_k) W^{(n)} A(\mathbf{p}(s)_k)^T]^{-1} \mathbf{c} \tag{35}$$

where the weighting matrix $W^{(n)}$ is diagonal with entries

$$w_i^{(n)} = \left\{ \left[q(\tau_i)^{(n-1)}_{k+1} \right]^2 + \epsilon \right\}^{p/2-1}, i = 1, ..., M. \tag{36}$$

The damping coefficient ϵ is chosen to be relatively large initially and decreases to a very small number when the above iteration is close to converge. Note that the IRLS algorithm converges in less than 100 iterations most of the time in our simulation study. Even though there is no theoretical guarantee that the resulting solution is globally optimal, we suspect that it does approach to near global minimum since the solution quality improves when using smaller p.

The above greedy algorithm can be interpreted as performing basis pursuit [13]. Specifically, given the stacked observation **c**, we want to find a good n term approximation using different sources as the basis functions from a general dictionary D. Denote by $\{\mathbf{g}_i\}$ the i-th basis being selected. Basis pursuit proceeds as follows.

1. Initialization:
   - Approximation: $\mathbf{s}_0 = 0$
   - Residual: $\mathbf{r}_0 = \mathbf{c}$
   - Basis collection: $\Gamma_0 = \phi$
2. Pure greedy search:

$$\mathbf{g}_{n+1} = \arg\max_{\mathbf{g}\in D} | < \mathbf{r}_n, \mathbf{g} > |, \Gamma_{n+1} = \Gamma_n \cup \{\mathbf{g}_{n+1}\}$$

$$\mathbf{s}_{n+1} = \mathbf{s}_n + < \mathbf{r}_n, \mathbf{g}_{n+1} > \mathbf{g}_{n+1}$$

$$\mathbf{r}_{n+1} = \mathbf{c} - \mathbf{s}_{n+1}$$

Unfortunately, identifying a basis in the greedy pursuit is equivalent to localizing the origin of a single source, which requires solving a nonlinear least squares problem. The regularization on release rate sequence and penalty on the number of unknown sources prevent the resulting optimization problem from being ill-posed. Note that when the basis functions in the dictionary satisfy certain mutual incoherence property, the greedy basis pursuit algorithm guarantees finding the best n term approximation [13].

3.4 Comparison with other regularization techniques

Tikhonov's regularization has been proposed in [16, 17] which essentially uses the objective function

$$J(s, \mathbf{p}(s), \mathbf{q}(s)) = \left\| \mathbf{c} - \sum_{i=1}^{s} A(\mathbf{p}(i))\mathbf{q}(i) \right\|_{\ell_2}^2 + \lambda \sum_{i=1}^{s} ||L\mathbf{q}(i)||_{\ell_2}^2 \tag{37}$$

where L controls the smoothness of $\mathbf{q}(i)$ with the approximate form

$$||L\mathbf{q}(i)||_{\ell_2}^2 \approx \int_0^{t_n} \left(\frac{d^N \mathbf{q}(i)}{dt^N} \right)^2 d\tau. \tag{38}$$

The popular choice for obtaining a smooth solution is $N = 2$. Unfortunately, the above regularization technique only works for continuous releases from well separated sources. We rely on the sparsity of $\mathbf{q}(s)$ to identify the model order s, which is suitable for localizing multiple sources of instantaneous release type.

Another sparsity enforced estimator was proposed in [4] which essentially minimizes the following objective function

$$J = \left|\left| A(\mathbf{p}(s))^T [\mathbf{c} - A(\mathbf{p}(s))\mathbf{q}(s)] \right|\right|_{\ell_\infty} + \lambda ||\mathbf{q}(s)||_{\ell_1}. \tag{39}$$

For known source locations, the estimated release rate guarantees to recover all possible sparse signals with a large probability [4]. However, the above objective function is a non-smooth function of $\mathbf{p}(s)$, which is difficult to optimize when both source locations and release rate sequences are unknown. In practice, we fix the source locations $\mathbf{p}(s)$ and solve

(39) via linear programming. Then we fix the release rate sequence $\mathbf{q}(s)$ and update $\mathbf{p}(s)$ in its gradient descent direction. The iteration continues until $\mathbf{p}(s)$ reaches a stationary solution and the sparsity of $\mathbf{q}(s)$ does not change.

3.5 Simulation of joint plume localization and release sequence estimation

We present the simulation study of source localization and release rate estimation using multiple sensors. We are interested in both model selection and source parameter estimation accuracy.

3.5.1 Scenario generation

Consider a single source located at (-40, 35, 12) with instantaneous release of $q(10) = 2 \cdot 10^5$. We assume that the wind speed $u = 1.8$ along x-axis and $K_x = K_y = 12$, $K_z = 0.2113$. Five sensors, located at (0, 0), (15, 15), (30, 30), (45, 45), (60, 60), respectively, collect concentration readings synchronously with 100 samples per sensor. All sensors are on the ground with zero elevation. We add Gaussian noise to the sensor readings with standard deviation $4 \cdot 10^{-3}$. Each sensor will have a plume detection when the concentration reading exceeds 0.01. Fig. 3 shows one realization of the concentration readings from the five sensors. We can see that sensor 1 has early detection while sensors 3-5 have relatively large peaks in the concentration readings.

We also considered the case of two sources where one source located at (-40, 35, 12) has the instantaneous release of $q(10)=2 \cdot 10^5$ and the other located at (-30, 15, 15) has the instantaneous release of $q(50)=1 \cdot 10^5$. Fig. 4 shows one realization of the concentration readings from the five sensors. Compared with Fig. 3, we can barely see the effect of the second source release due to the detection delay and source aggregation.

3.5.2 Model selection and parameter estimation accuracy

We want to compare our l_p-regularization method with Tikhonov's method [16, 17] (denoted by $p = 2$) and Dantzig selector [4] (denoted by $p = \infty$) for both one-source and two-source cases. Note that Tikhonov's method is not appropriate for estimating instantaneous release rate, which is non-smooth. However, it is meaningful to study how the incorrect assumption in regularization may affect model selection accuracy. We estimated the probability of identifying the correct number of sources based on 100 realizations of each case. For those instances where the number of sources is correctly identified, we also computed the root mean square (RMS) error of the location estimate for each source. In the case of $s = 2$, the RMS error of the second source is in parentheses. The results are listed in Table 1. We can see that in the single source case, our l_p-regularization method can identify the correct number of sources almost perfectly. In the two-source case, Tikhonov's method failed to identify the second source most of the time and Dantzig selector can only identify the correct number of sources with 64 out of 100 cases. Surprisingly, the proposed l_p-regularization method is able to find the correct model order with higher than 80% probability. As we reduce p, there is a slight increase in the probability of obtaining the correct number of sources due to the strong enforcement of sparsity. Among all cases where the first source is correctly identified, the root mean square error of the estimated release rate is $4.6 \cdot 10^4$ with $p = 1$. Note that the root mean square error of estimated location of the first source increases when we have a second source aggregated to it. Note also that the algorithm assuming the correct model order can only achieve the root mean square error of estimated location of the second source around 18 using l_p-regularized method with $p = 1$.

These observations suggest that the l_p-regularized least squares method is effective in joint model selection and parameter estimation for instantaneous source release.

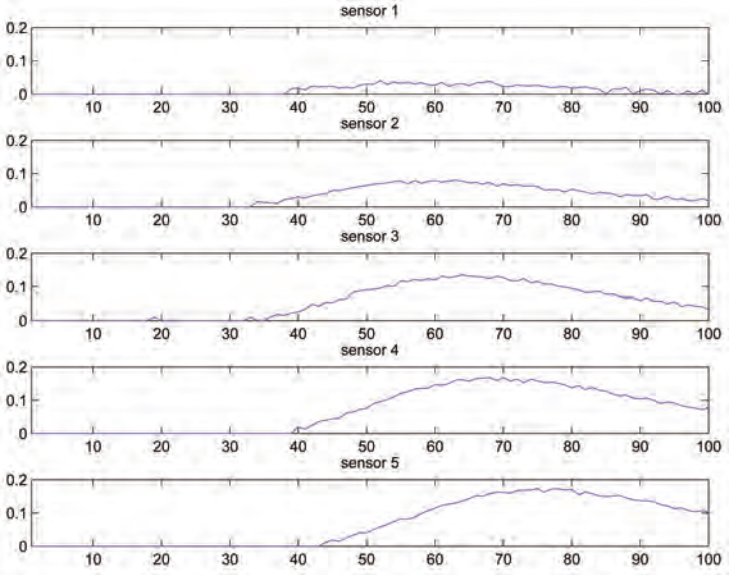


Fig. 3. Sensor readings for a single source with instantaneous release.

p	0.1	1	2	∞
$P(s{=}1)$	1.0	1.0	0.92	0.99
RMS error	10.8	10.6	10.2	11.1
$P(s{=}2)$	0.88	0.85	0.13	0.64
RMS error	14.9(21.2)	13.8(20.6)	12.0(18.4)	13.7(19.5)

Table 1. Comparison of Model Selection and Source Localization Accuracy with Different Regularization Methods

3.5.3 Model mismatch to continuous release source

Consider a single source located at (-40, 35, 12) with continuous release rate

$$q(t) = [e^{-0.1(t-10)}u(t-10) + 2e^{-0.5(t-50)}u(t-50)] \cdot 10^5.$$

One realization of the concentration readings from the five sensors is shown in Fig. 5. Note that the concentration readings from sensors 2-5 have not reached their peaks by the end of the samples. This will in general make the source parameter estimation more difficult. In 100 realizations, the l_p-regularized least squars method with p = 1 identified one source in 92 times and two sources in 8 times with their estimated locations close to each other. The incorrect identification of model order is due to the abrupt release at two time instances t = 10 and t = 50 with exponential decay of the release rate. The root mean square error of the estimated source location is 15.4 using the estimates from the correctly identified cases. Clearly, the l_p-regularized least squares method can tolerate slight model mismatch when the release rate sequence is not overly sparse.

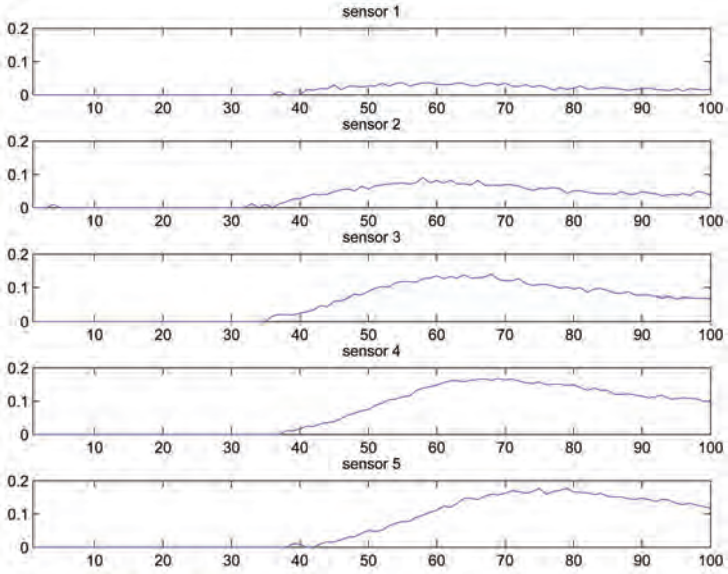


Fig. 4. Sensor readings for two sources, each with instantaneous release.

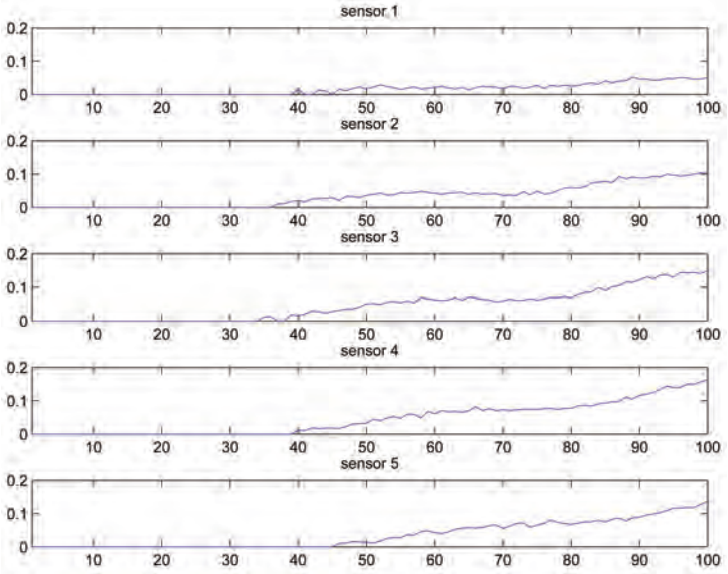


Fig. 5. Sensor readings for one source with continuous release.

4. Discussion and conclusions

In this chapter, we studied plume detection, localization and tracking problem with two different settings. For plume mapping with binary detection sensors, we formulated the problem as finding the most likely state sequence based on a fuzzy hidden Markov model.

Under the assumption that each sensor has high detection and low false alarm probability, we proposed a greedy heuristic decoding algorithm with much less computational cost than the well known Viterbi algorithm. The plume localization accuracy of our algorithm is close to the optimal decoder using Viterbi algorithm when tracking a single plume using randomly deployed sensors. Our algorithm is applicable to general decoding problem over a long observation sequence when the localization error probability of the Viterbi decoder is small.

There is a serious drawback of using FHMM for plume tracing. In our FHMM formulation, one can not distinguish whether a plume existence state is due to source releasing or plume propagation without knowing the whole state sequence. Thus one has to make tradeoff between the delay and localization accuracy. A refined plume propagation model based on more accurate sensor readings and contaminant transport physics was then used for source localization and release rate sequence estimation. When localizing unknown number of sources based on the observation of aggregated concentrations, we proposed an l_p-regularized least squares method to estimate the location and release rate of atmospheric pollution. For $0 \leq p \leq 1$, the method enforces sparsity of the release sequence of each identified source. The proposed greedy method can identify multiple sources of instantaneous release type and can also localize sources of continuous release. The accuracy of source parameter estimation has been examined for the cases where the number of sources and the corresponding locations are unknown.

In general, the least squares approach does not provide any measure of the estimation error. However, one can examine the residual and make additional assumptions such as additive Gaussian noise in order to quantify the covariance of the localization error. Through simulation study, we found that the proposed method is effective in localizing instantaneous release sources and has certain degree of tolerance to model mismatch. It is worth noting that the sensor locations, sampling rate and measurement accuracy can affect the source localization performance significantly. Finding the best sensor placement and sensing strategy in a given surveillance area is another important research theme and demands future work. We hope that with the advances in the development of greedy algorithms, many other challenging optimization tasks can be tackled with efficient and near optimal solutions.

5. References

[1] S. Boyd, and L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.

[2] S. M. Brennan, A. M. Mielke, and D. C. Torney, "Radiation Detection with Distributed Sensor Networks," *IEEE Computer*, pp. 57-59, August 2004.

[3] S. M. Brennan, A. M. Mielke, and D. C. Torney, "Radioactive Source Detection by Sensor Networks," *IEEE Nuclear Science*, 52(3), pp. 813-819, 2005.

[4] E. J. Candes and T. Tao,"The Dantzig Selector: Statistical Estimation When p Is Much Larger Than n," submitted to *Annals of Statistics*, 2005.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, first edition, MIT Press and McGraw-Hill, 1990.

[6] T. M. Cover, and J. A. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.

[7] R. Chartrand, "Exact Reconstruction of Sparse Signals via Nonconvex Minimization," *IEEE Signal Processing Lett.*, 14, pp. 707-710, 2007.

[8] J.-C. Chin, L.-H. Hou, J.-C. Hou, C. Ma, N.S. Rao, M. Saxena, M. Shankar, Y. Yong, and D.K.Y. Yau,"A Sensor-Cyber Network Testbed for Plume Detection, Identification, and Tracking," *6th International Symposium on Information Processing in Sensor Networks*, pp. 541-542, 2007.

[9] R. A. Dobbins, Atmospheric Motion and Air Pollution: An Introduction for Students of Engineering and Science, John Wiley & Sons, 1979.
[10] D. L. Donoho, "Compressed Sensing," *IEEE Trans. Information Theory*, 52, pp. 1289-1306, 2006.
[11] J. A. Farrell, S. Pang, and W. Li, "Plume Mapping via Hidden Markov Methods," *IEEE Trans. SMC-B*, 33(6), pp. 850-863, 2003.
[12] E. B. Fox, J. W. Fisher, and A. S. Willsky, "Detection and Localization of Material Releases with Sparse Sensor Configurations," *IEEE Trans. on Signal Processing*, 55(5), pp. 1886-1898, May 2007.
[13] P. S. Huggins, S. W. Zucker, "Greedy Basis Pursuit,", *IEEE Trans. Signal Processing*, 55(7), pp. 3760-3772, July 2007.
[14] H. Ishida, T. Nakamoto, T. Moriizumi, T. Kikas, and J. Janata, "Plume-Tracking Robots: A New Application of Chemical Sensors,"*Biological Bulletin*, 200, pp. 222-226, 2001.
[15] H. Ishida, G. Nakayama, T. Nakamoto, and T. Moriizumi, " Controlling A Gas/Odor Plume-Tracking Robot based on Transient Responses of Gas Sensors," *IEEE Sensors Journal*, 5(3), pp. 537-545, 2005.
[16] P. Kathirgamanathan, R. McKibbin and R. I. McLachlan, "Source Release Rate Estimation of Atmospheric Pollution from Non-Steady Point Source - Part 1: Source at A Known Location," *Res. Lett. Inf. Math. Sci.*, 5, pp. 71-84, 2003.
[17] P. Kathirgamanathan, R. McKibbin and R. I. McLachlan, "Source Release Rate Estimation of Atmospheric Pollution from Non-Steady Point Source - Part 2: Source at An Unknown Location," *Res. Lett. Inf. Math. Sci.*, 5, pp. 85-118, 2003.
[18] C. Kennedy, H. Ericsson, and P. L. R. Wong, "Gaussian Plume Modeling of Contaminant Transport," *Stoch. Environ. Res. Risk Assess*, 20, pp. 119-125, 2005.
[19] J. Luo, "Low Complexity Maximum Likelihood Sequence Detection under High SNR," submitted to *IEEE Trans. Information Theory*, Sept. 2006.
[20] M. A. Mohamed, and P. Gader, "Generalized Hidden Markov Models - Part I: Theoretical Frameworks", *IEEE Trans. on Fuzzy Systems*, 8(1), pp. 67-81, 2000.
[21] M. A. Mohamed, and P. Gader, "Generalized Hidden Markov Models - Part II: Application to Handwritten Word Recognition", *IEEE Trans. on Fuzzy Systems*, 8(1), pp. 82-94, 2000.
[22] A. Nehorai, B. Porat, and E. Paldi, "Detection and Localization of Vapor-Emitting Sources,"*IEEE Transactions on Signal Processing*, 43(1), pp. 243-253, 1995.
[23] G. Nofsinger, and G. Cybenko, "Distributed Chemical Plume Process Detection," *IEEE MILCOM*, Atlantic City, NJ, USA, 2005.
[24] M. Ortner, and A. Nehorai, "A Sequential Detector for Biochemical Release in Realistic Environments," *IEEE Transactions on Signal Processing*, 55(8), pp. 4173-4182, 2007.
[25] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. of the IEEE*, 77(2), pp. 257-286, 1989.
[26] N. Rao, "Identification of Simple Product-Form Plumes Using Networks of Sensors With Random Errors," *Proc. Int. Conf. on Information Fusion*, Florence, Italy, July 2006.
[27] G. Schwartz, "Estimating the Dimension of a Model," *Annals of Statistics*, vol.6, pp. 461-464, 1978.
[28] J. N. Seinfeld and S. N. Pandis, *Atmospheric Chemistry and Physics: From Air Pollution to Climate Change*, John Wiley & Sons, New Jersey, 1997.
[29] R. Tibshirani, "Regression Shrinkage and Selection via the LASSO," *Journal Royal Statistical Society B*, 58, pp. 267-288, 1996.
[30] T. Zhao and A. Nehorai, "Detecting and Estimating Biochemical Dispersion of A Moving Source in A Semi-Infinite Medium," *IEEE Transactions on Signal Processing*, 54(6), pp. 2213-2225, 2006.
[31] T. Zhao and A. Nehorai, "Distributed Sequential Bayesian Estimation of a Diffusive Source in Wireless Sensor Networks,", *IEEE Transactions on Signal Processing*, 55(4), pp. 1511-1524, 2007.

18

Greedy Type Bases in Banach Spaces[1]

Witold Bednorz
Department of Mathematics, Warsaw Univeristy
Poland

1. Introduction

Let $(X, \|\cdot\|)$ be a (real) Banach space. We refer to [38] or [28] as some introduction to the general theory of Banach spaces. Note that, as usual in the case, all the results we discuss here remain valid for complex scalars with possibly different constants. Let I be a countable set with possibly some ordering we refer to whenever considering convergence with respect to elements of I (wich will be denoted by $\lim_{i\to\infty}$).

Definition 1 *We say that countable system of vectors* $\Phi = (e_i, e_i^*)_{i\in I}$ *is biorthogonal if for* $i, j \in I$ *we have*

$$e_i^*(e_j) = \begin{cases} 1 & if \quad i = j \\ 0 & if \quad i \neq j \end{cases} . \tag{1}$$

Such a general class of systems would be inconvenient to work with, therefore we require biorthogonal systems to be aligned with the Banach space X we want to describe.

Definition 2 *We say that system* $\Phi = (e_i, e_i^*)_{i\in I}$ *is natural if the following conditions are satisfied:*

$$0 < \inf_{i\in I} \|e_i\| \leqslant \sup_{i\in I} \|e_i\| < \infty; \tag{2}$$

$$0 < \inf_{i\in I} \|e_i^*\| \leqslant \sup_{i\in I} \|x_i^*\| < \infty; \tag{3}$$

$$\overline{\operatorname{span}\{e_i : i \in I\}} = X. \tag{4}$$

Usually we assume also that $\|e_i\| = 1$ for all $i \in I$, i.e. we normalize the system. Note that if (4) holds then functionals $(e_i^*)_{i\in I}$ are uniquely determined by the set $\{e_i : i \in I\}$ and thus slightly abusing the convention we can speak about $(e_i)_{i\in I}$ being a biorthogonal system. Observe that if assumptions (1)-(4) are verified, then each $x \in X$ is uniquely determined by the values $(e_i^*(x))_{i\in I}$ and moreover $\lim_{i\to\infty} e_i^*(x) = 0$ for every $x \in X$.

Clearly the concept of biorthogonal system is to express each $x \in X$ as the series $\sum_{i\in I} e_i^*(x)e_i$ convergent to x. If such expansion exists for all $x \in X$ then we work in in the usual Schauder basis setting.

[1] Research is partially supported by the Foundation for Polish Science: Grant NP-37

Definition 3 *A natural system* Φ *is said to be Schauder basis* if $I = \mathbb{N}$ *and for any* $x \in X$ *the series* $\sum_{i=1}^{\infty} e_i^*(x)e_i$ *is convergent.*

However in this chapter we proceed in a slightly more general environment and do not require neither convergence of $\sum_{i\in 1} e_i^*(x)e_i$ nor fix a particular order on *I*. Obviously still the idea is to approximate any $x \in X$ by linear combinations of basis elements and therefore for any $x \in X$ and $J \subset I$ we define

$$P_J(x) := \sum_{j\in J} e_j^*(x)e_j, \tag{5}$$

whenever this makes sense. In particular it is well defined for any finite *J*. It suggests that for each $m = 0, 1, 2, \ldots$ we can consider the space of *m*-term approximations. Namely we denote by Σ_m the collection of all elements of *X* which can be expressed as linear combinations of *m* elements of $(e_i)_{i\in I}$, i.e.:

$$\Sigma_m := \{y = \sum_{j\in J} a_j e_j : \ J \subset I, \ |J| = m, \ a_j \in \mathbb{R}\},$$

Let us observe that the space Σ_m is not linear since the sum of two elements from Σ_m is generally in Σ_{2m} not in Σ_m. For $x \in X$ and for $m = 0, 1, 2, \ldots$ we define its best *m*-term approximation error (with respect to Φ)

$$\sigma_m(\Phi, x) = \sigma_m(x) = \inf\{\|x - y\| : \ y \in \Sigma_m\}$$

Commonly the system Φ is clear from the context and hence we can suppress it form the above notation. Observe that from (4) we acknowledge that for each $x \in X$ we have $\lim_{m\to\infty} \sigma_m(x) = 0$. There is a natural question one may ask, what has to be assumed for the best *m*-term approximation to exist, i.e. that there exists some $y \in \Sigma_m$ such that $\sigma_m(x) = \|x - y\|$. The question of existence of the best *m*-term approximation for a given natural system was discussed even in a more general setting in [4]. A detailed study in our context can be found in [39] from which we quote the following result:

Theorem 1 *Let* $(e_i, e_i^*)_{i\in I}$ *be a natural biorthogonal system in X. Assume that there exists a subspace* $Y \subset X^*$ *such that*

1. *Y is norming i.e. for all* $x \in X$

$$\sup\{|y(x)| : \ y \in Y \text{ and } \|y\| \leqslant 1\} = \|x\|.$$

2. *for every* $y \in Y$ *we have lim* $_{i\to\infty}$ $y(e_i) = 0$.

Then for each $x \in X$ *and m = 0, 1, 2, … there exists* $y \in \Sigma_m$ *such that* $\sigma_m(x) = \|x - y\|$.

The obvious candidate for being the norming subspace of *X** is $Y = \operatorname{span}(e_i^*, \ i \in I)$. Later we will show that this is the case of unconditional bases.

The idea of an approximation algorithm is that we construct a sequence of maps $T_m : X \to X$, $m = 0, 1, 2, \ldots$ such that for each $x \in X$, we have that $T_m(x) \in \Sigma_m$. The fundamental property which any admissible algorithm $(T_m)_{m\geq 0}$ should verify is that the error we make is comparable with the approximation error, namely

$$\|x - T_m(x)\| \leqslant C\sigma_m(x), \tag{6}$$

where C is an absolute constant. The potentially simplest approach is to use projection of the type (5). We will show later that in the unconditional setting for each m, $x \in X$ there exists projection P_J which has the minimal approximation error, namely $\|x - P_J x\| = \sigma_m(x)$. Among all the possible projections, one choice seems to be the most natural: we take a projection with the largest possible coefficients, that means we denote

$$\mathcal{G}_m(\Phi, x) := \mathcal{G}_m(x) = \sum_{j \in J} e_j^*(x) e_j$$

where the set $J \subset I$ is chosen in such a way that $|J| = m$ and $|e_j^*(x)| \geqslant |e_k^*(x)|$ whenever $j \in J$ and $k \notin J$. The collection of such $\mathcal{G}_m$, i.e. $(\mathcal{G}_m)_{m=0}^{\infty}$ will be called the Greedy Algorithm.

Clearly $\mathcal{G}_m$, $m = 0, 1, 2, \ldots$ have some surprising features which one should keep in mind, when working with this type of approximation (cf. [40]):

1. It may happen that for some x and m the element $\mathcal{G}_m(x)$ (i.e. the set J) is not uniquely determined by the previous conditions. In such case we pick any of them.
2. The operator $\mathcal{G}_m(x)$ is not linear (even if appropriate sets are uniquely defined).
3. The operator $\mathcal{G}_m(x)$ is discontinuous. To see it it suffices to fix $J_1, J_2 \subset I$ such that $J_1 \cap J_2 = \emptyset$ and $|J_1| = |J_2| = m$. We define two sequences of vectors

$$y_n = \frac{n+1}{n} \sum_{j \in J_1} e_j + \sum_{k \in J_2} e_k,$$

$$z_n = \sum_{j \in J_1} e_j + \frac{n+1}{n} \sum_{k \in J_2} e_k.$$

   Clearly both y_n and z_n converge to $\sum_{j \in J_1 \cup J_2} e_j$, but

$$\mathcal{G}_m(y_n) = \frac{n+1}{n} \sum_{j \in J_1} e_j \to \sum_{j \in J_1} e_j$$

   and

$$\mathcal{G}_m(z_n) = \frac{n+1}{n} \sum_{j \in J_2} e_j \to \sum_{j \in J_2} e_j.$$

4. Following the previous example we learn that $\mathcal{G}_m$ is continuous at the point $x \in X$ if and only if the set J used in the definition of $\mathcal{G}_m(x)$ is uniquely defined.
5. If $I = \mathbb{N}$ then there is a simple trick to define $\mathcal{G}_m$ uniquely, namely given $x \in X$ we define greedy ordering as the map $F : \mathbb{N} \to \mathbb{N}$ such that $\{j : e_j^*(x) \neq 0\} \subset F(\mathbb{N})$ and so that if $j < k$ then either $|e_{F(j)}^*(x)| > |e_{F(k)}^*(x)|$ or $|e_{F(j)}^*(x)| = |e_{F(k)}^*(x)|$ and $F(j) < F(k)$. With this notation the mth greedy approximation of x equals

$$\mathcal{G}_m(x) = \sum_{j=1}^{m} e^*_{F(j)} e_{F(j)}.$$

As announced we consider the greedy algorithm acceptable if it verifies (6). We formalize the idea in the following definitions:

Definition 4 *A natural biorthogonal system Φ is called a greedy basis if there exists a constant C such that for all $x \in X$* and *m* = 0, 1, 2, … *we have*

$$\|x - \mathcal{G}_m(\Phi, x)\| \leqslant C\sigma_m(\Phi, x).$$

The smallest constant *C* will be called the greedy constant of Φ.

Definition 5 *A natural biorthogonal system Φ is called quasi-greedy if for every $x \in X$ the norm limit $\lim_{m\to\infty} \mathcal{G}_m(\Phi, x)$ exists (and equals x).*

Clearly every greedy basis is quasi-greedy. We remark that those concepts were formall defined in [26] though implicit in earlier works of Temlyakov [30]-[33]. Throughout the chapter we study various properties of greedy and quasi greedy bases. Toward this goal let us introduce the following notation:

$$\varphi(m) := \sup\{\|\sum_{i\in I} x_n\| : \ |I| \leqslant m\},$$

$$\psi(m) := \inf\{\|\sum_{n\in I} x_n\| : \ |I| \geqslant m\},$$

$$\mathcal{E}_m := \sup_{x\in X, x\neq 0} \frac{\|x - \mathcal{G}_m(x)\|}{\sigma_m(x)},$$

$$\mathcal{M}_m := \sup_{k\leqslant m} \frac{\sup\{\|\sum_{j\in J} x_j\| : \ |J| = k\}}{\inf\{\|\sum_{j\in J} x_j\| : \ |J| = k\}}.$$

2. Unconditional bases

One of the most fruitful concepts in the Banach space theory concerns the unconditionality of systems. The principal idea of the approach is that we require the space to have a lot of symmetry which we hope to provide a number of useful properties. We refer to [37],[38] as some introductory feedback to this item.

Definition 6 *A biorthogonal system $\Phi = (e_i, e_i^*)_{i\in I}$ is unconditional if there exists a constant K such for all $x \in X$ and any finite $J \subset I$ we have have $\|P_J(x)\| \leqslant K\|x\|$. The smallest such constant K will be called unconditional constant.*

Remark 1 *Note that the above definition is equivalent to requiring that $\|P_J(x)\| \leqslant K\|x\|$ for all (not necessarily finite) $J \subset I$.*

Sometimes we refer to a stronger property which is called symmetry.

Definition 7 *An an unconditional system $\Phi = (e_i, e_i^*)_{i\in I}$ is symmetric if there exists a constant U such for all $x \in X$, any permutation $\pi : I \to I$ and random signs $(\varepsilon_i)_{i\in I}$ we have*

$$\|\sum_{i\in I} \varepsilon_i e_i^*(x) e_{\pi(i)}\| \leqslant U\|x\|$$

The smallest such constant *U* will be called symmetric constant.
Usually in the sequel we will assume that the unconditional system has the unconditional constant equal to 1. This is not a significant restriction since given unconditional system Φ in *X* one can introduce a new norm

$$|||x||| := \sup_{|\lambda_i|\leqslant 1} \|\sum_{i\in I} \lambda_i e_i^*(x) e_i\|.$$

By the classical extreme point argument one can check that this is an equivalent norm on *X*, more precisely $\|x\| \leqslant |||x||| \leqslant 2K\|x\|$ for $x \in X$ and Φ has unconditional constant 1 in $(X, |||\cdot|||)$. In the classical Banach space theory a lot of attention has been paid to understand some features of spaces which admits the unconditional basis. We quote from [1] a property we have announced in the introduction.
Proposition 1 *Let* $(e_i)_{i\in I}$ *be an unconditional basis for X (with constant K). Then* $Y = \operatorname{span}(e_i^*, i \in I)$ *verifies that*

$$K^{-1}\|x\| \leqslant \sup\{|y(x)| : y \in Y, \|y\| \leqslant 1\} \leqslant \|x\|,$$

for all $x \in X$
Proof. Let $x \in X$. Since $Y \subset X^*$, it follows immediately that

$$\sup\{|y(x)| : y \in Y, \|y\| \leqslant 1\} \leqslant \sup\{|x^*(x)| : x^* \in X^*, \|x^*\| \leqslant 1\} = \|x\|.$$

For the other inequality, pick $x^* \in S_{X^*}$ (from unit sphere in *X**) so that $x^*(x) = \|x\|$. Then for each finite *J* we have

$$K^{-1}|(P_J^* x^*)x| \leqslant \frac{|(P_J^* x^*)x|}{\|P_J^* x^*\|} \leqslant \sup\{|y(x)| : y \in Y, \|y\| \leqslant 1\}.$$

Now we let *J* tend to *I* and use that if $\|P_J x - x\| \to 0$ then $|(P_J^* x^*)x| = |x^*(P_J x)| \to \|x\|$.
■
Therefore according to Theorem 1 the optimal *m*-term approximation for unconditional system exists, i.e. $\sigma_m(x)$ is attained at some $y \in \Sigma_m$. We remark that there are a lot of classical spaces which does not admit any unconditional basis and even (e.g. *C*[0, 1] see [1]) cannot be embedded into a Banach space with such a structure.
In the greedy approximation theory we consider the class of unconditional bases as the fine class we usually tend to search for the optimal algorithm (see [14]). The reason is that for unconditional bases for a given $x \in X$ the best *m*-term approximation must be attained at some projection $P_J x$.
Proposition 2 *Let* $\Phi = (e_i, e_i^*)_{i\in I}$ *be a natural biorthogonal system with unconditional constant* 1. *Then for each* $x \in X$ *and each m* = 0, 1, 2, … *there exists a subset* $J \subset I$ *of cardinality m such that* $\|x - P_J x\| = \sigma_m(x)$.
Proof. Let us fix *m* and $x = \sum_{i\in I} a_i e_i \in X$. Let $y_m = \sum_{j\in J} b_j e_j$ be the best *m*-term approximation i.e. $\|x - y_m\| \leqslant \sigma_m(x)$ (the existence is guaranteed by Proposition 1). Note that

$$\|x - P_J x\| = \|x - y_m + P_J y_m - P_J x\| = \|(Id - P_{J_0})(x - y_m)\| \leqslant \|x - y_m\| = \sigma_m(x),$$

which completes the proof. ■

We turn to show that for unconditional systems $\mathcal{E}_m$ and $\mathcal{M}_m$ are comparable. The result we quote from [35] but for concrete systems (see [32]) the answer was known before.

Theorem 2 *If Φ is a natural biorthogonal system with unconditional constant 1, then* $\frac{1}{2}\mathcal{M}_m(\Phi) \leqslant \mathcal{E}_m(\Phi) \leqslant 2\mathcal{M}_m(\Phi)$.

Proof. We have shown in Proposition 2 that we can take the best m-term approximation of x as $T_m(x) = P_{J_0}x$. Clearly $\mathcal{G}_m(x)=P_Jx$ for some $J \subset I$. In order to estimate $\|x-\mathcal{G}_m(x)\|$ we write

$$x - \mathcal{G}_m(x) = x - P_{J_0}x + P_{J_0}x - P_Jx = (x - P_{J_0}x) + P_{J_0\setminus J}x - P_{J\setminus J_0}x =$$
$$P_{I\setminus J}(x - P_{J_0}x) + P_{J_0\setminus J}x$$

so using 1-unconditionality we obtain

$$\|x - \mathcal{G}_m(x)\| \leqslant \|x - T_mx\| + \|P_{J_0\setminus J}x\| \leqslant \sigma_m(x) + \|P_{J_0\setminus J}x\|.$$

Note that $\max\{|e_j^*(x)|: j \in J_0\setminus J\} := c \leqslant \min\{|e_j^*(x)|: j \in J\setminus J_0\}$ and also $|J_0\setminus J| = |J\setminus J_0| \leqslant m$. This implies that $\|P_{J_0\setminus J}x\| \leqslant c\|\sum_{j\in J_0\setminus J} e_j\|$ and $\|P_{J\setminus J_0}x\| \geqslant c\|\sum_{j\in J\setminus J_0} e_j\|$. Thus estimating c from the second inequality and substituting it into the first we get

$$\|P_{J_0\setminus J}x\| \leqslant \frac{\|P_{J\setminus J_0}x\|}{\|\sum_{j\in J\setminus I} e_j\|}\|\sum_{j\in J_0\setminus J} e_j\| \leqslant \mathcal{M}_m\|P_{J\setminus J_0}x\| \leqslant \mathcal{M}_m\sigma_m(x).$$

Consequently

$$\|x - \mathcal{G}_m(x)\| \leqslant \sigma_m(x)(1 + \mathcal{M}_m) \leqslant 2\mathcal{M}_m\sigma_m(x).$$

To show the converse inequality use the following result:

Lemma 1 *For each m there exists disjoint sets J_1 and J_2 with $|J_1| = |J_2| \leqslant m$ such that* $\|\sum_{j\in J_1} e_j\|\|\sum_{j\in J_2} e_j\|^{-1} \geqslant 2^{-1}\mathcal{M}_m$.

Proof. If $\mathcal{M}_m \leqslant 2$ the claim is obvious. Otherwise take sets J_1 and J_2 with $|J_1| = |J_2|$ such that $\|\sum_{j\in J_1} e_j\|\|\sum_{j\in J_2} e_j\|^{-1} \geqslant \max(2, \mathcal{M}_m - \varepsilon)$. For simplicity write

$$a = \|\sum_{j\in J_1} e_j\|, \quad b = \|\sum_{j\in J_2} e_j\|$$
$$a_1 = \|\sum_{j\in J_1\cap J_2} e_j\|, \quad a_2 = \|\sum_{J_1\setminus J_2} e_j\|.$$

With this notation we have $2 < (a/b) \leqslant (a/a_1)$ so $a_1 < a/2$. This implies

$$\frac{a}{b} \leqslant \frac{a_1 + a_2}{b} = \frac{a_1}{b} + \frac{a_2}{b} < \frac{a}{2b} + \frac{a_2}{b},$$

so $a_2/b > a/(2b)$. Thus we have to replace J_1 by any set of proper cardinality which contains $J_1\setminus J_2$ and is disjoint with J_2.

■

We take sets as in Lemma 1 and denote $|J_1| = |J_2| = k \leqslant m$. Let $J_3 \supset J_1$ be a set of cardinality m disjoint with J_2. Consider

$$x := (1+\varepsilon)\sum_{j\in J_2} e_j + (1+\varepsilon/2)\sum_{j\in J_3\setminus J_1} e_j + \sum_{j\in J_1} e_j.$$

Then $\mathcal{G}_m(x) = x - \sum_{j\in J_1} e_j$, so $\|x - \mathcal{G}_m(x)\| = \|\sum_{j\in J_1} e_j\|$. From Proposition 2 we learn that

$$\sigma_m(x) = \min\{\|P_S x\| : S \subset J_2 \cup J_3, \text{ and } |S| = m\} \leqslant$$
$$\leqslant \|P_{J_2} x\| \leqslant (1+\varepsilon)\|\sum_{j\in J_2} e_j\|.$$

This and Lemma 1 give

$$\mathcal{E}_m \geqslant \frac{\|\sum_{j\in J_1} e_j\|}{\sigma_m(x)} \geqslant \frac{\|\sum_{j\in J_1} e_j\|}{(1+\varepsilon)\|\sum_{j\in J_2} e_j\|} \geqslant \frac{1}{2(1+\varepsilon)}\mathcal{M}_m$$

Since ε is arbitrary it completes the proof.

■

More elaborate results of this type are presented in [29].

Theorem 3 *Let* Φ *be natural biorthogonal system with unconditional constant* 1. *Suppose that* $s(m)$ *is a function such that for some* $c > 0$

$$\psi(s(m)) \geqslant c\varphi(m), \quad \text{for } m = 1, 2, \ldots \tag{7}$$

Then

$$\|x - \mathcal{G}_{m+s(m)}(x)\| \leqslant C\sigma_m(x)$$

for some constants C and $m = 0, 1, 2, \ldots$.

Proof. Let us fix $x \in X$ with $\|x\| = 1$ and $m = 0, 1, 2, \ldots$. By Proposition 2, there exits a subset $J \subset I$ of cardinality m such that

$$\sigma_m(x) = \|x - P_J x\|,$$

and $J_0 \subset I$ a subset of cardinality $s(m) + m$ such that $\mathcal{G}_{s(m)+m}(x) = P_{J_0} x$. Using the unconditionality of the system we get

$$\|x - P_J\ x\ \| \geqslant \max\{\|x - P_{J\cup J_0}\ x\ \|, \|P_{J_0\setminus J}\ x\ \}\},$$
$$\|x - P_{J_0}\ x\ \| \leqslant \|x - P_{J\cup J_0}\ x\ \| + \|P_{J\setminus J_0}\ x\ \|.$$

Let $\delta = \inf_{j\in J_0} |x_j^*(x)|$. The again using unconditionality we derive

$$\|P_{J_0\setminus J}\ x\ \| \geqslant \|\delta \sum_{j\in J_0\setminus J} x_j\| \geqslant \delta\psi(s(m)). \tag{8}$$

Since for $j \in J \setminus J_0$ we have $|x_j^*(x)| \leqslant \delta$, we get

$$\|P_{J\setminus J_0}\ x\ \| \leqslant \delta \| \sum_{j\in J\setminus J_0} x_j \| \leqslant \delta\varphi(m). \tag{9}$$

From (8), (9) and (7) we get

$$\|P_{J\setminus J_0}\ x\ \| \leqslant C\|P_{J_0\setminus J}\ x\ \|$$

so

$$\|x - \mathcal{G}_{s(m)+m}(x)\| = \|x - P_{J_0}\ x\ \| \leqslant C(\|x - P_{J\cup J_0}\ x\ \| + \|P_{J_0\setminus J}\ x\ \|) \leqslant$$
$$\leqslant 2C\|x - P_J\ x\ \| \leqslant 2C\sigma_m(x).$$

■

Let $\Phi = (e_i, e_i^*)_{i\in I}$ be a biorthogonal system. The natural question rises when $e_i^*,\ i \in I$ is the unconditional system in *X**. The obvious obstacle may be that such system does not verify (4). For example the standard basis $(e_i)_{i=1}^{\infty}$ in l_1 cannot have its dual to be a basis in $l_1^* = l_\infty$, since the latter is not separable. However, if we consider it as a system in span$\{e_i^* :\ i \in I\}$, then it will satisfy all our assumptions and thus we denote such system by Φ*. Note that if Φ is unconditional then so is Φ*.

Theorem 4 *Let* Φ *be natural biorthogonal system with unconditional constant* 1. *Then*

$$\mathcal{M}_m(\Phi^*) \leqslant 2\log m\mathcal{M}_m(\Phi),$$

for m = 2, 3, ….

Proof. Let us fix $m,\ k \leqslant m$, and a set $J \subset I$ of cardinality *k*. We have

$$\|\sum_{j\in J} e_j^*\| \geqslant k\|\sum_{j\in J} e_j\|^{-1} \geqslant \frac{k}{\varphi(k)}. \tag{10}$$

On the other hand there exists $x \in X$ with $\|x\| = 1$ such that

$$\|\sum_{j\in J} e_j^*\| \leqslant 2\sum_{j\in J} |e_j^*(x)| \tag{11}$$

Let $\sigma : \{1, ..., |J|\} \to J$ be such that $|x_{\sigma(j)}^*| \leqslant |x_{\sigma(k)}^*|$ whenever $k \geqslant j$. From 1-unconditionality we deduce that

$$|e_{\sigma(j)}^*(x)|\|\sum_{k=1}^{j} e_{\sigma(k)}\| \leqslant \|x\| = 1$$

therefore

$$\sum_{j\in J} |e_j^*(x)| \leqslant \sum_{j=1}^{k} \psi(j)^{-1}. \tag{12}$$

Thus from (10),(11) and (12) using the fact that $\frac{\varphi(k)}{k}$ is decreasing, we obtain that

$$\mathcal{M}_m(\Phi^*) \leqslant 2\sup_{k\leqslant m}\frac{1}{k}\sum_{j=1}^{k}\frac{\varphi(k)}{\psi(j)} \leqslant 2\sup_{k\leqslant m}\sum_{j=1}^{k}\frac{1}{j}\frac{\varphi(j)}{\psi(j)} \leqslant$$
$$\leqslant 2\log m\sup_{j\leqslant m}\frac{\varphi(j)}{\psi(j)} \leqslant 2\log m\mathcal{M}_m(\Phi). \tag{13}$$

■

Theorems 3 and 4 are quoted from [40] but the almost the same arguments were used earlier in [11] and [27].

3. Greedy bases

The first step to understand the idea of greedy systems in Banach spaces is to give their characterization in terms of some basic notions. The famous result of Konyagin and Temlyakov [26] states that being a greedy basis is equivalent to be an unconditional and democratic basis. We start from introducing these two concepts.

The second concept we need to describe greedy bases concerns democracy. The idea is that we expect the norm $\|\sum_{j\in J}x_j\|$ being essentially a function of $|J|$ rather then from *J* itself.

Definition 8 *A biorthogonal system* Φ *is called democratic if there exists a constant D such that for any two finite subsets* $J_1, J_2 \subset I$ *with* $|J_1| = |J_2|$ *we have*

$$\|\sum_{j\in J_1}e_j\| \leqslant D\|\sum_{j\in J_2}e_j\|.$$

The smallest such constant *D* will be called a democratic constant of Φ.

We state the main result of the section.

Theorem 5 *If the natural biorthogonal system* Φ *is greedy with the greedy constant less or equal C, then it is unconditional with unconditional constant less or or equal C and democratic with the democratic constant less or equal* C^2. *Conversely if it is unconditional with constant K and democratic with constant D, then it is greedy with greedy constant less or equal* $K + K^3D$.

Proof. Assume first that Φ is greedy with the greedy constant *C*. Let us fix a finite set $J \subset I$ of cardinality *m*, $x \in X$ and a number $N > \sup_{i\in I}|e_i^*|$. We put $y := x - P_Jx + N\sum_{j\in J}e_j$. Clearly $\sigma_m(y) \leqslant \|x\|$ and $\mathcal{G}_m(y) = N\sum_{j\in J}e_j$. Thus

$$\|x - P_Jx\| = \|y - \mathcal{G}_m(y)\| \leqslant C\sigma_m(y) \leqslant C\|x\|. \tag{14}$$

Therefore Φ is unconditional according to Definition 6.

To show that Φ is democratic we fix two subsets $J_1, J_2 \subset I$ with $|J_1| = |J_2| = m$. Then we choose a third subset $J_3 \subset I$ such that $|J_3| = m$ and $J_1 \cap J_3 = \emptyset$, $J_2 \cap J_3 = \emptyset$. Defining $x = (1+\varepsilon)\sum_{j\in J_1}e_j + \sum_{j\in J_3}e_j$ we have that

$$(1+\varepsilon)\|\sum_{j\in J_1}e_j\|$$

and

$$\| \sum_{j\in J_3} e_j \| = \|x - \mathcal{G}_m(x)\| \leqslant C\sigma_m(x) \leqslant C(1+\varepsilon)\| \sum_{j\in J_1} e_j\|.$$

Analogously we get

$$\| \sum_{j\in J_2} x_j\| \leqslant C(1+\varepsilon)\| \sum_{j\in J_3} x_j\|$$

and the conclusion follows.

Now we will prove the converse. Fix $x \in X$ and m = 0, 1, 2, …. Choose $y_m = \sum_{j\in J} a_j e_j$ with $|J| = m$ and $\|x - y_m\| \leqslant \sigma_m(x) + \varepsilon$. Clearly

$$\mathcal{G}_m(x) = \sum_{j\in J_0} e_j^*(x) e_j = P_{J_0} x$$

for appropriate $J_0 \subset I$ with $|J_0| = m$. We write

$$\|x - \mathcal{G}_m(x)\| = \|x - P_{J_0}x + P_J x - P_J x\| = \|x - P_J x + P_{J\setminus J_0} - P_{J_0\setminus J}x\|. \quad (15)$$

Using unconditionality we get

$$\|x - P_J x - P_{J_0\setminus J}x\| = \|x - P_{J_0\cup J}x\| = \|P_{I\setminus(J_0\cup J)}(x - y_m)\| \leqslant K(\sigma_m(x) + \varepsilon) \quad (16)$$

and analogously

$$\|P_{J_0\setminus J}x\| \leqslant K(\sigma_m(x) + \varepsilon).$$

From the definition of $\mathcal{G}_m$ we infer that

$$\alpha = \min_{j\in J_0\setminus J} |x_j^*(x)| \geqslant \max_{j\in J\setminus J_0} |x_j^*(x)| = \beta,$$

so from unconditionality we get

$$K\|P_{J_0\setminus J}x\| \geqslant \alpha\| \sum_{j\in J_0\setminus J} e_j\| \quad (17)$$

and

$$\|P_{J\setminus J_0}x\| \geqslant K\beta\| \sum_{j\in J\setminus J_0} e_j\|, \quad (18)$$

Since $|J\setminus J_0| = |J_0\setminus J|$ from (17) and (18) and democracy we deduce that

$$\|P_{J\setminus J_0}x\| \leqslant K^2 D\|P_{J_0\setminus J}x\|. \quad (19)$$

From (15), (16) and (19) we get (ε is arbitrary)

$$\|x - \mathcal{G}_m(x)\| \leqslant (K + K^3 D)\sigma_m(x).$$

■

Remark 2 *The above proof is taken from [26]. However some arguments (except the proof that greedy implies unconditional), were already in previous papers [32] and [35].*

If we disregard constants Theorem 5 says that a system is greedy if and only if it is unconditional and democratic. Note that in particular Theorem 5 implies that a greedy system with constant 1 (i.e. 1-greedy) is 1-unconditional and 1-democratic. However this is not the characterization of bases with greedy constant 1 (see [40]). The problem of isometric characterization has been solved recently in [2]. To state the result we have to introduce the so called Property (A).

Let $(e_i)_{i=1}^{\infty}$ be a Schauder basis of X. Given $x \in X$, the support of x denoted supp consists of those $i \in \mathbb{N}$ such that $e_i^* \neq 0$. Let $M(x)$ denote the subset of supp where the coordinates (in absolute value) are the largest. Clearly the cardinality of $M(x)$ is finite for all $x \in X$. We say that 1-1 map $\pi : \mathrm{supp}x \to \mathbb{N}$ is a greedy permutation of x if $\pi(i) = i$ for all $i \in \mathrm{supp}x \setminus M(x)$ and if $i \in M(x)$ then, either $\pi(i) = i$ or $\pi(i) \in \mathbb{N} \setminus \mathrm{supp}x$. That is a greedy permutation of x puts those coefficients of x whose absolute value is the largest in gaps of the support of x, if there are any. If $\mathrm{supp}x \neq \mathbb{N}$ we will put $M_\pi^*(x) = \{j \in M(x) : \pi(j) \neq j\}$. Finally we denote by $\Pi_G(x)$ the set of all greedy permutation of x.

Definition 9 *A Schauder basis $(e_i)_{i=1}^{\infty}$ for Banach space X has property (A) if for any $x \in X$ we have*

$$\| \sum_{i \in \mathrm{supp}x} e_i^*(x) e_i \| = \| \sum_{i \in \mathrm{supp}x} \varepsilon_{\pi(i)} e_i^*(x) e_{\pi(i)} \|.$$

for all $\pi \in \Pi_G(x)$ and all signs $(\epsilon_i)_{i=1}^{\infty}$, $(\text{i.e.} \epsilon_i = \pm 1)$ with $\varepsilon_{\pi(i)} = 1$ *if* $i \notin M_\pi^*(x)$.

Note that property (A) is a weak symmetry condition for largest coefficients. We require that there is a symmetry in the norm provided its support has some gaps. When $\mathrm{supp}x = \mathbb{N}$ then the basis does not allow any symmetry in the norm of x. The opposite case occurs when $x = \sum_{j \in J_0} e_j$ and J_0 is finite, then $\|x\| = \| \sum_{j \in J} e_j \|$ for any $J \subset \mathbb{N}$ of cardinality $|J_0|$.

Theorem 6 *A basis $(e_i)_{i=1}^{\infty}$ for a Banach space X is* 1*-greedy if and only if it is* 1*- unconditional and satisfies property (A).*

Another important for application result is the duality property.

Remark 3 *Suppose that Φ is greedy basis and that $\varphi(m) \simeq m^\alpha$ with* $0 < \alpha < 1$. *Then Φ^* is also greedy.*

Proof. From Theorem 5 we know that Φ is unconditional, so we can renorm it to be 1-unconditional. Also, because Φ is greedy we have $\varphi(m) \simeq \psi(m)$. We repeat the proof of Theorem 4 but in (13) we explicitly calculate as follows:

$$\mathcal{M}_m(\Phi^*) \leqslant 2C \sup_{k \leqslant m} \frac{1}{k} \sum_{j=1}^{k} \frac{k^\alpha}{j^\alpha} \leqslant \mathrm{const},$$

so Φ^* is greedy

■

This is a special case of Theorem 5.1 from [11]. We recall that it was proved in [21] that each unconditional basis in L_p, $1 < p < 1$, has a subsequence equivalent to the unit vectors basis in l_p, so for each greedy basis Φ in L_p we have $\varphi(\Phi, m) \simeq m^{1/p}$. Thus we get:

Corollary 1 *If Φ is a greedy basis in L_p, $1 < p < 1$, then Φ^* is a greedy basis in L_q, $1/p + 1/q = 1$.*

4. Quasi greedy bases

In this section we characterize the quasi-greedy systems. The well known result of Wojtaszczyk [35] says quasi-greedy property is a kind of uniform boundedness principle.

Theorem 7 *A natural biorthogonal system is quasi greedy if and only if there exists a constant C such that for all $x \in X$ and $m = 0, 1, 2, \ldots$ we have*

$$\mathcal{G}_m(\Phi, x) \leqslant C\|x\|.$$

The smallest constant C in the above theorem will be called quasi greedy constant of the system Φ.

Proof. 1⇒2. Since the convergence is clear for x's with finite expansion in the biorthogonal system, let us assume that x has an infinite expansion. Take $x_0 = \sum_{j\in J} a_j e_j$ such that $\|x - x_0\| < \varepsilon$ where $J \subset I$ is a finite set and $a_j \neq 0$ for $j \in J$. If we take m big enough we can ensure that $\mathcal{G}_m(x - x_0) = \sum_{j\in J_0} e_j^*(x - x_0)e_j$ with $J_0 \supset J$ and $\mathcal{G}_m(x) = \sum_{j\in J_0} e_j^*(x)e_j$. Then

$$\|x - \mathcal{G}_m(x)\| \leqslant \|x - x_0\| + \|x_0 - \mathcal{G}_m(x)\| \leqslant \varepsilon + \|\mathcal{G}_m(x_0 - x)\| \leqslant (C+1)\varepsilon.$$

This gives 2.

2⇒1. Let us start with the following lemma.

Lemma 2 *If 2 does not hold, then for each constant K and each finite set $J \subset I$ there exist a finite set $J_0 \subset I$ disjoint from J and a vector $x = \sum_{j\in J_0} a_j e_j$ such that $\|x\| = 1$ and $\|\mathcal{G}_m(x)\| \geqslant K$ for some m.*

Proof. Let us fix M to be the minimum of the norms of the (linear) projections $P_\Omega(x) = \sum_{j\in\Omega} e_j^*(x)e_j$ where $\Omega \subset J$. Let us start with a vector x_1 such that $\|x_1\| = 1$ and $\|\mathcal{G}_m(x_1)\| \geqslant K_1$ where K_1 is a big constant to be specified later. Without loss of generality we can assume that all numbers $|e_i^*(x_1)|$ are different. For $x_2 = x_1 - \sum_{j\in J} e_j^*(x_1)e_j$ we have $\|x_2\| \leqslant M + 1$ and $\mathcal{G}_m(x_1) = \mathcal{G}_k(x_2) + P_\Omega(x_1)$ for some $k \leqslant m$ and $\Omega \subset J$. Thus $\|\mathcal{G}_k(x_2)\| \geqslant K_1 - M$ and for $x_3 = x_2\|x_2\|^{-1}$ we have $\|\mathcal{G}_k(x_3)\| \geqslant (K_1 - M)/(1+M)$. Let us put

$$\delta = \inf\{|e_i^*(\mathcal{G}_k(x_3))| : e_i^*(\mathcal{G}_k(x_3)) \neq 0\}$$

and take a finite set J_1 such that for $i \notin J_1$ we have $|e_i^*(x_3)| \leqslant \delta/2$. Let us take η very small with respect to $|J_1|$ and $|J|$ and find x_4 with finite expansion such that $\|x_3 - x_4\| < \eta$. If η is small enough we can modify all coefficients of x_4 from J_1 and J so that the resulting x_5 will have its k biggest coefficients the same as x_3 and $\|x_4 - x_5\| < \delta$. Moreover x_5 will have the form $x_5 = \sum_{j\in J_0} e_j^*(x_5)e_j$ with J_0 finite and disjoint from J. Since $\mathcal{G}_k(x_5) = \mathcal{G}_k(x_3)$, for $x = x_5\|x_5\|^{-1}$ we get $\|\mathcal{G}_k(x)\| \geqslant (K_1 - M)/(c\,(1+M))$ which can be made greater or equal K if we take K_1 big enough. ■

Using Lemma 2 we can apply the standard gliding hump argument to get a sequence o the vectors $y_n = \sum_{j\in J_n} a_j e_j$ with sets J_n disjoint and $\|y_n\| = 1$, a decreasing sequence of positive numbers $\varepsilon_n \leqslant 2^{-n}$ such that if $e_j^*(y_n) \neq 0$ then $|e_j^*(y_n)| \geqslant \varepsilon_n$ and a sequence of integers m_n such that $\|\mathcal{G}_{m_n}(y_n)\| \geqslant 2^n \prod_{j=1}^{n-1} \varepsilon_j^{-1}$. Now we put $x = \sum_{n=1}^{\infty} (\prod_{j=1}^{n-1} \varepsilon_j) y_n$. This series is clearly convergent in *X*. If we write $x = \sum_{i\in I} b_i e_i$ we infer that

$$\inf\{|b_j| : j \in \bigcup_{s=1}^{N} J_s \text{ and } b_j \neq 0\} \geqslant \prod_{l=1}^{N} \varepsilon_l \geqslant \max\{|b_j| : j \notin \bigcup_{s=1}^{N} J_s\}.$$

This implies that for $k = \sum_{l=1}^{N-1} |J_l| + m_N$ we have

$$\mathcal{G}_k(x) = \sum_{n\leqslant N} (\prod_{l=1}^{n-1} \varepsilon_l) y_n + \mathcal{G}_{m_N}((\prod_{l=1}^{N} \varepsilon_l) y_{N+1})$$

so

$$\|\mathcal{G}_k(x)\| \geqslant (\prod_{l=1}^{N} \varepsilon_l)\|\mathcal{G}_{m_N}(y_{N+1})\| - C \geqslant 2^{N+1} - C.$$

Thus $\mathcal{G}_m(x)$ does not converge to *x*

■

One of the significant features of quasi greedy systems is that they are closely related to the unconditionality property.

Remark 4 *Each unconditional system is quasi greedy.*

Proof. Note that for an unconditional system $\Phi = (e_i, e_i^*)_{i\in I}$ and each $x \in X$ the series $\sum_{i\in I} e_i^*(x) e_i$ converges unconditionally (we can change the order of *I*). In particular the convergence holds for any finite-set approximation of *I* and hence Φ is quasi greedy.

■

There is a result in the opposite direction, which shows that quasi-greedy bases are rather close to unconditional systems.

Definition 10 *A system Φ is called unconditional for constant coefficients if there exits constants $c_1 > 0$ and $c_2 < 1$ such that for finite $J \subset I$ and each sequence of signs $(\varepsilon_j)_{j\in J} = \pm 1$ we have*

$$c_1 \|\sum_{j\in J} e_j\| \leqslant \|\sum_{j\in J} \varepsilon_j e_j\| \leqslant c_2 \|\sum_{j\in J} e_j\|. \tag{20}$$

Proposition 3 *If (Φ) has a quasi-greedy constant C then it is unconditional for constant coefficients with $c_1 = C^{-1}$ and $c_2 = C$.*

Proof. For a given sequence of signs $(\varepsilon_j)_{j\in J}$ let us define the set $J_0 = \{j \in J : \varepsilon_j = 1\}$. For each ε > 0 and ε < 1 we apply Theorem 7 and we get

$$\|\sum_{j\in J_0} e_j\| \leqslant C\|\sum_{j\in J_0} e_j + \sum_{j\in J\setminus J_0} (1-\varepsilon) e_j\|.$$

Since this is true for each ε > 0 we easily obtain the right hand side inequality in (20). The other inequality follows by analogous arguments.

■

The quasi greedy bases may not have the duality property. For example for the quasi greedy basis in l_1, constructed in [12] the dual basis is not unconditional for constant coefficients and so it is not quasi greedy. On the other hand dual of a quasi greedy system in a Hilbert space is also quasi greedy (see Corollary 4.5 and Theorem 5.4 in [11]). Otherwise not much has been proved for quasi greedy bases.

5. Examples of systems

In this section we discuss a lot of concrete examples of biorthogonal systems. We remark here that all of the discussed concepts of: greedy, quasi greedy, unconditional symmetric and democratic systems, are up to a certain extent independent of the normalization of the system. Namely we have (cf. [40]):

Remark 5 If $(\lambda_i)_{i\in I}$ *is a sequence of numbers such that*

$$0 < \inf_{i\in I} |\lambda_i| \leqslant \sup_{i\in I} |\lambda_i| < \infty$$

and $\Phi = (x_i, x_i^*)$ *is a system which satisfies any of the Definitions 4-8, then the system* $(\lambda_i e_i, \lambda_i^{-1} e_i^*)_{i\in I}$ *verifies the same definitions.*

The most natural family of spaces consists of L_p spaces $1 \leq p \leq \infty$ and some of their variations, like rearrangement spaces. As for the systems we will be mainly interested in wavelet type systems, especially the Haar system or similar, and trigonometric or Wlash system.

5.1 Trigonometric systems

Clearly standard basis in l_p, $p > 1$ is greedy. The straightforward generalization of such system into $L_p(\mathbb{R})$ space is the trigonometric system $(e^{ijt})_{j\in\mathbb{Z}}$. Such system may be complicated to the Walsh system in $L_p(\mathbb{R}^d)$, given by $(e^{i\langle j,t\rangle})_{j\in\mathbb{Z}^d}$, where $t \in \mathbb{R}^d$. Unfortunately the trigonometric system is not quasi greedy even in L_p. To show this fact we use Proposition 3, i.e. we prove that such systems are not unconditional for constant coefficients whenever $p \neq 2$.

Suppose that for some fixed $1 \leq p < \infty$ trigonometric system verifies (20). Then taking the average over signs we get

$$\left(\int_0^1 \| \sum_{j=1}^N r_j(t) e^{ijn} \|_p^p dt\right)^{1/p} \simeq \| \sum_{j=1}^N e^{ijn} \|_p.$$

The symbol r_j in the above denotes the Rademacher system. The right hand side (which is the L_p norm of the Dirichlet kernel) is of order $N^{1-\frac{1}{p}}$ if $p > 1$ and of order log$N$ when $p = 1$. Changing the order of integration and using the Kchintchine inequality we see that the left hand side is of order $\sqrt{N}$. To decide the case $p = \infty$ we recall that the well-known Rudin Shapiro polynomials are of the form $p_N(s) = \sum_{j=1}^N \pm e^{ijs}$ for appropriate choice of $\|p_N\|_\infty \simeq \sqrt{N}$ while the L_∞ norm of the Dirichlet Kernel is clearly equal to N. This violates (20). Those results are proved in [40], [30], [8] and [35].

5.2 Haar systems

We first recall the definition of Haar system in L_p space. The construction we describe here is well known an we follow its presentation from [40]. We start from a simple (wavelet) function:

$$h(t) = \begin{cases} 1 & \text{if} \quad 0 \leqslant t < 1/2 \\ -1 & \text{if} \quad 1/2 \leqslant t < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Clearly $\operatorname{supp} h = [0, 1)$. For pair $(j, k) \in \mathbb{Z}^2$ we define the function $h_{j,k}(t) := h(2^j t - k)$. The support of $h_{j,k}$ is dyadic interval $I = I(j, k) = [k2^{-j}, (k+1)2^{-j}]$. The usual procedure is to index Haar functions by dyadic intervals I and write h_I instead of $h_{j,k}$. We denote by $\mathcal{D}$ the set of all dyadic subintervals of $\mathbb{R}$. It is a routine exercise to check that the system $\{h_{j,k} : (j, k) \in \mathbb{Z}^2\} = \{h_I : I \in \mathcal{D}\}$ is complete orthogonal system in $L_2(\mathbb{R})$. Note that whenever we consider the Haar system in a specified function space X on $\mathbb{R}$ we will consider the normalized system $h_I / \|h_I\|_X$.

There are two common Haar systems in $\mathbb{R}^d$:

1. The tensorized Haar system, denoted by $\mathfrak{h}_d^p$ and defined as follows: If $J = J_1 \times \ldots \times J_d$ where $J_1, \ldots, J_d \in \mathcal{D}$, then we put $h_J(t_1, \ldots, t_d) := h_{J_1}(t_1) \ldots h_{J_d}(t_d)$. One checks trivially that the system $\{h_J : J \in \mathcal{D}^d\}$ is a complete, orthogonal system in $L_2(\mathbb{R}^d)$. We will consider this system normalized in L_p with $1 \leq p \leq \infty$, i.e. $\mathfrak{h}_d^p = \{H_J^p : J \in \mathcal{D}^d\}$, where $H_J^p = \|h_J\|_p^{-1} h_J$. The main feature of the system is that supports of the functions are dyadic parallelograms with arbitrary sides.
2. The cubic Haar system, denoted by $\mathfrak{h}_d^p$ defined as follows: We denote by $h^1(t)$ the functions $h(t)$ defined in (21) and by $h^0(t)$ the function $1_{[0,1]}$. For fixed $d = 1, 2, \ldots$ let $\mathcal{C}$ denotes the set of sequences $\delta = (\delta_1, \ldots, \delta_d)$ such that $\delta_i = 0$ or 1 and $\sum_{i=1}^d \delta_i > 0$. For $\delta \in \mathcal{C}$, $j \in \mathbb{Z}$ and $k \in \mathbb{Z}^d$ we define a function $h_{j,k}^\delta$ on $\mathbb{R}^d$ by the formula

$$h_{j,k}^\delta(t_1, \ldots, t_d) := 2^{jd/2} \prod_{i=1}^d h^{\delta_i}(2^j x - k_i). \tag{22}$$

   Again it is a routine exercise to show that the system $(h_{j,k}^\delta)$ where δ varies over $\mathcal{C}$, i varies over $\mathbb{Z}$ and k varies over $\mathbb{Z}^d$ is a complete orthonormal system in $L_2(\mathbb{R}^d)$. As before we consider the system normalized in $L_p(\mathbb{R}^d)$, namely $\mathfrak{h}_d^p = \{H_\alpha^p\}_{\alpha \in \mathcal{J}(d)}$ where $\mathcal{J}(d) = \mathcal{C} \times \mathbb{Z} \times \mathbb{Z}^d$ and for $\alpha = (\delta, j, k) \in \mathcal{J}(d)$ we have $H_\alpha^p = \|h_{j,k}^\delta\|_p^{-1} h_{j,k}^\delta$. The feature of this system is that supports of the functions are all dyadic cubes. Therefore one can restrict the Haar system $\mathfrak{h}_d^p$ to the unite cube $[0, 1]^d$. We simply consider all Haar functions whose supports are contained in $[0, 1]^d$ plus the constant function. In this way we get the Haar system in $L_p[0, 1]^d$.

The above approach can be easily generalized to any wavelet basis. In the wavelet construction we have a multivariate scaling function $\varphi^0(t)$ and the associated wavelet $\varphi^1(t)$

on $L_p(\mathbb{R})$. We assume that both φ^0 and φ^1 have sufficient decay to ensure that $\varphi^0, \varphi^1 \in L_1(\mathbb{R}) \cap L_\infty(\mathbb{R})$. Clearly functions $1_{[0,1]}$ and $h(t)$ are the simplest example of the above setting, i.e. of scaling and wavelet function respectively. This concept may be extended to $\mathbb{R}^d$, i.e we can define a tensorized wavelet basis, though since we do not study such examples in this chapter we refrain from detailing the construction.

5.3 Haar systems in L_p spaces

Since Haar systems play important role in the greedy analysis we discuss some of their properties. The main tool in our analysis of L_p will be the Khintchine inequality which allows to use an equivalent norm on the space.

Proposition 4 *If $\Phi = (e_i, e_i^*)_{i\in I}$ is an unconditional system in L_p, $1 < p < \infty$, then the expression*

$$\|\|x\|\|_p = \Big(\int \Big(\sum_{n\in I} |e_i^*(x)|^2 |e_i(s)|^2 ds\Big)^{p/2}\Big)^{1/p} \tag{23}$$

gives an equivalent norm on L_p.

The above proposition fails for $p = 1$ but if we introduce the norm given by (23) for $p = 1$, then we obtain a new space denoted as H_1, in which the Haar system h_1^1 is unconditional. The detail construction of the space may be found in [37], 7.3.

We show that one of our Haar systems h_d^p is greedy whereas the second one $\mathfrak{h}_d^p$ is not. We sketch briefly these results. The first result was first proved in [33] but we present argument given in [22] and [40] which is a bit easier.

Theorem 8 *The Haar h_d^p is greedy basis in $L_p(\mathbb{R}^d)$ for $d = 1, 2, \ldots$ and $1 < p < \infty$. The system h_1^1 is greedy in H_1.*

Proof. The unconditionality of the Haar system is clear from Proposition 4. Therefore we only need to prove that h_d^p is democratic in $L_p(\mathbb{R}^d)$ for $d = 1, 2, \ldots$ (and also in H_1). Let $J \subset \mathcal{J}(d)$ be a finite set. Note that if the cube Q is the support of the Haar function H_α^p, then $|H_\alpha^p| = |Q|^{-1/p} 1_Q$. Thus, for each $t \in \mathbb{R}^d$, the non-zero values of the Haar functions $H_\alpha^p(t)$ belong to a geometric progression with ratio 2^d. Then we check that for a given $t \in \mathbb{R}^d$ there are at most 2^d-1 Haar functions which take a given non zero value at this point. Thus defining $2^{M(t)} := \max_{\alpha\in J} |H_\alpha^p(t)|^p$, we obtain that

$$2^{M(t)} \geqslant c(d) \sum_{\alpha\in J} |H_\alpha^p(t)|^p$$

for some constant $c(d) > 0$. So

$$\Big(\int \Big(\sum_{\alpha\in J} |H_\alpha^p(t)|^2\Big)^{p/2}\Big)^{1/p} \geqslant \Big(\int 2^{M(t)} dt\Big)^{1/p} \geqslant$$

$$\geqslant \Big(\int c(d) \sum_{\alpha\in J} |H_\alpha^p(t)|^p dt\Big)^{1/p} = c(d)^{1/p} |J|^{1/p}.$$

We recall that for a given $t \in \mathbb{R}^d$ there are at most 2^d-1 Haar functions which have the same non zero value at this point. Therefore, following the same geometric progression argument we see that for each $t \in \mathbb{R}^d$ we have

$$\sum_{\alpha\in J}|H_\alpha^p(t)|^2 \leqslant C(d)|H_{\alpha_0}^p(t)|^2.$$

for some constant $C(d) < \infty$ and $\alpha_0 \in J$ depending on t. Thus

$$\left(\int\left(\sum_{\alpha\in J}|H_\alpha^p(t)|^2\right)^{p/2}dt\right)^{1/p} \leqslant \left(\int C(d)\sum_{\alpha J}|H_\alpha^p(t)|^p dt\right)^{1/p} \leqslant C(d)^{1/p}|J|^{1/p}.$$

It shows that $\left(\int\left(\sum_{\alpha\in J}|H_\alpha^p(t)|^2\right)^{p/2}dt\right)^{1/p}$ is comparable with $|J|^{1/p}$, which in the view of Proposition 4 completes the proof.

■

The second result shows that $\mathfrak{h}_d^p$ is not greedy in L_p. We recall that for as system, $\mathfrak{h}_d^p$ we have used intervals $I \in \mathcal{D}^d$ as the indices. We first prove the following:

Proposition 5 *For $d = 1, 2, \ldots$ and $1 < p < \infty$ in $L_p(\mathbb{R}^d)$ we have*

$$\left(\sum_{I\in J}|a_I|^p\right)^{1/p}(\log|J|)^{\left(\frac{1}{2}-\frac{1}{p}\right)} \leqslant \left\|\sum_{I\in J}a_I h_I^d\right\| \leqslant \left(\sum_{I\in J}|a_I|^p\right)^{1/p} \tag{24}$$

for $p \leq 2$, and

$$\left(\sum_{I\in J}|a_I|^p\right)^{1/p} \leqslant \left\|\sum_{I\in J}a_I h_I^d\right\| \leqslant (\log|J|)^{\left(\frac{1}{2}-\frac{1}{p}\right)d}\left(\sum_{I\in J}|a_I|^p\right)^{1/p} \tag{25}$$

Proof. The right hand side inequality in (24) is easy. We simply apply the Holder inequality with exponent $\frac{2}{p} \geqslant 1$ to the inside sum and we get

$$\left(\int_{\mathbb{R}^d}\left(\sum_{I\in J}|a_I h_I^d(t)|^2\right)^{p/2}dt\right)^{1/p} = \left(\sum_{I\in J}|a_I|^p\right)^{1/p}. \tag{26}$$

To show the left hand side we will need the following result:

Lemma 3 *For $d = 1$ and $1 \leq p < \infty$ and for any finite subset $J \subset \mathcal{D}$ we have*

$$2^{-1/p}|J|^{1/p} \leqslant \left(\int_{\mathbb{R}}\left(\sum_{I\in J}|h_I^1(t)|^2\right)^{p/2}dt\right)^{1/p}.$$

Proof. Let us denote $2^{M(t)} = \max_{I\in J}|h_I^1(t)|^p$. From the definition of the Haar system we obtain that $2^{M(t)} \geqslant \frac{1}{2}\sum_{I\in J}|h_I^1(t)|^p$ so

$$\left(\int_{\mathbb{R}^d}\left(\sum_{I\in J}|h_I^1(t)|^2\right)^{p/2}dt\right)^{1/p} \geqslant \left(\int_{\mathbb{R}}2^{M(t)}dt\right)^{1/p} \geqslant \left(\frac{1}{2}\int_{\mathbb{R}}\sum_{I\in J}|h_I^1(t)|^p dt\right)^{1/p} = 2^{-1/p}|J|^{1/p}.$$

■

Now we fix $d = 1$ and $1 < p \leq 2$. Let $\sigma : \{1, 2, \ldots, |J|\} \to J$ be such that $|a_{\sigma(i)}|$ is a decreasing sequence. Fix s such that $2^s \leqslant |J|$ and we put

$$f_k(t) = (\sum_{j=2^k}^{2^{k+1}-1} |a(\sigma(j))h^d_{\sigma(j)}(t)|^2)^{1/2}$$

Then

$$(\int_{\mathbb{R}} (\sum_{I\in J} |h^d_I(t)|^2)^{p/2} dt)^{1/p} \geqslant (\int_{\mathbb{R}} (\sum_{k=0}^{s} f_k^2(t))^{p/2} dt)^{1/p} \geqslant (\int_{\mathbb{R}} (\sum_{k=0}^{s} (f_k^p(t))^{2/p})^{p/2} dt)^{1/p} \geqslant$$
$$\geqslant ((\sum_{k=0}^{s} (\int_{\mathbb{R}} f_k^p(t) dt)^{2/p})^{p/2})^{1/p} \geqslant (\sum_{k=0}^{s} (\int_{\mathbb{R}} f_k^p(t) dt)^{2/p})^{1/2}.$$

Hence using Lemma 3 we obtain that

$$(\int_{\mathbb{R}} (\sum_{I\in J} |h^d_I(t)|^2)^{p/2} dt)^{1/p} \geqslant (\sum_{k=0}^{s} 2^{2(k-1)/p} |a_{\sigma(2^k)}|^2)^{1/2}.$$

Since

$$\sum_{I\in J} |a_I|^p = \sum_{j=1}^{|J|} |a_{\sigma(j)}|^p \leqslant \sum_{k=0}^{s} 2^k |a_{\sigma(2^k)}|^p \leqslant s^{1-p/2} (\sum_{k=0}^{s} 2^{2k/p} |a_{\sigma(2^k)}|^2)^{p/2},$$

we derive

$$(\int_{\mathbb{R}} (\sum_{I\in J} |h^d_I(t)|^2)^{p/2} dt)^{1/p} \geqslant 2^{-1/p} (\log |J|)^{-(1-p/2)1/p} (\sum_{I\in J} |a_I|^p)^{1/p} =$$
$$= 2^{-1/p} (\log |J|)^{1/2-1/p} (\sum_{I\in J} |a_I|^p)^{1/p}.$$

Therefore we have established (24) for $d = 1$. We turn to show the left hand side inequality in (24) by induction on d. Suppose we have (24) valid for d-1. Given a finite set $J \subset \mathcal{D}^d$ we write each $I \in J$ as $I = A \times B$ with $A \in \mathcal{D}$ and $B \in \mathcal{D}^{d-1}$ and then $h^d_I(t) = h_A(t_1) h^{d-1}_B(\xi)$ where $\xi = (t_2, \ldots, t_d)$. We denote $S := (\int_{\mathbb{R}} (\sum_{I\in J} |h^d_I(t)|^2)^{p/2} dt)^{1/p}$ and estimate

$$S = (\int_{\mathbb{R}} \int_{\mathbb{R}^{d-1}} (\sum_{I\in J} |a_I h_A(t_1)|^2 |h^{d-1}_B(\xi)|^2)^{p/2} dt_1 \, d\xi)^{1/p} =$$
$$= \int_{\mathbb{R}} (\int_{\mathbb{R}^{d-1}} (\sum_B ((\sum_A |a_I h_A(t_1)|^2) |h^{d-1}_B(\xi)|^2)^{p/2} d\xi) dt_1)^{1/p}. \tag{27}$$

For each t_1 we apply the inductive hypothesis (note that the number of different B's is at most J) and we continue the estimates

$$S \geqslant C(d-1,p)(\log|J|)^{(d-1)(1/2-1/p)}\Big(\int_{\mathbb{R}}\sum_{B}\Big(\sum_{A}|a_I h_A(t_1)|^2 dt_1\Big)^{1/p} \geqslant$$
$$\geqslant C(d-1,p)(\log|J|)^{(d-1)(1/2-1/p)}\Big(\sum_{B}\int_{\mathbb{R}}\Big(\sum_{A}|a_I h_A(t_1)|^2\Big)^{p/2} dt_1\Big)^{1/p} \tag{28}$$

Now we apply the estimate (24) for $d = 1$ and we continue as

$$S \geqslant C(d-1,p)(\log|J|)^{(d-1)(1/2-1/p)}\Big(\sum_{B}\sum_{A}|a_I|^p\Big)^{1/p} C(1,p)(\log|J|)^{1/2-1/p} =$$
$$= C(d,p)(\log|J|)^{d(1/2-1/p)}\Big(\sum_{I\in J}|a_I|^p\Big)^{1/p}. \tag{29}$$

Due to Proposition 4 we can complete the proof of (24). The inequality (25) follows by duality from (24) for $1 < p \leq 2$.

■

Note that if we work in the setting where all $a_I = 1$, then actually one can show, using Lemma 3, that for $d = 1$, $\|\sum_{I\in J} h_I^1\|$ is just comparable with $|J|^{1/p}$. Therefore we can start the induction from $d = 2$ and thus derive:

Proposition 6 *For $d = 1, 2, \ldots$ and $1 < p \leq 2$ in $L_p(\mathbb{R}^d)$ we have*

$$c(d,p)|J|^{1/p}(\log|J|)^{(\frac{1}{2}-\frac{1}{p})(d-1)} \leqslant \|\sum_{I\in J} h_I^d\| \leqslant C(d,p)|J|^{1/p} \tag{30}$$

for $2 \leq p < \infty$, and

$$c(d,p)|J|^{1/p} \leqslant \|\sum_{I\in J} h_I^d\| \leqslant C(d,p)(\log|J|)^{(\frac{1}{2}-\frac{1}{p})(d-1)}|J|^{1/p} \tag{31}$$

The inequalities (30) and (31) finally lead to the main result for $\mathfrak{h}_p^d$ systems which was conjectured in [32] and proved in [35].

Theorem 9 *Suppose that for $1 < p < \infty$ we consider the system $\mathfrak{h}_p^d$ in $L_p(\mathbb{R}^d)$ space. Then*

$$\mathcal{E}_m \simeq (\log m)^{(d-1)|1/2-1/p|}. \tag{32}$$

Proof. Proposition 6 combined with Theorem 2 shows that $\mathcal{E}_m \leqslant C(\log m)^{|1/2-1/p|(d-1)}$. The estimate from below was proved in [32].

■

Corollary 2 *For $d = 1, 2, \ldots$ and $2 < p < \infty$ in $L_p(\mathbb{R}^d)$ we have*

$$\varphi(\mathfrak{h}_d^p, m) \simeq m^{1/p} \tag{33}$$

$$\psi(\mathfrak{h}_d^p, m) \simeq m^{1/p}(\log m)^{(1/2-1/p)(d-1)}, \tag{34}$$

whereas for $p \leq 2$

$$\varphi(\mathfrak{h}_d^p, m) \simeq m^{1/p}(\log m)^{(1/2-1/p)(d-1)} \tag{35}$$

$$\psi(\mathfrak{h}_d^p, m) \simeq m^{1/p}. \tag{36}$$

Note that Corollary 2 implies that (7) is verified with $s(m) \simeq m(\log m)^{\frac{|p-2|}{2}(d-1)}$. Consequently we deduce from Theorem 3 that for a given $x \in X$ there exist $s(m) \simeq m(\log m)^{\frac{|p-2|}{2}(d-1)}$ coefficients from which we should choose m to find near best m-term approximation. Therefore it seems to be intriguing problem to find the algorithm which provides the near optimal approximation for $\mathfrak{h}_d^p$

5.4 Haar systems in other spaces

One could expect that if there exists the Haar system h_d^p in $L_p(\mathbb{R}^d)$ the same construction should work in rearrangement spaces. We recall that that a rearrangement invariant space is a Banach space $(X, \|\cdot\|)$ whose elements are measurable functions on measure space (Ω, μ) satisfying the following conditions

1. if $x \in X$ and y is a measurable function such that $|y(\omega)| \leqslant |x(\omega)|$, μ-a.e.
2. if $x \in X$ and y has the same distribution as x, i.e. for all $\lambda \in \mathbb{R}$ we have $\mu(x \le \lambda) = \mu(y \le \lambda)$ then $y \in X$ and $\|x\| = \|y\|$.

The main result of [42] states that L_p are the only rearrangement spaces for which the normalized Haar system is greedy.

Theorem 10 *Let X be a rearrangement invariant space on* $[0, 1]^d$. *If a Haar system* h^d *normalized in X is a greedy basis in X, then* $X = L_p[0, 1]^d$ *for some* $1 < p < \infty$.

On the other hand there are examples of bizzare rearrangement spaces (see [20]) for which there exists some greedy basis. However it was conjectured in [42] that for classical different from L_p rearrangement spaces (e.g. Lorentz, Orlicz) this is not possible. We recall that Lorentz $L_{p,q}(\mathbb{R}^d)$ is a Lorentz rearrangement space with the norm $\|x\|_{p,q} = (\int_0^1 x^*(t)^q t^{\frac{q}{p}-1} dt)^{1/q}$, where x^* is non-increasing rearrangement of x (uniquely determined). It was shown in [42] that if for $p \neq q$ there exists greedy basis in $L_{p,q}$, then it has rather unusual properties.

The second interesting class of examples comprise Orlicz spaces. We recall that $L_\varphi(\mathbb{R}^d)$is an Orlicz rearrangement space with the norm $\|x\|_\varphi = \inf\{C : \int \varphi(|x|/C)d\mu \leqslant 1\}$, where φ is some convex, increasing, $\varphi(0) = 0$ function. Such spaces where analyzed recently in [16] where some extension of Theorem 10 has been proved. We say that space has non-trivial Boyd indices if

$$0 < \sup_{0<t<1} \sup_{s>0} \frac{\varphi(st)}{\varphi(s)\varphi(t)} \leqslant \inf_{1<t<\infty} \sup_{s>0} \frac{\varphi(st)}{\varphi(s)\varphi(t)} < \infty.$$

Theorem 11 Let $L_\varphi(\mathbb{R}^d)$ be an Orlicz spaces with non-trivial Boyd indices. An wavelet basis is democratic in $L_\varphi(\mathbb{R}^d)$ if and only if $L_\varphi(\mathbb{R}^d) = L_p(\mathbb{R}^d)$ for some $1 < p < \infty$.

5.5 Functions of bounded variations

Let $\Omega \subset \mathbb{R}^d$ be an open subset. Let us recall that a function $f \in L_1(\Omega)$ has bounded variation if all its distributional derivatives $\frac{\partial f}{\partial x_j}$ are measures of bounded variation. The space of all such functions equipped with the norm

$$\|f\|_{BV} = \sum_{j=1}^{d} \|\frac{\partial f}{\partial x_j}\|$$

is denoted by $BV(\Omega)$. This function space is of importance for the geometric measure theory, calculus of variation, image processing and other areas. Clearly whenever $\|f\|_{BV} < \infty$ then $f \in L_p$, where $p = d/(d-1)$ by the classical embedding theorems. Observe that $BV(\mathbb{R}^d)$ is a non separable space so it cannot have any countable system satisfying (4). On the other hand one may ask whether the Haar system normalized to $BV(\mathbb{R}^d)$ (which we denote by h^d_{BV}) has some stability property, i.e. is quasi greedy on $\overline{\operatorname{span}\{\mathsf{h}^d_{BV}\}} \subset BV(\mathbb{R}^d)$. Generalizing some of the previous results (e.g. [7],[36],[41]) it was proved in [5] that the following holds:

Theorem 12 *Suppose that* $\Phi = (e_i, e_i^*)_{i\in I}$ *is a normalized wavelet basis generated by some compactly supported scaling function (see our discussion in Section about Haar Systems). Then if* $f \in BV(\mathbb{R}^d)$, $d \geq 2$ *the following inequality holds*

$$|\mathcal{G}_m(\Phi, f)|_{BV} \leqslant C(p,d)\|f\|_{BV} \tag{37}$$

for some constant $C(p, d)$ *depending on p, d only.*

This is however not much satisfactory result since $\overline{\operatorname{span}\{\mathsf{h}^d_{BV}\}}$ is not a very natural space. A natural separable space of $BV(\mathbb{R}^d)$ is the Sobolev space $W_1^1(\mathbb{R}^d)$, i.e. the space of all $f \in BV(\mathbb{R}^d)$ such that $\frac{\partial f}{\partial x_j}$ are absolutely continuous measures for $j = 1, 2, \ldots, d$. A natural and interesting problem which rises in this context is to find a smooth wavelet basis which is quasi greedy in $W_1^1(\mathbb{R}^d)$. We remark that $W_1^1(\mathbb{R}^d)$ does not have unconditional basis, so it does not have a greedy basis. On the other hand an immediate consequence of Theorem 13 is that $W_1^1(\mathbb{R}^d)$ has a quasi greedy basis.

6 Examples of greedy and quasi greedy bases

In this section we provide a class of basic examples for natural systems which share the greedy or quasi greedy property.

6.1 Greedy bases

There to basic examples of greedy bases which we often refer to:

1. the natural basis in l_p, $p \geq 1$;
2. the Haar system h^p_d for $L_p(\mathbb{R}^d)$.

It occurs that these natural systems can be useful when combined with some theoretical methods of producing greedy bases.

The first approach is based on the fact that being greedy (or quasi greedy) is an isomorphic property. Therefore whenever $(e_i)_{i\in I}$ is a greedy system in Banach space X and $T : X \to Y$ is a linear isomorphism, then $(T(e_i))_{i\in I}$ is a greedy system in Y. We mention two practically useful examples of this remark:

1. Consider L_p, $1 < p < \infty$ space. If $\mathcal{B}$ is a good wavelet basis (cf. [37] Theorem 8.13) normalized to L_p then it is equivalent to the Haar system h_p. Thus such all systems are greedy.
2. It is known (cf. [37], Chapter 9) that good wavelet bases in Besov space $\mathcal{B}^p_{\alpha,p}$ when properly normalized are equivalent to the unit vector basis in l_p, thus greedy for $1 \le p < \infty$.

The second approach is to use the dual basis (see Remark 3). In particular (see Corollary 1) we have shown that dual basis of h^d_p in L_p, $1 < p < \infty$ is greedy in L_q, were $1/p+1/q =1$. However one has to be careful when using Remark 3, since without the additional assumption that $\varphi(m) \simeq m^\alpha$ for some $0 < \alpha < 1$ it may be not true that dual basis is greedy in its linear closure. The simplest example of such a case may be constructed for the system h^1_1 in H_1 (the space of integrable functions with the norm given by (23)). The dual system is the system h^∞_1 considered in the space *VMO*. It was proved in [29] that $\mathcal{E}_m(\mathsf{h}^\infty_1) \simeq \sqrt{\log m}$ in the space *VMO*, so we have a natural example of a greedy system whose dual is not greedy. Actually one can show that the space *VMO* does not have any greedy system.

Now we turn to discuss other examples of greedy bases in L_p. The simplest case is of $p = 2$, i.e. when we consider Hilbert space. Clearly every orthonormal basis, and more generally, every Riesz basis is greedy in a Hilbert space, since they are the only unconditional systems in L_2. This easily follows from Proposition 4.

In L_p for $1 < p < \infty$, $p \neq 2$, the situation is not as simple. Except wavelet bases it is a hard question to provide other examples of greedy bases. We state below the Kamont [23] construction of a generalized Haar system in [0, 1]:

The first function is $1_{[0,1]}$. Next we divide [0, 1] into two subintervals I_l and I_r (nontrivial but generally not equal) and the next function is of the form $a1_{I_l} + b1_{I_r}$ and is orthogonal to the previous function. We repeat this process on each of intervals I_l and I_r and continue in this manner.

If we make sure that the lengths of subintervals tend to zero the system will span $L_p[0, 1]$ for $1 \le p < \infty$. One of the main results of [23] states that each generalized Haar system (normalized in $L_p[0, 1]$) is equivalent to a subsequence of h^p_1, so is greedy.

An example of a basis in L_p for $p > 2$ which is greedy and not equivalent to a subsequence of the Haar system h^p_1 was given in [35]. It follows from Corollary 1 that such an example exists also for $1 < p < 2$.

6.2 Quasi greedy bases

As we have mentioned in Remark 4 all unconditional system are quasi greedy. This observation however shows that unfortunately the greedy approximation can be very inefficient when used in this case. For example for the natural basis in $l_1 \oplus c_0$ which is unconditional we have $\mathcal{E}_m \simeq m$.

Obviously to show other examples one has to investigate spaces without unconditional bases. Some examples were given in [26] but the general treatment was presented in [35]

and recently generalized in [10]. In both papers the approach is quite abstract and uses the existence of good complemented subspace. A very general result (Corollary 7.3 from [10]) is as follows.

Theorem 13 *If X has a basis and contains a complemented subspace S with a symmetric basis, where S is not isomorphic to c_0, then X has a quasi greedy basis.*

We recall that X is a $\mathcal{L}_\infty$ space if there exists $\lambda \geq 1$ and a directed net Y_α of finite dimensional subspaces of X, where each Y_α is λ-isomorphic to an l_∞^n space such that $X = \overline{\bigcup_\alpha Y_\alpha}$. This class includes every complemented subspace of $C(K)$. In [10] (Corollary 8.6) there was proved a characterization of $\mathcal{L}_\infty$ spaces which admits a greedy basis.

Theorem 14 *The space c_0 is the unique infinite dimensional $\mathcal{L}_\infty$ space, up to isomorphism, with a quasi greedy basis. Moreover c_0 has a unique quasi greedy basis up to equivalence.*

Therefore neither $C[0, 1]$ nor the disc algebra A (which trivially shares $\mathcal{L}_\infty$-property) do not have any quasi greedy basis.

Since clearly $L_1[0, 1]$ does contain complemented symmetric subspace (which is necessarily isomorphic to l_1, see e.g. Proposition 5.6.3 in [1]) we obtain from Theorem 13 that $L_1[0, 1]$ has a quasi greedy basis. Since it is known that $L_1[0, 1]$ does not have unconditional (in particular greedy) this is a good kind of basis. On the other hand it is none of the classical systems. For example the Haar basis (and other wavelet bases) are not quasi greedy in $L_1(\mathbb{R})$. To see it note that for $I_n = [0, 2^{-n}]$, $n = 1, 2, \ldots, N$, we have $\| \sum_{n=1}^N H_{I_n}^1 \|_1 \simeq \text{const}$, while $\| \sum_{n=1}^N (-1)^n H_{I_n}^1 \|_1 \simeq \log N$, so (20) is violated.

7. Basic sequences

We call a sequence $(e_i)_{i \in I}$ in a Banach space X a basic sequence if it is a basis for $\overline{\text{span}\{e_i, i \in I\}}$. The unconditional sequence problem is that we ask whether or not in any infinite dimensional Banach space there exists a quasi greedy sequence. The problem was regarded as perhaps the single most important problem in the approximation theory. Eventually a counterexample was found by Gowers and Maurey in [18]. The construction which is extremely involved has led to a variety of other applications (see e.g. [25], [17], [19]). However there is still open a bit weaker version of the problem:

Conjecture 1 *In every infinite Banach space X there exists a quasi greedy basic sequence.*

Some partial positive results are given in [13] and [3]. Roughly speaking there is shown in thees papers that whenever our space X is far from c_0 (in a certain sense) then there exists quasi a greedy sequence.

8. Greedy bases are best in *Lp*

In this section we assume for simplicity that we work with Schauder bases. From recent works [9] and [36] it became apparent that greedy basis in L_p is a natural substitute for an orthonormal basis in a Hilbert space. Let us explain brifley what does it mean.

8.1 Comparing bases

In [9] the following general problem is discussed. Let $\mathcal{F}$ be a certain Banach space continuously embedded into L_p and let $\mathcal{F}_0$ be its unit ball. For a given basis $B = (e_i)_{i=1}^\infty$ in L_p we introduce the quantities

$$\sigma_m(B,\mathcal{F}) = \sup_{x\in\mathcal{F}_0} \sigma_m(B,x), \quad m \geq 0$$

We are looking for a basis B which gives the best order of decay $\sigma_m(B,\mathcal{F})$. It is natural to expect that the best basis has to have close connection with the class $\mathcal{F}$. We shall say that $\mathcal{F} \subset X$ is aligned with B if for each $\sum_{i=1}^{\infty} a_i e_i \in \mathcal{F}$ and $|b_i| \leqslant |a_i|$ we have that $\sum_{i=1}^{\infty} b_i e_i \in \mathcal{F}$. The following was proved in [9] (Theorem 4.2).

Theorem 15 *Let B be a greedy basis for X with the property $\varphi(m) \approx m^{\frac{1}{p}}$, for some $p > 1$. Assume that $\mathcal{F}$ is aligned with B and for some $\alpha \in \mathbb{R}$, $\beta > 0$, we have*

$$\limsup_{m\to\infty} (\log m)^{\alpha} m^{\beta} \sigma_m(B,\mathcal{F}) > 0,$$

Then for any unconditional basis B' we have

$$\limsup_{m\to\infty} (\log m)^{\alpha+\beta} m^{\beta} \sigma_m(B',\mathcal{F}) > 0.$$

The theorem implies that in some sense a greedy basis aligned with $\mathcal{F} \subset X$ is the best among all unconditional bases. Certainly it seems that if they are best in the class of fine bases, greedy bases should be best among all the possible bases. Unfortunately all the admissible methods require the second basis to be unconditional.

The first paper in this direction was by Kashin [24] who proved that if X is L_2 space then for each orthogonal basis B we have $\sigma_m(B, \mathrm{Lip}_\alpha) \geqslant C(\alpha) m^{-\alpha}$, where $0 < \alpha \leq 1$ and Lip_α is a class of Lipschitz functions according to the metric $d(s,t) = \|s-t\|^{\alpha}$. Next step was due to Donoho (see [14], [15]) who proved under the assumption $X = L_2$ that if $\mathcal{F}$ is aligned with an orthogonal basis B, such that $\lim \sup_{m\to\infty} m^{\beta} \sigma_m(B,\mathcal{F}) > 0$, for some $\beta > 0$, then for $\gamma > \beta$ we have $\lim \sup_{m\to\infty} m^{\gamma} \sigma_m(B',\mathcal{F}) > 0$. Then by DeVore, Temlyakov and Petrova [9] the result was extended from L_2 spaces to L_p, yet with a loss of some logarithmic factor.

Theorem 15 has been recently improved in [6]. We first formulate the following condition

$$\sum_{k=1}^{n} \frac{2^k}{\varphi(2^k)} \leqslant A \frac{2^n}{\varphi(2^n)}, \quad \text{for } n \geqslant 1. \tag{38}$$

Clearly if $\varphi(m) \approx m^{\frac{1}{p}}$, $p > 1$ then (38) is verified. The condition says that φ verifies a kind of Δ_2 condition in ∞ (i.e. it cannot be linear in ∞).

In what follows, we will need some of the basic concepts of the Banach space theory. First let us recall the definition of type and cotype. Namely, if $(\varepsilon_i)_{i=1}^{\infty}$ is a sequence of independent Rademacher variables, we say that X has type 2 if there exists a universal constant C_1 such that

$$\mathbf{E}\|\sum_{i=1}^{n} \varepsilon_i x_i\| \leqslant C_1 (\sum_{i=1}^{n} \|x_i\|^2)^{\frac{1}{2}}, \quad \text{for } n \geqslant 1,\ x_i \in X,$$

and X is of cotype 2 if there exists a universal constant C_2 such that

$$\mathbf{E}\|\sum_{i=1}^{n}\varepsilon_i x_i\| \geqslant C_2(\sum_{i=1}^{n}\|x_i\|^2)^{\frac{1}{2}}, \quad \text{for } n \geqslant 1,\ x_i \in X.$$

In particular the L_p spaces have type 2 if $p \geq 2$ and cotype 2 if $1 \leq p \leq 2$. For more comprehensive information see for example, [38], Chapter III A. Since we work with bases, we need a definition of type and cotype 2 in these settings. A basis B is called Riesz basis if

$$\|\sum_{i=1}^{\infty}a_i e_i\| \leqslant A_1(\sum_{i=1}^{\infty}|a_i|^2)^{\frac{1}{2}},$$

and Bessel basis, if

$$(\sum_{i=1}^{\infty}|a_i|^2)^{\frac{1}{2}} \leqslant A_2\|\sum_{i=1}^{\infty}a_i e_i\|,$$

where A_1, A_2 are universal constants. Obviously if X has type or cotype 2 then B is Riesz or Bessel basis respectively.

We can formulate the main result of the section.

Theorem 16 *Let X be a Banach space and let B be a greedy and Riesz basis (or greedy and Bessel basis) which satisfies (38) (the Δ_2 condition). Suppose that K is aligned with B and that B' is an unconditional basis for X. There exist absolute constants $C > 0$ and $\tau \in \mathbb{N}$ such that*

$$\sigma_{2^n}(B,K) \leqslant C\sum_{k=n-\tau}^{\infty}\sigma_{2^k}(B',K), \quad \text{for } n \in \mathbb{N},\ n \geqslant \tau.$$

It is possible to prove a weaker version of Theorem 16 in which we do not assume B to be Riesz or Bessel basis and which exactly implies Theorem 15. However the main class of examples consists of L_p spaces $\varphi(m) \approx m^{\frac{1}{p}}$ for all greedy bases in L_p) and in this setting we can benefit from the fact that L_p spaces are of type or cotype 2 (each unconditional basis B is Riesz or Bessel). Thus we can apply Theorem 16 for L_p spaces and consequently remove the additional logarithmic factor in Theorem 15.

Corollary 3 (of Theorem 16) *Suppose that X is L_p space, $p > 1$ and $\mathcal{F}$ is aligned with a greedy basis B. If B verifies*

$$\limsup_{m\to\infty}(\log m)^{\alpha}m^{\beta}\sigma_m(B,\mathcal{F}) > 0, \quad \text{where } \beta > 0,\ \alpha \in \mathbb{R},$$

then for each unconditional basis B' in X the following inequality holds

$$\limsup_{n\to\infty}(\log m)^{\alpha}m^{\beta}\sigma_m(B',\mathcal{F}) > 0.$$

8.2 Tools

In this section, we derive some preliminary results that we shall need later. The following lemma holds.

Lemma 4 *If B is unconditional basis and verifies (38) (the Δ_2 condition), then the following inequality holds:*

$$ADK\|\sum_{i=1}^{2^n} a_i e_i\| \geqslant 2^{-n}\varphi(2^n)\sum_{i=1}^{2^n}|a_i|.$$

Proof. We can assume that $|a_i| \geqslant |a_{i+1}|$, an thus since B is unconditional, we have $\|\sum_{i=1}^{2^n} a_i e_i\| \geqslant K^{-1}\|\sum_{i=1}^{2^k}|a_{2^k}|e_i\| = D^{-1}K^{-1}\varphi(2^k)|a_{2^k}|$, for $k = 0, 1, \ldots, n$. Hence by (38) we obtain

$$\sum_{i=1}^{2^n}|a_i| \leqslant \sum_{k=0}^{n} 2^k|a_{2^k}| \leqslant DK\sum_{k=0}^{n} 2^k\varphi(2^k)^{-1}\|\sum_{i=1}^{2^n} a_i e_i\| \leqslant ADK2^n\varphi(2^n)^{-1}\|\sum_{i=1}^{2^n} a_i e_i\|.$$

■

Our main class of examples consists of L_p spaces, $p > 1$ for which the assumptions in Theorem 16 are clearly verified. In order to use Theorem 16 for much larger classes of Banach spaces, we need a simple characterization whether a greedy basis B is Riesz or Bessel in terms of $\varphi(n)$ numbers.

Lemma 5 *Suppose B is a greedy basis (democratic and unconditional). If* $\varphi(n)$ *satisfies*

$$\sum_{n=0}^{\infty}\varphi(2^n)^{-2n}2^n \leqslant L_2, \tag{39}$$

then B is Bessel basis and if

$$\sum_{n=0}^{\infty}\varphi(2^n)^2 2^{-n} \leqslant L_1, \tag{40}$$

then B is Riesz basis.

Proof. We can assume that $|a_i| \geqslant |a_{i+1}|$. The unconditionality of B implies

$$\|\sum_{i=1}^{\infty} a_i e_i\| \geqslant K^{-1}\|\sum_{i=1}^{2^k}|a_{2^k}|e_i\| = D^{-1}K^{-1}|a_{2^k}|\varphi(2^k),$$

for $k = 0, 1, 2, \ldots$. Hence by (39)

$$\sum_{i=1}^{\infty}|a_i|^2 \leqslant \sum_{k=0}^{\infty}|a_{2^k}|^2 2^k \leqslant$$

$$\leqslant D^2K^2\sum_{k=0}^{\infty} 2^k\varphi(2^k)^{-2}\|\sum_{i=1}^{\infty} a_i e_i\|^2 \leqslant L_2\, D^2K^2\|\sum_{i=1}^{\infty} a_i e_i\|^2.$$

Thus $A_2\|\sum_{i=1}^{\infty} a_i e_i\| \geqslant (\sum_{i=1}^{\infty}|a_i|^2)^{\frac{1}{2}}$, where $A_2 = \sqrt{L_2}DK$. Similarly assuming that $|a_i| \geqslant |a_{i+1}|$ and the fact that B is a democratic basis, we have that

$$\|\sum_{i=1}^{\infty} a_i e_i\| \leqslant K\sum_{k=0}^{\infty}\|\sum_{i=2^k}^{2^{k+1}-1}|a_{2^k}|e_i\| \leqslant DK\sum_{k=0}^{\infty}|a_{2^k}|\varphi(2^k).$$

Thus using the Schwartz inequality and (40) we get

$$\|\sum_{i=1}^{\infty} a_i e_i\| \leqslant DK(\sum_{k=0}^{\infty} |a_{2^k}|^2 2^k)^{\frac{1}{2}}(\sum_{k=0}^{\infty} \varphi(2^k)2^{-k})^{\frac{1}{2}} \leqslant \sqrt{2L_1}DK(\sum_{i=1}^{\infty} |a_i|^2)^{\frac{1}{2}},$$

where we applied the following inequali

$$\sum_{k=0}^{\infty} |a_{2^k}|^2 2^k \leqslant |a_1|^2 + 2\sum_{k=1}^{\infty} \sum_{i=2^{k-1}+1}^{2^k} |a_i|^2 \leqslant 2\sum_{i=1}^{\infty} |a_i|^2.$$

Consequently $\|\sum_{i=1}^{\infty} a_i e_i\| \leqslant A_1(\sum_{i=1}^{\infty} |a_i|^2)^{\frac{1}{2}}$, where $A_1 = \sqrt{2L_1}DK$.

■

Remark 6 *If we assume only that* $\sup_{n\geqslant 0} \varphi(2^n)^{-2}2^n \leqslant L_2'$ *or* $\sup_{n\geqslant 0} \varphi(2^n)2^{-n} \leqslant L_1'$, *then mimicking the proof of Lemma 5 we obtain respectively*

$$(\sum_{i=1}^{2^n} |a_i|^2)^{\frac{1}{2}} \leqslant A_2' n^{\frac{1}{2}} \|\sum_{i=1}^{2^n} a_i e_i\|, \quad or \quad \|\sum_{i=1}^{2^n} a_i e_i\| \leqslant A_1' n^{\frac{1}{2}} (\sum_{i=1}^{2^n} |a_i|^2)^{\frac{1}{2}}.$$

Remark 7 *If* $\varphi(m) \approx m^{\frac{1}{p}}$, *where* $1 < p < 2$ *or* $p > 2$ *then respectively (39) or (40) holds true. Thus each greedy basis B such that* $\varphi(m) \approx m^{\frac{1}{p}}$, *where* $p > 1$, $p \neq 2$, *is Bessel or Riesz basis. Furthermore for all* $p > 1$, *if* $\varphi(m) \approx m^{\frac{1}{p}}$ *the condition from Remark 6 is verified.*

Lemma 6 *Let* $(\varepsilon_i)_{i=1}^{\infty}$ *be a sequence of independent Rademacher variables and* a_i, $a_{i,j} \in \mathbb{R}$, $i, j \in \mathbb{N}$. *We have*

$$\mathbf{E}|\sum_{i=1}^{\infty} a_i \varepsilon_i|^2 = \sum_{i=1}^{\infty} |a_i|^2, \quad \mathbf{E}|\sum_{i,j=1}^{\infty} a_{i,j}\varepsilon_i\varepsilon_j|^2 \leqslant 2\sum_{i,j=1}^{\infty} |a_{i,j}|^2.$$

Proof. The first equality is classical and easy so we only prove the second one. If $\sum_{i,j=1}^{\infty} |a_{i,j}|^2 = \infty$ then there is nothing to prove, otherwise we have

$$\mathbf{E}|\sum_{i,j=1}^{\infty} a_{i,j}\varepsilon_i\varepsilon_j|^2 = \sum_{i,j=1}^{\infty} |a_{i,j}|^2 + \sum_{i,j=1}^{\infty} a_{i,j}a_{j,i} \leqslant 2\sum_{i,j=1}^{\infty} |a_{i,j}|^2,$$

where we have used the inequality $2ab \leqslant a^2 + b^2$.

■

Lemma 7 *Let* $B = (e_i), B' = (e_i')$ *are respectively greedy and unconditional basis for X. Let* $e_j' = \sum_{i=1}^{\infty} d_{j,i}e_i$ *and* $e_i = \sum_{j=1}^{\infty} c_{i,j}e_j'$, *and let K' be the unconditionality constant for B'. If B is Riesz or Bessel basis, then*

$$\sum_{i=1}^{2^n}\sum_{l=1}^{2^n}\sum_{j=1}^{\infty} |c_{i,j}|^2|d_{j,l}|^2 \leqslant c2^n,$$

where c is a certain constant (not depending on n).

Proof. Fix $i \geq 1$. By the unconditionality of B' and B and the Bessel property of B we have

$$1 = \|e_i\| = \|\sum_{j=1}^{\infty} c_{i,j} e'_j\| \geqslant (K')^{-1} \|\sum_{j=1}^{\infty} \varepsilon_j c_{i,j} e'_j\| \geqslant$$

$$\geqslant (KK')^{-1} \|\sum_{l=1}^{2^n} (\sum_{j=1}^{\infty} c_{i,j} d_{j,l} \varepsilon_j) e_l\| \geqslant (A_2 KK')^{-1} (\sum_{l=1}^{2^n} |\sum_{j=1}^{\infty} c_{i,j} d_{j,l} \varepsilon_j|^2)^{\frac{1}{2}}$$

Thus due to Lemma 6 we obtain

$$(A_2 KK')^2 \geqslant \mathbf{E} \sum_{l=1}^{2^n} |\sum_{j=1}^{\infty} c_{i,j} d_{j,l} \varepsilon_j|^2 = \sum_{l=1}^{2^n} \sum_{j=1}^{\infty} |c_{i,j}|^2 |d_{j,l}|^2,$$

and hence $\sum_{i=1}^{2^n} \sum_{l=1}^{2^n} \sum_{j=1}^{\infty} |c_{i,j}|^2 |d_{j,l}|^2 \leqslant (A_2 KK')^2 2^n$.

Now fix $l \geq 1$. Due to the Riesz property of B and the unconditionality of B' and B we obtain

$$A_1 (\sum_{i=1}^{2^n} |a_i|^2)^{\frac{1}{2}} \geqslant \|\sum_{i=1}^{2^n} a_i e_i\| = \|\sum_{j=1}^{\infty} (\sum_{i=1}^{2^n} a_i c_{i,j}) e'_j\| \geqslant (K')^{-1} \|\sum_{j=1}^{\infty} (\sum_{i=1}^{2^n} a_i c_{i,j} \varepsilon_j) e'_j\| =$$

$$= (K')^{-1} \|\sum_{l=1}^{\infty} (\sum_{i=1}^{2^n} a_i (\sum_{j=1}^{\infty} c_{i,j} d_{j,l} \varepsilon_j)) e_l\| \geqslant (KK')^{-1} |\sum_{i=1}^{2^n} a_i (\sum_{j=1}^{\infty} c_{i,j} d_{j,l} \varepsilon_j)|.$$

If we take $a_i = \sum_{j=1}^{\infty} c_{i,j} d_{j,l} \varepsilon_j$, then by Lemma 6 we get

$$(A_1 KK')^2 \geqslant \mathbf{E} \sum_{i=1}^{2^n} |\sum_{j=1}^{\infty} c_{i,j} d_{j,l} \varepsilon_j|^2 = \sum_{i=1}^{2^n} \sum_{j=1}^{\infty} |c_{i,j}|^2 |d_{j,l}|^2.$$

It proves that $\sum_{i=1}^{2^n} \sum_{l=1}^{2^n} \sum_{j=1}^{\infty} |c_{i,j}|^2 |d_{j,l}|^2 \leqslant (A_1 KK')^2 2^n$.

■

Remark 8 *If we assume only that* $\sup_{n \geqslant 0} \varphi(2^n)^{-2} 2^n \leqslant L'_2$ *or* $\sup_{n \geqslant 0} \varphi(2^n)^2 2^{-n} \leqslant L'_1$ *then applying Remark 6 in the above proof (instead of Riesz or Bessel property) we obtain*

$$\sum_{i=1}^{2^n} \sum_{l=1}^{2^n} \sum_{j=1}^{\infty} |c_{i,j}|^2 |d_{j,l}|^2 \leqslant cn2^n,$$

for some universal constant $c < \infty$.

8.3 Proof of main result

Proof of Theorem 1. Fix $n \geq 1$, $\delta > 0$. First we assume that $\sigma_{2^n}(B, \mathcal{F}) < \infty$. The definition of $\sigma_{2^n}(B, \mathcal{F})$ implies that there exists $x \in \mathcal{F}_0$ such that

$$\sigma_{2^n}(B, \mathcal{F}) \leqslant (1+\delta) \sigma_{2^n}(B, x) \leqslant (1+\delta) \|x - \mathcal{G}_{2^n}(B, x)\|.$$

Observe that $\|x-\mathcal{G}_{2^n}(B,x)\|$ does not depend on the basis B renumeration, so we can and we will assume that $|e_i^*(x)| \geqslant |e_{i+1}^*(x)|$ for $i=1,2,...$ and $\mathcal{G}_{2^n}(B,x)=\sum_{i=1}^{2^n} e_i^*(x)e_i$. Since $\mathcal{F}$ is aligned to B, whenever $\sum_{i=1}^{\infty} a_i e_i \in \mathcal{F}_0$ and $|b_i| \leqslant |a_i|$, we have $\sum_{i=1}^{\infty} b_i e_i \in u\mathcal{F}_0$, where u is a universal constant. Consequently

$$u^{-1}|e_{2^k}^*(x)|\sum_{i=1}^{2^k}\varepsilon_i e_i \in \mathcal{F}_0, \quad \varepsilon_i \in \{-1,1\}.$$

It proves that denoting $y_k := e_{2^k}^*(x)$, the cube

$$\mathcal{F}_{n,k} := \{u^{-1}y_k \sum_{i=1}^{2^k}\varepsilon_i e_i, \quad \varepsilon_i \in \{-1,1\}\}$$

is contained in $\mathcal{F}_0$. Applying the triangle inequality we obtain

$$\|x-\mathcal{G}_{2^n}(B,x)\| = \|\sum_{i=2^n+1}^{\infty} e_i^*(x)e_i\| \leqslant \sum_{k=n}^{\infty}\|\sum_{i=2^k+1}^{2^{k+1}} e_i^*(x)e_i\|.$$

Thus due to the unconditionality we get

$$\|\sum_{i=2^k+1}^{2^{k+1}} e_i^*(x)e_i\| \leqslant K\|\sum_{i=2^k+1}^{2^{k+1}} |e_{2^k}^*(x)|e_i\| \leqslant DK|e_{2^k}^*(x)|\varphi(2^k),$$

hence

$$\begin{aligned}\sigma_{2^n}(B,\mathcal{F}) &\leqslant (1+\delta)\|x-\mathcal{G}_{2^n}(B,x)\| \leqslant (1+\delta)DK\sum_{k=n}^{\infty}|e_{2^k}^*(x)|\varphi(2^k) = \\ &= (1+\delta)DK\sum_{k=n}^{\infty}|y_k|\varphi(2^k).\end{aligned} \tag{41}$$

Fix $k \in \mathbb{N}$. Let $(\varepsilon_i)_{i=1}^{\infty}$ be a sequence of independent Rademacher variables. For simplicity we denote $d_j^k = (d_{j,1},...,d_{j,2^k})$, $c_j^k = (c_{1,j},...,c_{2^k,j})$ and consequently we have

$$\langle c_j^k,\varepsilon\rangle = \sum_{i=1}^{2^k}\varepsilon_i c_{i,j}, \quad \langle d_j^k,\varepsilon\rangle = \sum_{i=1}^{2^k}\varepsilon_i d_{j,i}.$$

Observe that $\hat{x} = u^{-1}y_k\sum_{i=1}^{2^k}\varepsilon_i e_i \in \mathcal{F}_{n,k} \subset \mathcal{F}_0$ and thus $\sigma_{2^m}(B',\mathcal{F}) \geqslant \sigma_{2^m}(B',\hat{x})$ for m = 0, 1, 2, …. By definition $\sigma_{2^m}(B',\hat{x}) = \inf_{S\in\Sigma_{2^m}(B')}\|u^{-1}y_k\sum_{i=1}^{2^k}\varepsilon_i e_i - S\|$ and therefore

$$\sigma_{2^m}(B',\mathcal{F}) \geqslant \inf_{S\in\Sigma_{2^m}(B')}\|u^{-1}y_k\sum_{i=1}^{2^k}\varepsilon_i e_i - S\| = \inf_{|\Lambda|=2^m}\inf_{a_j^{\Lambda}\in\mathbb{R}}\|u^{-1}y_k\sum_{j=1}^{\infty}\langle c_j^k,\varepsilon\rangle e_j' - \sum_{j\in\Lambda}a_j^{\Lambda}e_j'\|.$$

Furthermore, the unconditionality implies

$$\|u^{-1}y_k\sum_{j=1}^{\infty}\langle c_j^k,\varepsilon\rangle e_j' - \sum_{j\in\Lambda}a_j^{\Lambda}e_j'\| \geqslant (K')^{-1}\|u^{-1}y_k\sum_{j\notin\Lambda}\langle c_j^k,\varepsilon\rangle e_j'\| =$$

$$= (uK')^{-1}|y_k|\|\sum_{j=1}^{\infty}\langle c_j^k,\varepsilon\rangle e_j' - \sum_{j\in\Lambda}\langle c_j^k,\varepsilon\rangle e_j'\| = (uK')^{-1}|y_k|\|\sum_{i=1}^{2^k}\varepsilon_i e_i - \sum_{j\in\Lambda}\langle c_j^k,\varepsilon\rangle e_j'\|,$$

thus

$$\sigma_{2^m}(B',\mathcal{F}) \geqslant (uK')^{-1}|y_k| \inf_{|\Lambda|=2^m}\|\sum_{i=1}^{2^k}\varepsilon_i e_i - \sum_{j\in\Lambda}\langle c_j^k,\varepsilon\rangle e_j'\|. \tag{42}$$

Again using the unconditionality and $e_j' = \sum_{i=1}^{\infty} d_{j,i}e_i$ we get

$$\|\sum_{i=1}^{2^k}\varepsilon_i e_i - \sum_{j\in\Lambda}\langle c_j^k,\varepsilon\rangle e_j'\| = \|\sum_{i=1}^{2^k}(\varepsilon_i - \sum_{j\in\Lambda}d_{j,i}\langle c_j^k,\varepsilon\rangle)e_i - \sum_{i=2^k+1}^{\infty}\sum_{j\in\Lambda}d_{j,i}\langle c_j^k,\varepsilon\rangle e_i\| \geqslant$$

$$\geqslant K^{-1}\|\sum_{i=1}^{2^k}(\varepsilon_i - \sum_{j\in\Lambda}d_{j,i}\langle c_j^k,\varepsilon\rangle)e_i\|. \tag{43}$$

Now we apply Lemma 1 in the case of $a_i = \varepsilon_i - \sum_{j\in\Lambda}d_{j,i}\langle c_j^k,\varepsilon\rangle$ and derive that

$$\|\sum_{i=1}^{2^k}\varepsilon_i e_i - \sum_{j\in\Lambda}\langle c_j^k,\varepsilon\rangle e_j'\| \geqslant A^{-1}K^{-2}2^{-k}\varphi(2^k)\sum_{i=1}^{2^k}|\varepsilon_i - \sum_{j\in\Lambda}d_{j,i}\langle c_j^k,\varepsilon\rangle|.$$

Observe that $|\varepsilon_i - \sum_{j\in\Lambda}d_{j,i}\langle c_j^k,\varepsilon\rangle| \geqslant 1 - \varepsilon_i\sum_{j\in\Lambda}d_{j,i}\langle c_j^k,\varepsilon\rangle$, hence by (42) and (43) we have

$$uAK^2K'\sigma_{2^m}(B',\mathcal{F}) \geqslant |y_k|\varphi(2^k)(1 - 2^{-k}\sup_{|\Lambda|=2^m}\sum_{i=1}^{2^k}\sum_{j\in\Lambda}\varepsilon_i d_{j,i}\langle c_j^k,\varepsilon\rangle) =$$

$$= |y_k|\varphi(2^k)(1 - 2^{-k}\sup_{|\Lambda|=2^m}\sum_{j\in\Lambda}\langle d_j^k,\varepsilon\rangle\langle c_j^k,\varepsilon\rangle).$$

The Schwartz inequality gives

$$\sum_{j\in\Lambda}\langle d_j^k,\varepsilon\rangle\langle c_j^k,\varepsilon\rangle \leqslant |\Lambda|^{\frac{1}{2}}(\sum_{j\in\Lambda}|\langle d_j^k,\varepsilon\rangle|^2|\langle c_j^k,\varepsilon\rangle|^2)^{\frac{1}{2}} \leqslant 2^{\frac{m}{2}}(\sum_{j=1}^{\infty}|\langle d_j^k,\varepsilon\rangle|^2|\langle c_j^k,\varepsilon\rangle|^2)^{\frac{1}{2}}.$$

Applying the inequality $\mathbf{E}|Y| \leqslant (\mathbf{E}|Y|^2)^{1/2}$, Lemma 6 and Lemma 7, we get

$$\mathbf{E}(\sum_{j=1}^{\infty}|\langle d_j^k,\varepsilon\rangle|^2|\langle c_j^k,\varepsilon\rangle|^2)^{\frac{1}{2}}\leqslant(\sum_{j=1}^{\infty}\mathbf{E}|\langle d_j^k,\varepsilon\rangle|^2|\langle c_j^k,\varepsilon\rangle|^2)^{\frac{1}{2}}=$$

$$=(\sum_{j=1}^{\infty}\mathbf{E}|\sum_{i,l=1}^{2^k}c_{i,j}d_{j,l}\varepsilon_i\varepsilon_l|^2)^{1/2}\leqslant(2\sum_{i=1}^{2^k}\sum_{l=1}^{2^k}\sum_{j=1}^{\infty}|c_{i,j}|^2|d_{j,l}|^2)^{\frac{1}{2}}\leqslant\sqrt{2c}2^{\frac{k}{2}}.$$

Thus

$$uAK^2K'\sigma_{2^m}(B',\mathcal{F})\geqslant|y_k|\varphi(2^k)(1-\sqrt{2c}2^{-k}2^{\frac{m}{2}}2^{\frac{k}{2}})=|y_k|\varphi(2^k)(1-\sqrt{2c}2^{\frac{m-k}{2}}).$$

Taking $m = k - \tau$, and using (41) we get

$$(1+\delta)uADK^3K'\sum_{k=n}^{\infty}\sigma_{2^{k-\tau}}(B',\mathcal{F})\geqslant(1+\delta)(1-\sqrt{2c}2^{-\frac{\tau}{2}})DK\sum_{k=n}^{\infty}|y_k|\varphi(2^k)\geqslant$$

$$\geqslant(1-\sqrt{2c}2^{-\frac{\tau}{2}})\sigma_{2^n}(B,\mathcal{F}).$$

We can find suitable τ such that $1>\sqrt{2c}2^{-\frac{\tau}{2}}$. Since $\delta > 0$ is arbitrary we obtain

$$\sigma_{2^n}(B,\mathcal{F})\leqslant C\sum_{k=n-\tau}^{\infty}\sigma_{2^k}(B',\mathcal{F}),$$

where $C=uADK^3K'(1-\sqrt{2c}2^{-\frac{\tau}{2}})^{-1}$. This completes the proof when $\sigma_{2^n}(B,\mathcal{F})<\infty$. In the case of $\sigma_{2^n}(B,\mathcal{F})=\infty$, given $M < 1$ we can find x such that $\|x-\mathcal{G}_n(B,x)\|\geqslant M$. Mimicking the previous argument we prove that

$$M\leqslant C\sum_{k=n-\tau}^{\infty}\sigma_{2^k}(B',\mathcal{F}).$$

Since M is arbitrary, it completes the proof in the case of $\sigma_{2^n}(B,\mathcal{F})=\infty$.

■

Proof of Corollary 1. Obviously if X is L_p space, then B is Riesz (for $p \geq 2$) or Bessel (for $1 \leq p \leq 2$) basis. Moreover since $\varphi(m)\approx m^{\frac{1}{p}}$ (see Section 2 in [9]), the basis B satisfies the Δ_2 condition and thus we can apply Theorem 16. Assume that $\limsup_{n\to\infty}n^{\alpha}2^{\beta n}\sigma_{2^n}(B',K)=0$. That means for every $\varepsilon > 0$ there exists $N(\varepsilon) \in \mathbb{N}$ such that $\sigma_{2^k}(B',K)\leqslant\varepsilon k^{-\alpha}2^{-\beta k}$, for $k\geqslant N(\varepsilon)$. Thus for $n > N(\varepsilon) + \tau$ we have

$$\sum_{k=n-\tau}^{\infty}\sigma_{2^k}(B',K)\leqslant\varepsilon\sum_{k=n-\tau}^{\infty}k^{-\alpha}2^{-\beta k}.$$

Observe that $\sum_{k=n-\tau}^{\infty}k^{-\alpha}2^{-\beta k}\leqslant cn^{-\alpha}2^{-\beta n}$, where c is a universal constant (which depends on α, β, τ only). Theorem 16 implies that

$$\sigma_l(B,\mathcal{F})\leqslant\sigma_{2^n}(B,\mathcal{F})\leqslant\varepsilon cCn^{-\alpha}2^{-\beta n},\ \text{ for } n\geqslant N(\varepsilon)+\tau,\ l\geqslant 2^n,$$

which is impossible since $\limsup_{n\to\infty}(\log_2 n)^{\alpha} n^{\beta}\sigma_n(B,\mathcal{F})>0.$

■

Remark 9 *Using Remark 8 instead of Lemma 7 in the proof of Theorem 16 and then mimicking the argument from Corollary 1 (but for general Banach spaces and greedy basis B such that* $\varphi(m)\approx m^{\frac{1}{p}}$ *) we get Theorem 15.*

Remark 10 *Results of [9] do not exclude the possibility that for some other unconditional basis B we have* $\lim_{m\to\infty}\sigma_m(B,\mathcal{F})/\sigma_m(B',\mathcal{F})=0.$ *It was conjectured in [40] that it is impossible.*

9. References

[1] Albiac, F. and Kalton, N.J. (2005) Topics in Banach Space Theory. *Graduate Texts in Mathematics* Springer

[2] Albiac, F. and Wojtaszczyk, P. (2006) Characterization of 1-greedy bases. *J. Approx. Theory* 138, 65-86.

[3] Arvanitakis, D. and Alexander, D. (2006) Weakly null sequences with an unconditional subsequence. *Proc. Amer. Soc.* 134, 67-74.

[4] Baishanski, B. (1983) Approximation by polynomials of given length, *Illinois J. Math* 27, 449-458.

[5] Bechler, P. DeVore, R. Kamont, A. Petrova, G. (2007) Greedy Wavelet Projections are Bounded on *BV* . *Trans. Amer. Math.*

[6] Bednorz, W. (2008) Greedy Bases Are Best for *m*-term approximation. *J. Approx. Theory* (soon)

[7] Cohen, A. DeVore, R. Perushev, P. and Xh, H. (1999) Nonlinear approximation and the space *BV* ($\mathbb{R}^2$), *Amer. J. Math.* 121, 587-629.

[8] Cordoba, A. and Fernandez, P. (1998) Convergence and divergence of decreasing rearranged Fourier series, *SIAM J Math Anal*, 29, 5, 1129-1139.

[9] DeVore, R., Petrova, G. and Temlyakov V. (2003). Best basis selection for approximation in L_p. *J. of FoCM* 3, pp. 161-185.

[10] Dilworth, S.J, Kalton N.J., Kutzarova, D. (2003) On the existence of almost greedy Banach spaces. 159, No 1., 67-101.

[11] Dilworth, S.J, Kalton N.J., Kutzarova, D. and Temlyakov (2001) V.S. The thresholding greedy algorithm, greedy bases and duality. *IMI-Preprints*

[12] Dilworth, S.J. and Mitra, D. (2001) A conditional quasi-greedy basis of l_1. *Studia Math.* 144 95-100.

[13] Dilworth, S.J. Odell, E. Schlumprecht, Th and Zsak, A. (2008) Partial Unconditionality, *Preprint*.

[14] Donoho, D. (1993) Unconditional bases are optimal for data compression and for statistical estimation. *Appl. Comput. Harmon. Anal.* 1 100-115.

[15] Donoho, D. L. (1996) Unconditional Bases and Bit-Level Compression. *Appl. Comp. Harm. Anal.* 3, 388-392.

[16] Garrios, G, Hernandez, E. and Martell, J.M. (2008) Wavelets, Orlicz Spaces and Greedy Bases. *App. Comput. Harmon. Anal.* 70-93.

[17] Gowers, W.T. (1996) A new dichotomy for Banach spaces, *Geom. Funct. Anal.* 6, 1083-1093.

[18] Gowers, W.T, and Maurey, B. (1993) The unconditional basic sequence problem. *J. Amer. Math. Soc.* 6, 851-874.

[19] Gowers, W.T. (1997) Banach spaces with small spaces of operators. *Math. Ann.* 307, 543-568.

[20] Johnson, W.B. Maurey, B. Schechtman and G, Tzafriri, L. (1979) Symmetric structures in Banach spaces, *Mem. Amer. Math. Soc.*

[21] Kadec, M.I. and Pelczynski, A. (1962) Bases lacunary sequences and complemented subspaces in the spaces L_p, *Studia Math.* 21 661-676.

[22] Kalton, N.J. Leranoz, C. and Wojtaszczyk, P. (1990) Uniqueness of unconditional bases in quasi-Banach spaces with applications to Hardy spaces, *Israel Math. J.* 72, 299-311.

[23] Kamont, A. (2001) General Haar systems and greedy approximation. *Studia Math.* 145, 165-184.

[24] Kashin, B.S. (1985). Approximation properties of complete orthonormal systems. (Russian) *Studies in the theory of functions of several real variables and the approximation of functions. Trudy Mat. Inst. Steklov.* 172, 187-191.

[25] Komorowski, R.A. and Tomczak-Jeagerman, N. (1995) Banach spaces without local unconditional structure. *Israel J. Math.* 89, 205-226.

[26] Konyagin, S.V. and Temlyakov, V.N. (1999) A Remark on Greedy Approximation in Banach Spaces. *East J. Approx.* 5 1-15.

[27] Konyagin, S.V. and Temlyakov, V.N. (2002) Greedy approximation with regard to bases and general minimal systems. *Industrial Math. Institute*

[28] Lindenstrauss, J. and Tzafriri, L. (1977) Classical Banach Spaces I,II. *Classics in Mathematics*, Springer, Berlin.

[29] Oswald, P. (2001) Greedy algorithms and best *m*-term approximation with respect to biorthogonal systems. *J.Fourier Anal. Appl.*, 325-341.

[30] Temlyakov, V.N. (1998) Greedy algorithm and *m*-term trigonometric approximation, *Constr. Approx.*, 569-587.

[31] Temlyakov, V.N. 1998) Greedy algorithm and *m*-term trigonometric approximation. *Adv. Comput. Math.* 8, 249-265

[32] Temlyakov, V.N. (1998) Non-linear *m*-term approximation with regard to multivariate Haar system, *Esat J. Approx*, 4, 87-106.

[33] Temlyakov, V.N. 1998) Best *m*-term Approximation and Greedy Algorithms *Adv. Comput. Math.* 249-265.

[34] Temlyakov, V.N. (2002) Nonlinear approximation with regard to bases. *Approximation Theory, X* 373-402, Vanderbilt University Press, Nashville, TN.

[35] Wojtaszczyk, P. (2000) Greedy algorithm for general biorthogonal systems, *J. Approx. Theory* 107, 293-314

[36] Wojtaszczyk, P. Projections and nonlinear approximation in the space BV ($\mathbb{R}^d$), *Proc. London Math. Soc.*

[37] Wojtaszczyk, P. (1997) A Mathematical Introduction to Wavelets, *London Math. Soc. Student Texts*, Cambridge University Press, UK.

[38] Wojtaszczyk, P. (1996) Banach Spaces for Analysts, *Cambridge studies in advanced mathematics*, 25, Cambridge University Press, UK.

[39] Wojtaszczyk, P. (2002) Existence of best *m*-term approximation, *Functiones et Approximatio* 30, 127-133.

[40] Wojtaszczyk, P. (2002). Greedy type bases in Banach spaces, *Constructive Theory of Function Theory, Varna 2002* 136-156, Darba, Sofia

[41] Wojtaszczyk, P. (2003) Projections and nonlinear approximation in the space BV ($\mathbb{R}^d$), *Proc. London Math. Soc.*, 87 417-497.

[42] Wojtaszczyk, P. (2006) Greediness of the Haar system in rearrangement spaces *Banach Center Publications: Approximation and Probability*, 385-395, Warsaw.

19

Hardware-oriented Ant Colony Optimization Considering Intensification and Diversification

Masaya Yoshikawa
Meijo University
Japan

1. Introduction

Swarm intelligence is the technological modeling of behaviors of social insects, such as the ant or the honeybee. Although each element comprising swarm intelligence is simple, high grade intelligence emerges when the elements gather to form a swarm. Ant Colony Optimization (Dorigo, M, et al., 1997), which is called ACO, is one of the swarm intelligence and has been attracting much attention recently. The ACO represents a general name of the algorithm inspired by feeding behavior of ants. It has been applied to various combinatorial optimization problems (Ramkumar, A.S. et al., 2006), including the travelling salesman problem (TSP), the floorplanning problem (Luo, R., et al., 2007) and the scheduling problem (Sankar, S.S., et al., 2005). The basic model of the ACO is the ant system (AS) that was proposed by Dorigo et al. (1996), and many ACOs applied to TSP are based on the AS. However, these ACOs require a lot of calculation time, because the optimization process is based on repetitive searches by plural numbers of ants.

In this chapter, a novel hardware-oriented ACO (H-ACO) is proposed to achieve high-speed optimization based on mechanism of ACO algorithm. The characteristics of the H-ACO is as follows: (1) all calculations can be performed with only addition, subtraction, and shift operation, instead of the floating point arithmetic and power calculation which are adopted in conventional ACO; (2) a new technique using Look-Up-Table (LUT) is introduced; and (3) in addition to upper and lower limits, benchmarks are set to the pheromone amount. Experiments using benchmark data prove effectiveness of the proposed algorithm.

The organization of this chapter is as follows: Section 2 describes the search mechanism of ACO and briefly surveys the ACO research in terms of the computational time. Section 3 explains H-ACO. Section 4 reports the results of computer simulations applied to travelling salesman problem. Section 5 summarizes the chapter.

2. Preliminaries

2.1 Ant colony optimization

Ant Colony System is one of the expansion algorithm of AS, and it shows better capability than genetic algorithm and simulated annealing when applying to TSP. Therefore, we adopt Ant Colony System as a base algorithm and the target problem is TSP. Hereafter, ACO indicates Ant Colony System.

The search mechanism of ACO utilizes the static evaluation value and the dynamic one. The static evaluation value called heuristic value is peculiar information of the target problem, and the dynamic evaluation value is pheromone amount. Usually, a reciprocal number of the distance is adopted as the heuristic value, when ACO is applied to TSP. Specifically, ant k in city i selects the move to city j according to probability p^k and it is defined as follows.

$$p^k(i,j) = \frac{[\tau(i,j)][\eta(i,j)]^\beta}{\sum_{l \in n^k} [\tau(l,j)][\eta(l,j)]^\beta} \tag{1}$$

Where, $\tau(i,j)$ is a pheromone value between city i and city j, $\eta(i,j)$ is a reciprocal number of the distance between city i and city j, β is a parameter which controls the balance between static evaluation value and dynamic one, and n^k is a set of un-visit cities.

On the other hand, a pheromone amount on each route is calculated by using two pheromone update rules. One is a local update rule and the other is a global update rule. The local update rule is applied to the route which is selected by equation (1), and it is defined as follows.

$$\tau(i,j) \leftarrow (1-\psi)\tau(i,j) + \psi\tau_0 \tag{2}$$

Where, ψ is a decay parameter in the local update rule, $\tau 0$ is initial value of pheromone. Thus, the local update rule adds the pheromone to the selected route when the ant moves. The global update rule adds pheromone to the best tour (the completed route) of all tours. The best tour usually indicates the shortest tour. The global update rule is defined as follows.

$$\tau(i,j) \leftarrow (1-\rho)\tau(i,j) + \rho\Delta\tau(i,j)$$

$$\Delta\tau(i,j) = \begin{cases} 1/L^+ & if\ (i,j) \in T^+ \\ 0 & otherwise \end{cases} \tag{3}$$

Where, T^+ is the best tour, and L^+ is the total distance of the best tour.

2.2 Related work

Examples of dedicated hardware for ACO are found by Haibin et al. (2007), Nakano et al. (2006), and others. Haibin et al. (2007) proposed the hardware architecture that combines genetic algorithm and ACO, and they showed the validity of the combined architecture. Nakano et al. (2006) implemented the partial function of ACO on FPGA (Filed Programmable Gate Array), and they demonstrated more high-speed than software processing. The authors also proposed the dedicated hardware of ACO (Yoshikawa, M., et al., 2007). However, the developed hardware was based on ordinary ACO algorithm, and adopted floating point arithmetic as calculation of pheromone update rules.

Regarding other meta-heuristic algorithm, the dedicated hardware approach to reduce calculation time are reported by Scotte et al. (1995), Imai at al. (2002), and Frye et al.(1991). Scott et al. (1995) developed a hardware-based GA and demonstrated its superiority to software in speed and solution quality. Imai et al. (2002) proposed a processor element constituting parallel GA, and achieved the parallelism due to the number effect. Frye et al. (1991) developed the dedicated hardware for neural network using analog device.

Thus, no studies have ever seen, to our knowledge, the hardware oriented ACO algorithm which does not utilize floating point arithmetic.

3. Hardware-oriented ACO

3.1 Pheromone update rule

In the H-ACO, in order to control the trade-off between intensification (exploitation of the previous solutions) and diversification (exploration of the search space), upper and lower limits are set in a manner similar to the Max-Min AS. Here, a new benchmark of the pheromone is also introduced. Using this benchmark, an increment of the pheromone is determined. An example in which the pheromone is added is shown in Fig.1, where the horizontal axis indicates the time and the vertical axis indicates the pheromone value.

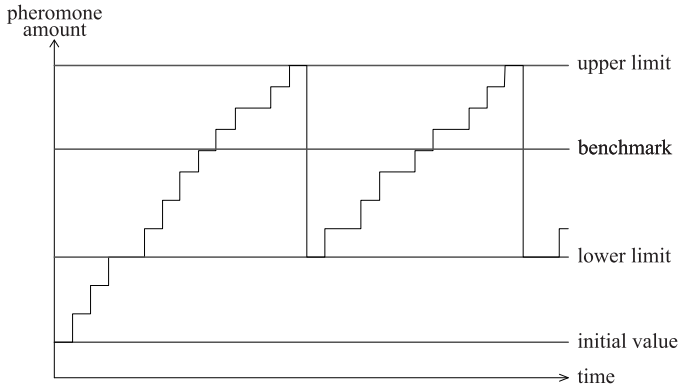


Fig. 1. Example of transition of pheromone

The pheromone value (pheromone amount) is added by performing the search starting from the initial value $\tau 0$. When the pheromone value is smaller than the benchmark, a large number of the pheromones are added from a viewpoint of intensification.
On the other hand, when the pheromone value is larger than the benchmark, a small number of pheromones are added to diversify the search space.
When a particular tour (route) has a large number of pheromones, this indicates that the tour is often selected. When the pheromone value reached the upper limit, the pheromone value is reduced to the lower limit. By this operation, the probability of other tours being selected is increased from a viewpoint of diversification. In other words, the H-ACO is controlled to perform a global search in this case.
A new equation (4) is introduced to the local update rule and it is defined as follows.

$$\tau(i,j) = \tau + \beta$$

$$\beta = \begin{cases} 8 & if \quad 8 < \tau < std \\ 4 & if \quad std < \tau < max \\ 0 & if \quad 8 = \tau \end{cases} \tag{4}$$

Where, *std* represents the benchmark, *max* is upper limit, and the initial value of pheromone ($\tau 0$) is set to 8. In the local update rule, if $\tau(i,j) = 8$, the increment is 0, i.e., no local update will

be performed. In other words, trapping by a local optimal solution can be avoided without adding excessive pheromones to the tour at early stages of the search.
Moreover, high-speed processing can be also realized by reducing the number of processing steps. For example, when the numbers of ants (agents) and cities are denoted as m and n, respectively, the number of processing steps required for the local update rule of $m \times n$ times can be reduced in the first search.
The global update rule is defined by the equation (5).

$$\tau(i, j) = \tau + \varepsilon$$

$$\varepsilon = \begin{cases} 8 & if \quad \tau < std \\ 0 & otherwise \end{cases} \tag{5}$$

As regards the global update rule, if it is applied only when the pheromone value is smaller than the benchmark, a larger global search can be performed.
Fig. 2 shows an example in which the local and global update rules are applied. As shown in the figure, the local update rule is only performed to the tours that pheromone have been added to in the global update rule.

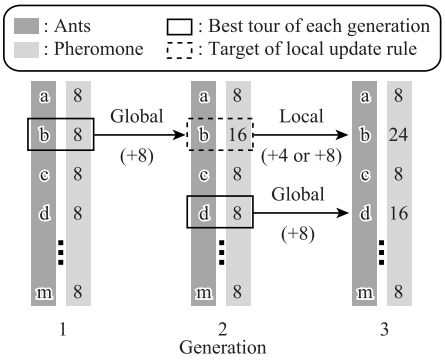


Fig. 2. Example of local update rule and global update rule

That is, in the H-ACO, a global search is performed at the early stage of search, and a local search is performed as the search progressed. Thus, H-ACO can achieve not only speed-up of the pheromone update procedure by reducing the number of processing steps, but also effective search by controlling of intensification and diversification.

3.2 Selection method using Look-Up-Table technique

In the general ACO, when the city of the next destination is selected, a power calculation, as shown in equation (1), is required. In addition, the heuristics value, as the information peculiar to a certain problem, is a decimal because it is the reciprocal of distance. Therefore, when the heuristics value is realized in a dedicated hardware, a floating point arithmetic unit is required. To simplify hardware resources, the number of processing steps and the control, however, power calculations and floating point arithmetic operations are not suitable for the hardware.
In the H-ACO, a selection technique based on the LUT system is introduced. As the heuristics value, the value that is obtained by converting the reciprocal of distance to the

positive integer is used. As the pheromone value, only the values that are multiples of 4, as shown in the equation (4) and (5) of the pheromone update rule, are used. Then, the LUT, into which both the heuristics and pheromone values are input, is created. Fig.3 shows an example of the LUT. By using the integral value and the LUT system, it ensures that all the operations can be performed by simple addition, subtraction, and shift operations.

$\eta(i,j)$ \ $\tau(i,j)$	8	16	24	32	· · ·
0 - 99	8	16	24	32	· · ·
100 - 199	8	16	24	32	· · ·
200 - 299	16	24	32	40	· · ·
300 - 399	24	32	40	48	· · ·
	⋮	⋮	⋮	⋮	

Fig. 3. Example of Look-Up-Table

4. Experiments and discussion

In order to verify the validity of the H-ACO, several comparative experiments are conducted. First, in order to evaluate search performance, the H-ACO is compared with the conventional ACO described in Section 2. 1. The experimental platform is a Pentium 4 3.0 GHz and the program is described by the C language. As experimental data, the original data from 50 cities, in which the optimal solution is already known, and the travelling salesman problem library (TSP.LIB) benchmark data of 100 cities are used. The experimental results are shown in Figs. 4 and 5.

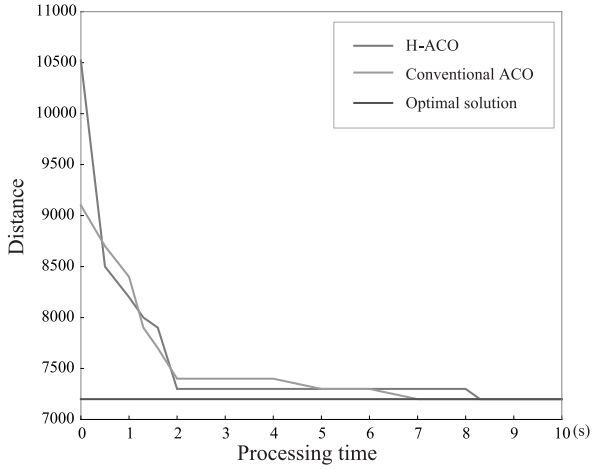


Fig. 4. Result of 50 cities

In both figures, the horizontal axis indicates the processing time and the vertical axis indicates the total distance of the route. As shown in both figures, the search performance of

the H-ACO, which does not use decimal operation and power calculation, is similar to that of the conventional ACO, which does require their use.

Next, an experiment to evaluate the controllability of parameters is conducted. In the conventional ACO, the balance of the pheromone value and the heuristics value is controlled by parameter β in the equation (1) and the balance of the information on the past behaviour and that on the new behavior is controlled by decay parameters ψ and ρ (the evaporation rate) in the equations (2) and (3).

In the H-ACO, the LUT value, upper limit, lower limit, and benchmark are used as the parameters, instead of parameters ψ, ρ and β, to control these balances.

The experimental results with various LUT values are shown in Fig. 6. Fig.6 (1) shows the result of setting the pheromone value larger than the heuristics value.

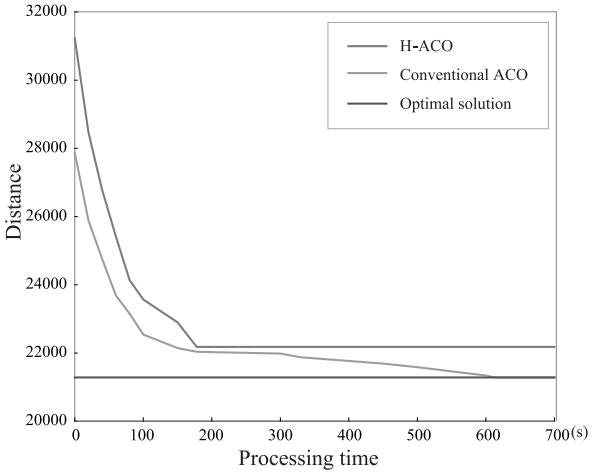


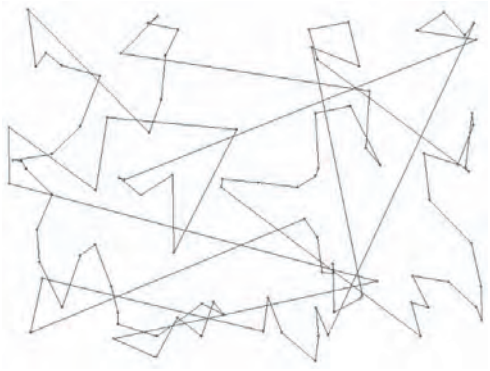
Fig. 5. Result of 100 cities

Fig.6. (3) shows the result when the pheromone value is set to be smaller than the heuristics value.

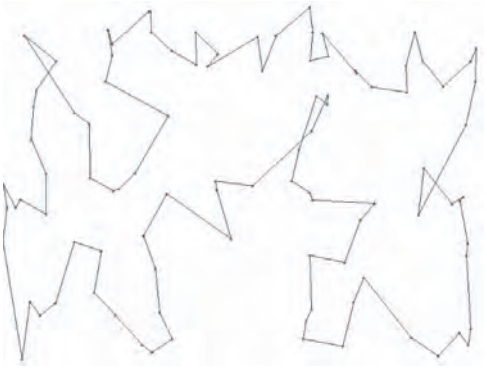
As shown in Fig.6 (1), since the influence of the heuristics value (in this case, it is the distance between the cities) is small, this search is similar to a random search and many tours are intersected.

As shown in Fig.6 (3), since the influence of the distance between the cities is great, the cities (the distance between which is small) are selected in the initial stage of the route construction; that is, a behavior similar to the greedy algorithm is observed. Therefore, in the final stage, to complete the route, cities with large distances between them are selected.

As shown in Fig.6 (2), the pheromone value and the heuristics value are well-balanced, and an effective search is realized. Thus, the technique of using the LUT value, which has been newly introduced to the H-ACO, instead of parameter β and power calculation, which are used in the conventional ACO, is clearly shown to provide an effective selection of the cities with well-balanced pheromone and heuristics values.



(1) $\tau(i, j)$ is larger than $\eta(i, j)$.

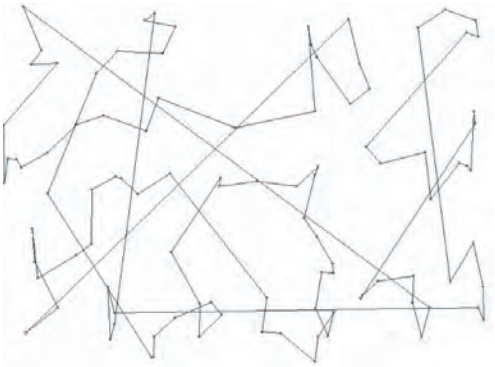


(2) Good balance between $\tau(i, j)$ and $\eta(i, j)$.

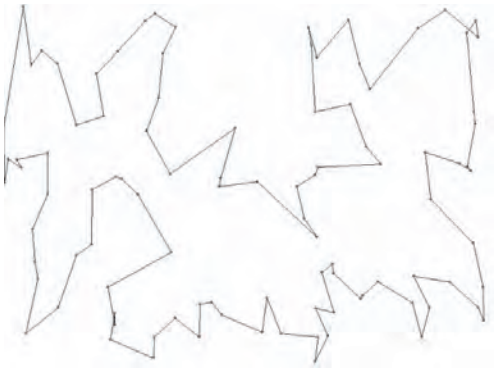
(3) $\tau(i, j)$ is smaller than $\eta(i, j)$.

Fig. 6. Result of several kinds of LUTs

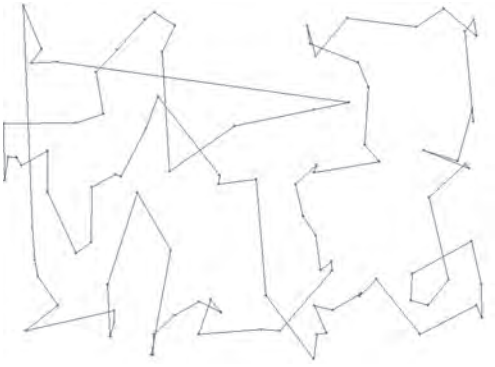
Fig. 7 shows the results of the experiments in which various upper limits and benchmarks of the pheromone value are used.



(1) Both value are high.



(2) Both value are middle.



(3) Both value are low.

Fig. 7. Result of several pairs of upper limit and benchmark

In these experiments, since the relative relationship between the upper limit and the benchmark is considered as maintained even if the lower limit is fixed, only the upper limit

and the benchmark are changed. Figs.7 (1), (3), and (2) show the results of the experiments, in which the upper limit and the benchmark are set to be high, low, and intermediate, respectively.
As shown in Fig.7 (1), since the upper limit was set high, pheromone is accumulated over a long period. This means that, since the past information is considered important, no progress is observed in the search.
As shown in Fig.7 (3), since the distance between the upper limit and the benchmark is small due to the low upper limit, this search is similar to a random search.
In contrast, as shown in Fig.7 (2), when the upper limit and the benchmark are well-balanced, a satisfactory solution is obtained.
Thus, simply by adjusting the upper limit and the benchmark, the same effect as using the decay parameters, which controlled the information on the past behavior and the information on the new behavior, can be realized. Based on the above experimental results, the proposed H-ACO is confirmed to provide a similar solution searching mechanism and ability as seen with the conventional ACO, without the need for floating point arithmetic operations and power calculations.

5. Conclusion

In this chapter, we proposed a novel hardware-oriented ACO algorithm. The proposed algorithm introduced new pheromone update rules using the LUT. It enabled all calculations for optimization with only addition, subtraction, and shift operation. Moreover, it controlled the trade-off between exploitation of the previous solutions and exploration of the search space effectively. As a result, the proposed algorithm achieved not only high speed processing, but also maintenance of the quality of solutions. Experiments using benchmark data proved effectiveness of the proposed algorithm.

6. References

Dorigo, M., Maniezzo, V. & Colorni,A.(1996). Ant system: optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man and Cybernetics*, Part B, Vol.26, No.1,pp.29-41.

Dorigo, M. & Gambardella, L.M.(1997). Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation*, Vol.1, No.1, pp.53-66.

Frye, R.C., Rietman, E.A. & Wong, C.C. (1991). Back-propagation learning and nonidealities in analog neural network hardware, *IEEE Transactions on Neural Networks*, Vol.2, No.1, pp.110-117.

Haibin, D. & Xiufen, Y. (2007). Progresses and Challenges of Ant Colony Optimization-Based Evolvable Hardware, *Proceedings of IEEE Workshop on Evolvable and Adaptive Hardware*, pp.67-71.

Imai, T., Yoshikawa, M., Terai, H. & Yamauchi, H. (2002).Scalable GA-Processor Architecture and Its Implementation of Processor Element, *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol.3, pp.3148-3151.

Luo, R. & Sun, P.(2007). A Novel Ant Colony Optimization Based Temperature-Aware Floorplanning Algorithm, *Proceedings of Third International Conference on Natural Computation*, Vol.4, pp.751-755.

Nakano, M., Iida, M. & Sueyoshi, T. (2006). An Implementation of Ant Colony Optimization for the MaTriX Processing Engine, *Proceedings of IEICE-RECONF2006-27*, Vol.106, No.247, pp.1-6.

Ramkumar, A.S. & Ponnambalam, S.G. (2006). Hybrid Ant colony System for solving Quadratic Assignment Formulation of Machine Layout Problems, *Proceedings of IEEE Conference on Cybernetics and Intelligent Systems*, pp.1-5.

Sankar, S.S., Ponnambalam, S.G., Rathinavel, V. & Visveshvaren, M.S. (2005). Scheduling in parallel machine shop: an ant colony optimization approach, *Proceedings of IEEE International Conference on Industrial Technology*, pp.276-280.

Scott, S.D., Samal, A. & Seth, S. (1995). HGA:A Hardware-Based Genetic Algorithm, *Proceedings of Iternational Symposium on Field-Programmable Gate Array*, pp.53-59.

Yoshikawa, M. & Terai,H. (2007). Architecture for high-speed Ant Colony Optimization, *Proceedings of IEEE International Conference on Information Reuse and Integration*, pp.95-100.

20

Heuristic Algorithms for Solving Bounded Diameter Minimum Spanning Tree Problem and Its Application to Genetic Algorithm Development

Nguyen Duc Nghia and Huynh Thi Thanh Binh
Ha Noi University of Technology
Viet Nam

1. Introduction

The bounded diameter minimum spanning tree (*BDMST*) problem is a combinatorial optimization problem that appears in many applications such as wire-based communication network design when certain aspects of quality of service have to be considered, in ad-hoc wireless network (K. Bala, K. Petropoulos, T.E. Sterm, 1993) and in the areas of data compression and distributed mutual exclusion algorithms (K. Raymond, 1989; A. Bookstein, S. T. Klein, 1996). A more comprehensive discussion of the real-world applications of BDMST was given in Abdalla's seminal dissertation (Abdalla, 2001).

Before the *BDMST* problem can be formally stated, we need some definitions relating to tree diameter and center. Given a tree T, the maximal eccentricity of vertex v is the length (measured in the number of edges) of the longest path from v to other vertices. The diameter of a tree T, denoted as diam(T), is the maximal eccentricity over all nodes in T (i.e the length of maximal path between two arbitrary vertices in T). Suppose that a diameter of a tree is defined by the path v_0, v_1, v_2,…, $v_{[k/2]}$, $v_{[k/2]+1}$, …, v_k. If k is even then $v_{[k/2]}$ is called a center of the tree. If k is odd then $v_{[k/2]}$ and $v_{[k/2]+1}$ are centers of the tree. In that case, the edge ($v_{[k/2]}$, $v_{[k/2]+1}$) is called a center edge.

Let $G = (V, E)$ be a connected undirected graph with positive edge weights $w(e)$. The *BDMST* problem can be formulated as follows: among spanning trees of G whose diameters do not exceed a given upper bound $k \geq 2$, find the spanning tree with the minimal cost (sum of the weights on edges of the trees). As in almost all studies of the *BDMST* problem, and without lost of generality, we will assume that G is a complete graph.

More precisely, the problem can be stated as:

Find a spanning tree T of G that minimizes

$$W(T) = \sum_{e \in T} w(e)$$

subject to

$$\text{diam}(T) \leq k .$$

This problem is known to be *NP*-hard for $4 \le k < |V|-1$ (M.R.Garey & D.S.Johnson, 1979).
In this chapter, we introduce the heuristic algorithms for solving *BDMST*: *OTTC* (Abdall, 2001), *RGH* (R.Raidl & B.A.Julstrom, 2003), RGH_1 (Binh et.at, 2008a), *RGH-I* (A. Singh & A.K. Gupta, 2007), *CBRC* (Binh et al., 2008b). In order to inlustrate the effectiveness of the proposed algorithms, we apply them for initializing the population of our new genetic algorithm with multi-parent recombination operator for solving given problem. Then results of computational experiments are reported to show the efficiency of proposed algorithms.
The chapter is organized as follows. In the next section (section 2), we briefly overview works done in solving *BDMST* problems. Section 3 deals with new heuristic algorithm for solving *BDMST* problem. Section 4 describes our new genetic algorithm which uses heuristic algorithms that already presented in previous section to solve *BDMST* problem. The details of experiments and the comparative computational results are given and discussed in the last section of the chapter.

2. Previous work on the BDMST problem

Techniques for solving the *BDMST* problem may be classified into two categories: exact methods and inexact (heuristic) methods. Exact approaches for solving the *BDMST* problem are based on mixed linear integer programming (N.R.Achuthan et al., 1994), (L Gouveia et al., 2004). More recently, Gruber and Raidl suggested a branch and cut algorithm based on compact 0-1 integer linear programming (M. Gruber & G.R. Raidl, 2005). However, being deterministic and exhaustive in nature, these approaches could only be used to solve small problem instances (e.g. complete graphs with less than 100 nodes).
(Abdalla et al., 2000) presented a greedy heuristic algorithm - the One Time Tree Construction (*OTTC*) for solving the *BDMST* problem. *OTTC* is based on Prim's algorithm in (R. Prim, 1957). It starts with a set of vertices, initially containing a randomly chosen vertex. The set is then repeatedly extended by adding a new vertex that is nearest (in cost) to the set, as long as the inclusion of the new node does not violate the constraint on the diameter of the tree. This algorithm is time consuming, and its performance is strongly dependent on the starting vertex.
Raidl and Julstrom proposed in (G.R. Raidl & B.A. Julstrom, 2003) a modified version of *OTTC*, called Randomized Greedy Heuristics (*RGH*). *RGH* starts from a centre by randomly selecting a vertex and keeping it as the fixed center during the search. It then repeatedly extends the spanning tree from the center by adding a randomly chosen vertex from the remaining vertices, and connecting it to a vertex that is already in the tree via an edge with the smallest weight. The obtained results showed that on Euclidean instances *RGH* performs better than *OTTC*, whereas on non-Euclidean instances the situation is reversed.
RGH could be summarized in the following pseudo-code (G.R. Raidl & B.A. Julstrom, 2003)

```
T  ← ∅;
U  ← V;
v0  ← random(U);
U ← V – {v0};
C ← {v0};
depth[v0] ← 0;
if (odd(k)) {
       v1  ← random(U);
```

```
        T  ← {(v0,v1)};
        U ← U − {v1};
        C ← C ∪ {v1};
        depth[v1] ← 0;
}
while (U ≠ ∅) {
        v ← random(U);
        u ← argmin {c(x,v): x ∈ C};
        T ← T ∪ {(u,v)};
        U ← U − {v} ;
        depth[v] ← depth[u] + 1;
        if (depth[v] < [k/2])
                C ← C ∪ {v} ;
}
return T;
```

Raidl and Julstrom proposed a genetic algorithm for solving *BDMST* problems which used edge-set coded (G.R. Raidl & B.A. Julstrom, 2003) (*JR-ESEA*) and permutation-coded representations for individuals (B.A. Julstrom & G.R. Raidl, 2003) (*JR-PEA*). Permutation-coded evolutionary algorithms were reported to give better results than edge-set coded, but usually are much more time consuming. Another genetic algorithm, based on a random key representation, was derived in (B.A. Julstrom, 2004), sharing many similarities with the permutation-coded evolutionary algorithms. In (M. Gruber & G.R. Raidl, 2005), Gruber used four neighbourhood types to implement variable neighbourhood local search for solving the *BDMST* problem. They are: arc exchange neighbourhood, level change neighbourhood, node swap neighbourhood, and center change level neighbourhood. Later, (M. Gruber et al., 2006), re-used variable neighbourhood searches as in (M. Gruber & G.R. Raidl, 2005), embedding them in Ant Colony Optimization (*ACO*) and genetic algorithms for solving the *BDMST* problem. Both of their proposed algorithms (*ACO* and *GA*) exploited the neighbourhood structure to conduct local search, to improve candidate solutions. In (Nghia & Binh, 2007), Nghia and Binh proposed a new recombination operator which uses multiple parents to do the recombination in their genetic algorithm. Their proposed crossover operator helped to improve the minimum and mean weights of the evolved spanning trees.

More recently, in (A. Singh & A.K. Gupta, 2007), Alok and Gupta derived two improvements for *RGH* heuristics (given in (G.R. Raidl & B.A. Julstrom, 2003)) and some new genetic algorithms for solving *BDMST* problems (notably the *GA* known as *PEA-I*). *RGH-I* in (A. Singh & A.K. Gupta, 2007) iteratively improves the solution found with *RGH* by using level change mutation. It was shown in (A. Singh & A.K. Gupta, 2007) that *RGH-I* has better results than all previously-known heuristics for solving the *BDMST* problem. *PEA-I* employs a permutation-coded representation for individuals. It uses uniform order-based crossover and swap mutation as its genetic operators. *PEA-I* was shown to be the best *GA* of all those tried on the *BDMST* problem instances used in (A. Singh & A.K. Gupta, 2007). In (Binh et al., 2008a), Binh et al., also implement another variant of *RGH*, which is called RGH_1. RGH_1 is similar to *RGH*, except that when a new vertex is added to the expanding spanning tree, it is chosen at random, and connected to a randomly chosen vertex that is already in the spanning tree.

3. New greedy heuristic algorithm (center-based recursive clustering)

Our new greedy heuristics is based on *RGH* in (G.R. Raidl & B.A. Julstrom, 2003) and *NRGH* in (Nghia and Binh, 2007), called *CBRC*. We extend the concept of center to every level of the partially constructed spanning tree. The algorithm can be seen as recursively clustering the vertices of the graph, in that every in-node of the spanning tree is the center of the sub-graph composed of nodes in the subtree rooted at this node. It is inspired from our observation (and other such as in (A. Abdalla et.al, 2000), (G.R. Raidl and B.A. Julstrom, 2003) that good solutions to the *BDMST* problem usually have "star-like structures" as can be seen (for a Euclidean graph) in Figure 1.

In a star-like structure, the vertices of the graph are grouped in clusters, and the clusters are connected by a link between their centers. Pseudocode for the new heuristic based on this observation, known as Center-Based Recursive Clustering (*CBRC*), is presented below:

```
1. T ← ∅;
v0 ← Choose_a_Center(V)
U ← V − {v0};
C ← {v0};
depth[v0] ← 0;
If k is odd then
{
 v1 ← Choose_a_Center(U)
    T ← {(v0, v1)};
    U ← U − {v1};
    C ← C ∪ {v1};
    depth[v1] ← 0;
}
2. //Group vertices in U into cluster(s)
   //with centers at v0 or v1
    For each node w in U do
    {
      If k is even then
      {
       w becomes child of v0 ;
       depth[w]=1;
       T ← T ∪ {(w,v0)};
       }
      Else // k is odd
       If Distance(w,v0) ≤ Distance(w,v1) then
         {
           w becomes child of v0;
           depth[w]=1;
           T ← T ∪ {(w,v0)};
         }
       Else
         {
           w becomes child of v1;
           depth[w]=1;
           T ← T ∪ {(w,v1)};
         }
     } //end for
3. Loop
```

```
    V= set of leaves in U with depths < ⌊k/2⌋;
    v= Choose_a_Center(V);
    if(v is empty)
      Break; // Jump out of the loop
    U = U − {v};
  For each leaf node w in U do
   {
    If Distance(w,v) ≤ Distance(w, parent(w)) then
       w becomes child of v;
       depth[w]=depth[v] +1;
       T=T -{(w,parent(w))}+{(w,v)};

  }
```

The algorithm above is a general framework for *CBRC*. It employs two abstract functions, namely, *Choose_a_Center* and *Distance*. The implementations of these functions are expected to affect the performance of the heuristics, and the best choice could depend on the problem instance. We propose below some possible implementations of these two functions.

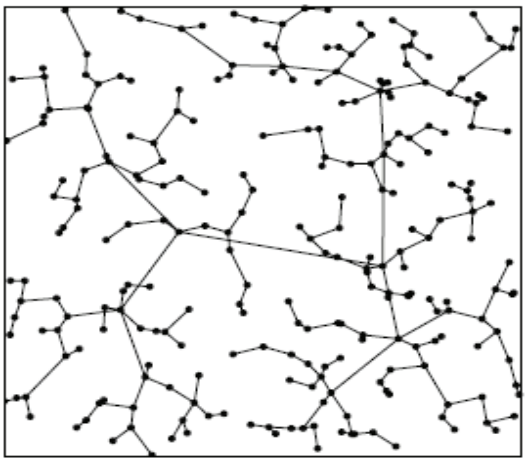


Fig. 1. A "star-like" structure of a typical solution to the BDMST problem.

Implementations of *Choose_a_Center* function:

- v is a center of U if $\sum w \in U$ Distance(v, w) $\rightarrow$ min. If there is more than one such v then choose from them randomly.
- Rank all vertices in U according to $\sum w \in U$ Distance(v, w), then choose v randomly from the first h% of the vertices.
- Conduct h-tournament selection, $\sum w \in U$ Distance(v, w) as the vertex for v.
- Choose v randomly (i.e. it does not depend on Distance at all).

Implementations of the Distance function:

- Distance(u, v) = c(u, v).
- Distance(u, v) = cost the of shortest path between u and v (used for Non-Eclidean graphs).

It can be seen from the pseudo-code of *CBRC* that none of the combinations of *Distance* and *Choose_a_Center* from the above implementations increase the asymptotic computational complexity of the heuristic to more than $O(n^3)$. It is also possible to apply post-

improvement, as proposed in (A. Singh and A.K. Gupta, 2007) to *CBRC* just as for *RGH*. The resulting heuristic is known as *CBRC-I*. In the next section, *CBRC* is tested on some benchmark Euclidean instances of the *BDMST* problem.

4. Proposed genetic algorithm

Genetic algorithm has proven effective on *NP*-hard problem. Much works research on NP-hard problem, particularly in problems relating to tree have been done. Several studies proposed representations for tree (J.Gottlieb et al., 2000), (G.R.Raidl & B.A.Julstrom, 2003), (B.A.Julstrom & G.R.Raild, 2003), (B.A.Julstrom, 2004), (Martin Gruber et al., 2006), (Franz Rothlauf, 2006). This section presents the genetic algorithm for solving *BDMST* problem.

4.1 Initialization

Use *OTTC*, RGH_1, *CBRC, RGH* heuristic algorithms described above for initializing population and edge list for chromosome code.

4.2 Recombination operator

Using *k*-recombination operator as in (Nghia and Binh, 2007).

4.3 Mutation operator

Using four mutations operators: edge delete mutation, center move mutation, greedy edge replace mutation, subtree optimize mutation as in (G.R.Raidl & B.A.Julstrom, 2003).

5. Computational results

5.1 Problem instances

The problem instances used in our experiments are the *BDMST* benchmark problem instances used in (G.R. Raidl & B.A. Julstrom, 2003), (A. Singh & A.K. Gupta, 2007), (Nghia & Binh, 2007), (Binh et al., 2008a) . They are Euclidean instances. All can be downloaded from http://www.sc.snu.ac.kr/~xuan/BDMST.zip. Euclidean instances are complete random graphs in the unit square. We chose the first five instances of each problem size on Euclide instances (number of vertices) n = 100, 250, 500, and 1000, the bounds for diameters being 10, 15, 20, 25 correspondingly (making up 20 problem instances in total).

5.2 Experiment setup

We created two sets of experiments. In the first set of experiment, we compare the performance of the heuristic algorithms: *OTTC*, *RGH*, RGH_1, *CBRC*. The detail of the comparison between other heuristic algorithm for solving *BDMST* problem such as *CBTC*, *RGH-I*, *CBRC-I* can be refered to (Binh et al., 2008b), (A. Singh and A.K. Gupta, 2007).

There are several heuristic algorithms for solving *BDMST* problem as mentioned above but no research has concerned with their effectiveness in application to develop hybrid genetic algorithm. Therefore, in second set of experiment, we will try to fix this problem.

In the second set of experiment, we tested six genetic algorithm algorithms for solving *BDMST* problem. All of the genetic algorithms use recombination and mutation operator mentioned in section 4 but initialized by different heuristic algorithm. GA_1, GA_2, GA_3 uses

CBRC, *OTTC*, RGH_1 algorithm correspondent for initializing the population. GA_4 uses *CBRC*, *OTTC*, RGH_1 , *RGH* for initialization the population with the same rate for each heuristic. GA_5 uses RGH_1, *CBRC* for initializing, the rate of them in the population are 30 and 70. GA_6 uses RGH_1, *OTTC*, *CBRC* for initializing, the rate of them in the population are 35, 35 and 30.

	GA_1	GA_2	GA_3	GA_4	GA_5	GA_6
CBRC	100%	0%	0%	25%	70%	30%
RGH	0%	0%	0%	25%	0%	0%
OTTC	0%	100%	0%	25%	0%	35%
RGH_1	0%	0%	100%	25%	30%	35%

Fig. 2. The rate of the heuristic algorithms use for initialization of the population in each experiment genetic algorithm

5.3 System setting

In the first experiment, the system was run 300 times for each instances. In the second experiment, the population size for GA_1, GA_2, GA_3, GA_4 , GA_5, GA_6 was 100. The number of generations was 500. All *GA*s populations used tournament selection of size 3 and crossover rate of 0.5. The mutation rates for center level change, center move, greedy edge mutation, and subtree optimize mutation were 0.7, 0.2, 0.8, and 0.5 respectively.
Each system was allocated 20 runs for each problem instance. All the programs were run on a machine with Pentium 4 Centrino 3.06 GHz CPU using 512MB RAM.

5.4 Results of computational experiments

The experiment shows that:

- Figure 3, 4, 5, 6, 7, 8, 9, 10, 11 show that the proposed heuristic algorithm, called *CBRC* have the best result than *RGH*, *OTTC*, RGH_1. It means that the solution found by *CBRC* algorithm is the best solution in comparison with the other known heuristic algorithm for solving *BDMST* problem on all the instances with n = 100, 250, 500 and 1000 (n is the number of vertices).
- Figure 15 shows that the best solution found by GA_1 have better result about 22% than the *CBRC* which is used for initialization the population in GA_1 on all 20 problem instances.
- Figure 16 shows that sum up of the best solution found by GA_2 have better result about approximately four times than the *OTTC* which is used for initialization the population in GA_2 on all 20 problem instances.
- Figure 17 shows that sum up of the best solution found by GA_3 have better result about over 10 times than the RGH_1 which is used for initialization the population in GA_3 on all 20 problem instances.
- Figure 11 shows that sum up of the best solution found by *CBRC* have better result about 6.5 times than the *OTTC* and 17 times than RGH_1 while the the figure 18 shows that sum up of the best solution found by GA_1 have better result about 0.8% times than the GA_2 and approximately 2% than GA_3.

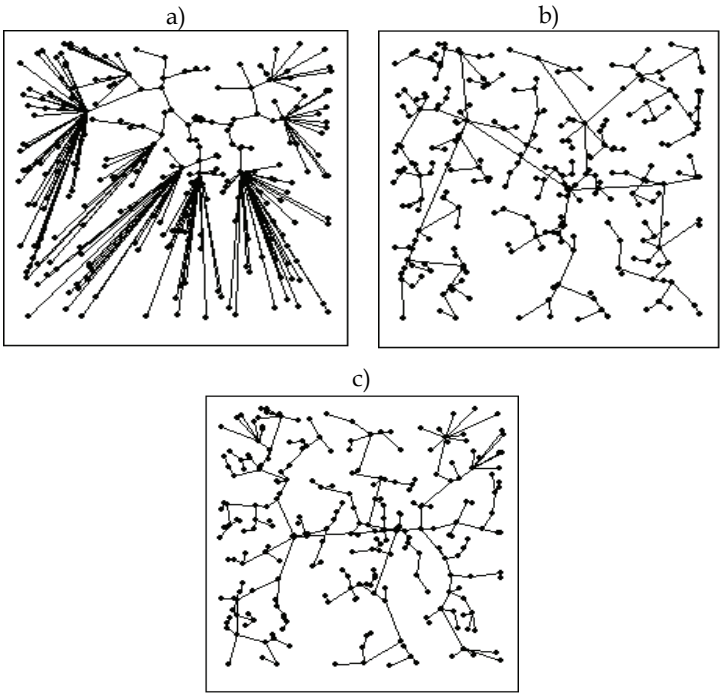


Fig. 3. The best solution found by the heuristics: *OTTC*, *RGH* and *CBRC* on the problem instance with n = 250 and k = 15, test 1:

(a) *OTTC*, weight=42.09; (b) *RGH*, weight=15.14; (c) *CBRC*, weight = 13.32.

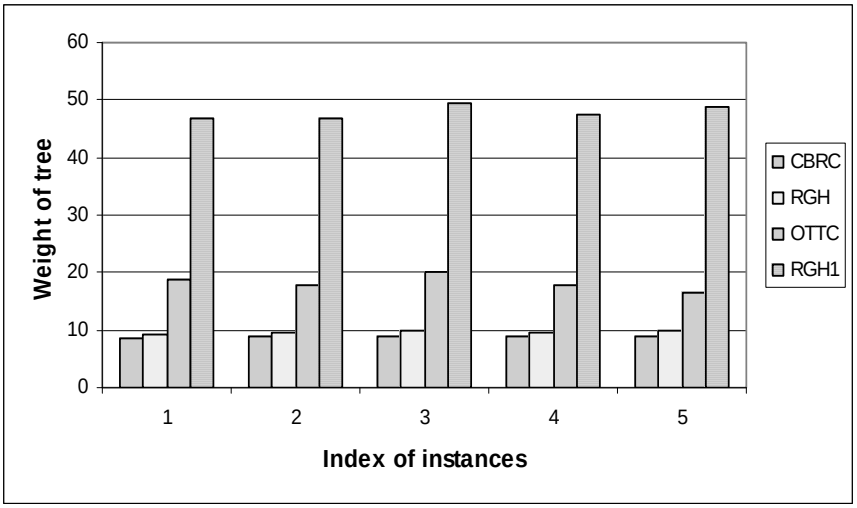


Fig. 4. The best solution found by the four heuristics: *CBR*C, *RGH*, *OTT*C, RGH_1 on the problem instance with n = 100 and k = 10.

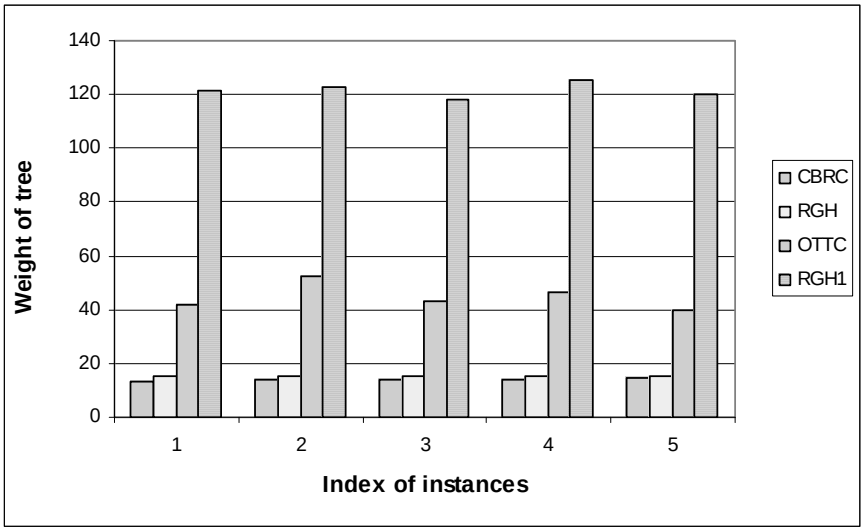


Fig. 5. The best solution found by the four heuristics: *CBRC*, *RGH*, *OTTC*, RGH_1 on the problem instance with $n = 250$ and $k = 15$

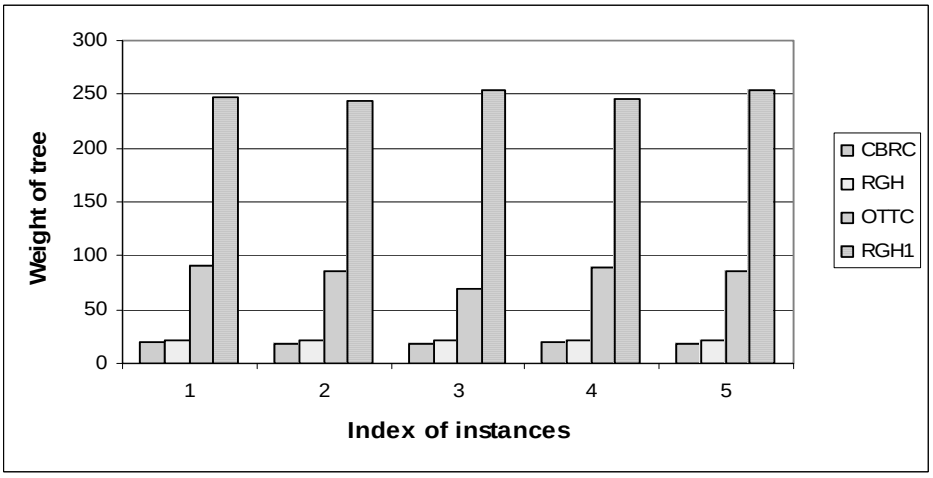


Fig. 6. The best solution found by the four heuristics: *CBRC*, *RGH*, *OTTC*, RGH_1 on the problem instance with $n = 500$ and $k = 20$.

- Figure 18 shows that among GA_1, GA_2, GA_3, GA_4, GA_5, GA_6, sum up of the best solution found by GA_6 have bettest result than the other, otherwise GA_3 have worest result.
- Figure 19 shows that GA_1 have smallest sum of standard deviation otherwise GA_3 have largest sum of standard deviation.
- Figure 20 shows that among GA_1, GA_2, GA_3, GA_4, GA_5, GA_6, the number of instances found best result by GA_5 and GA_6 are biggest otherwise the number of instances found best result by GA_2 and GA_3 are smallest.

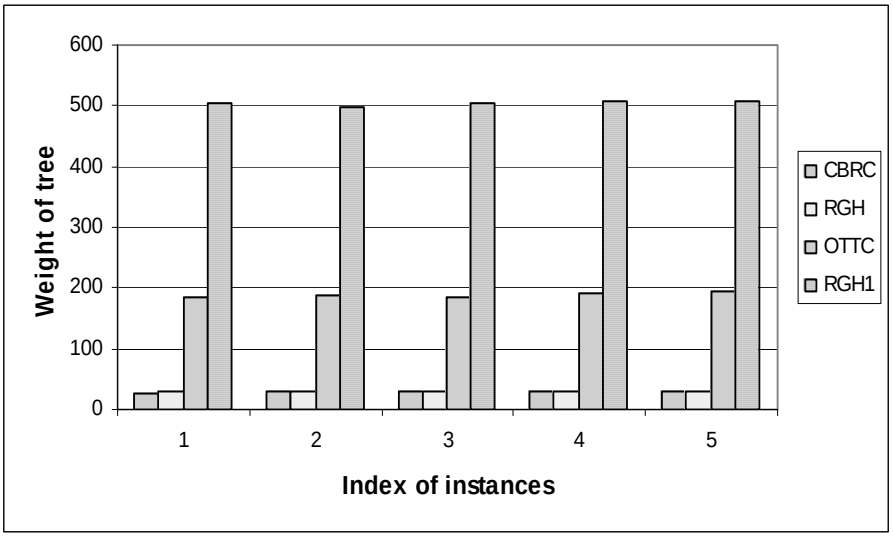


Fig. 7. The best solution found by the four heuristics: *CBRC*, *RGH*, *OTTC*, RGH_1 on the problem instance with n = 1000 and k = 25.

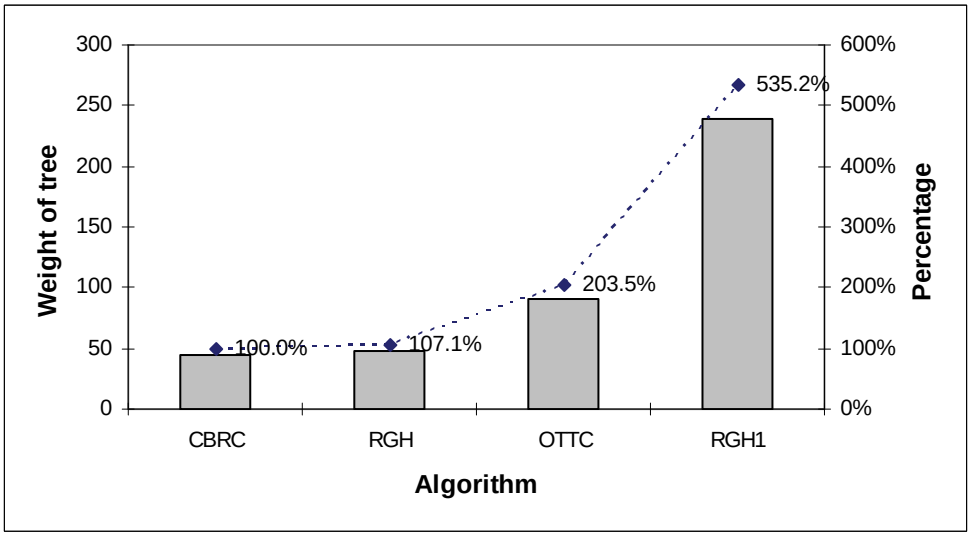


Fig. 8. Comparision of the best solution found by the four heuristics: *CBRC*, *RGH*, *OTTC*, RGH_1 on all the problem instance with n = 100 (5 instances), k = 10

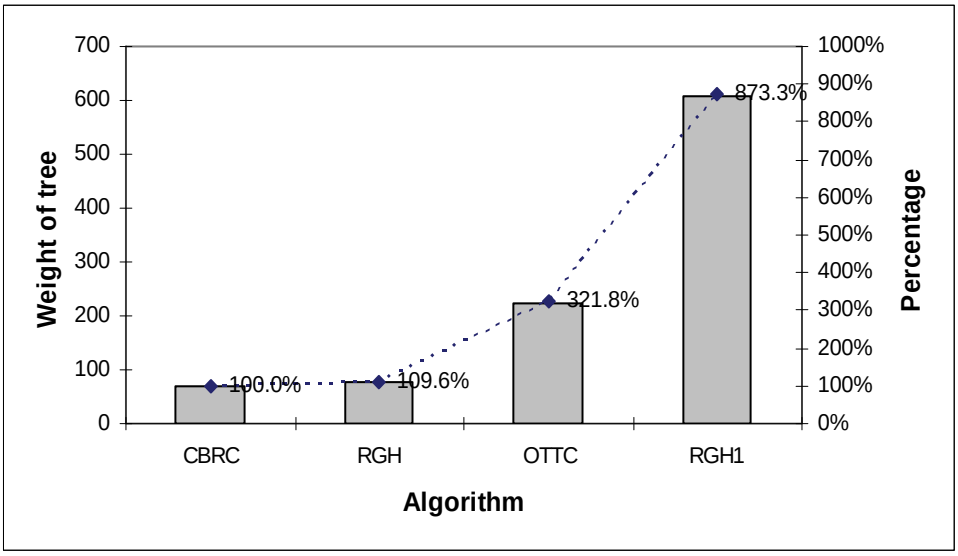


Fig. 9. Comparision between the best solution found by the four heuristics: *CBRC*, *RGH*, *OTTC*, RGH_1 on all the problem instance with n = 250 (5 instances), k = 15

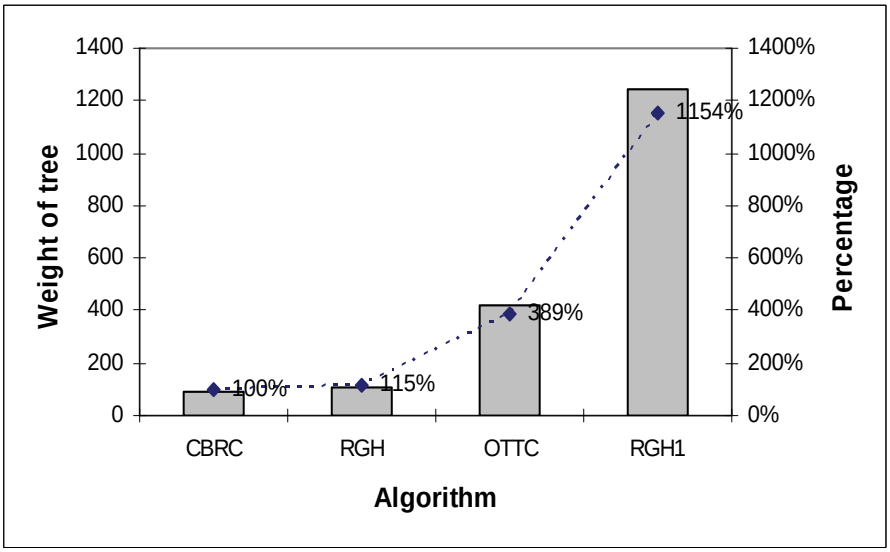


Fig. 10. Comparision between the best solution found by the four heuristics: *CBRC*, *RGH*, *OTTC*, RGH_1 on all the problem instance with n = 500 (5 instances) , k = 20

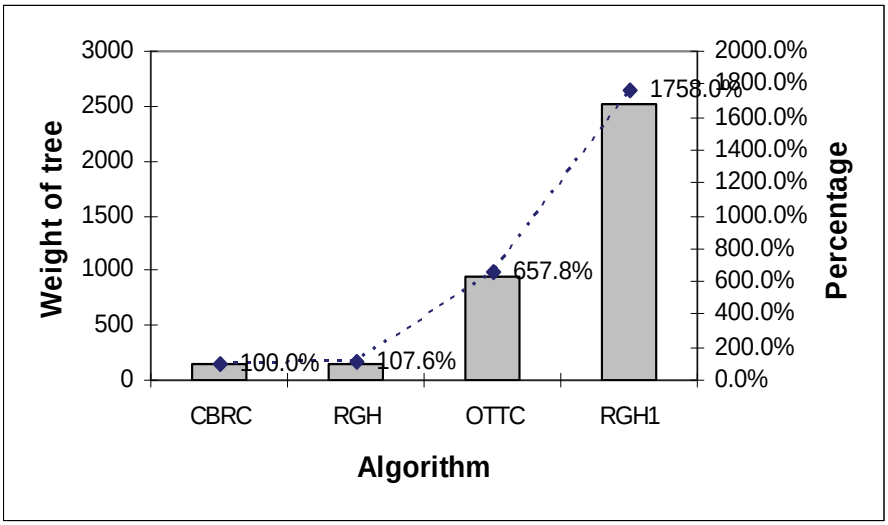


Fig. 11. Comparision between the best solution found by the four heuristics: *CBRC*, *RGH*, *OTTC*, RGH_1 on all the problem instance with n = 1000 (5 instances) , k = 25

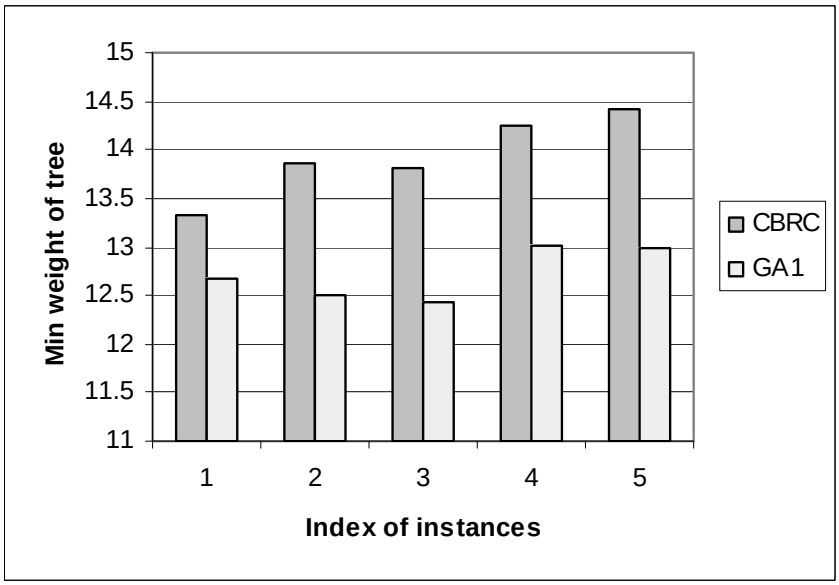


Fig. 12. The best solution found by the *CBRC* and GA_1 on all the problem instance with n = 250, k = 15

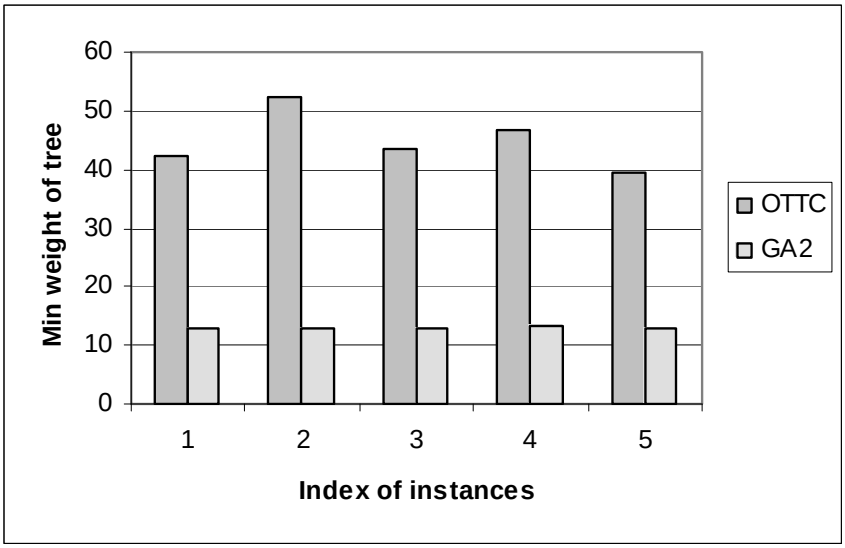


Fig. 13. The best solution found by the *OTTC* and GA_2 on all the problem instance with n = 250, k = 15

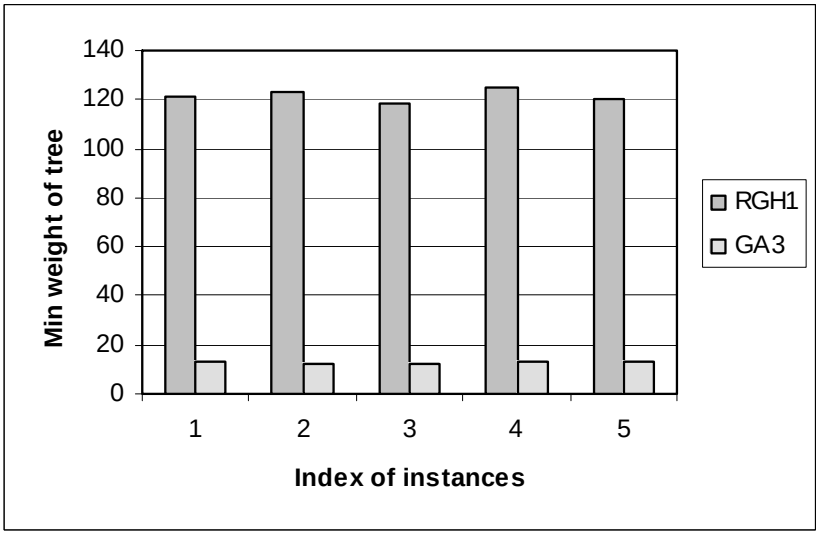


Fig. 14. The best solution found by the RGH_1 and GA_3 on all the problem instance with n = 250, k = 15

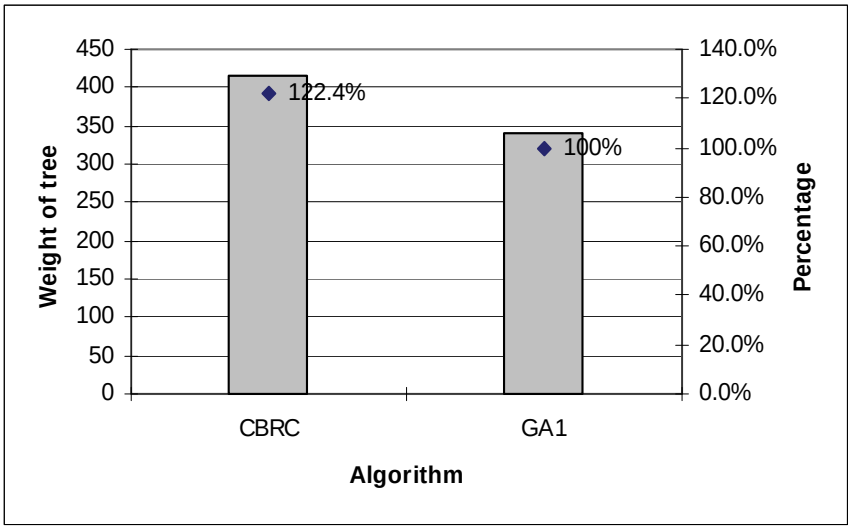


Fig. 15. Sum up the best solution found by the *CBRC* and GA_1 on all the problem instances (20 instances)

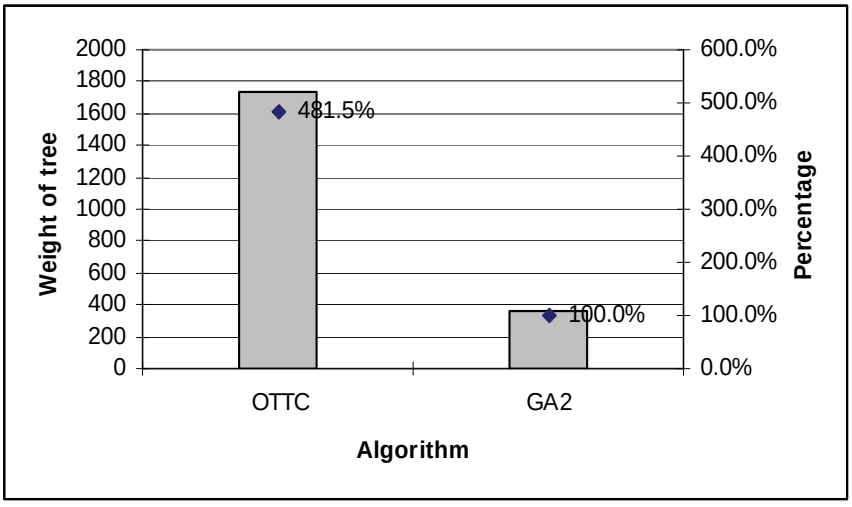


Fig. 16. Sum up the best solution found by the *OTTC* and GA_2 on all the problem instances (20 instances)

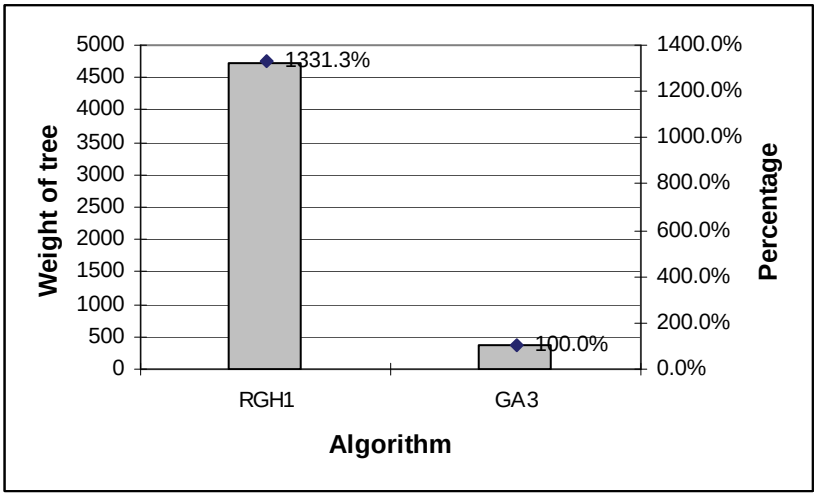


Fig. 17. Comparision between the best solution found by the RGH_1 and GA_3 on all the problem instances (20 instances)

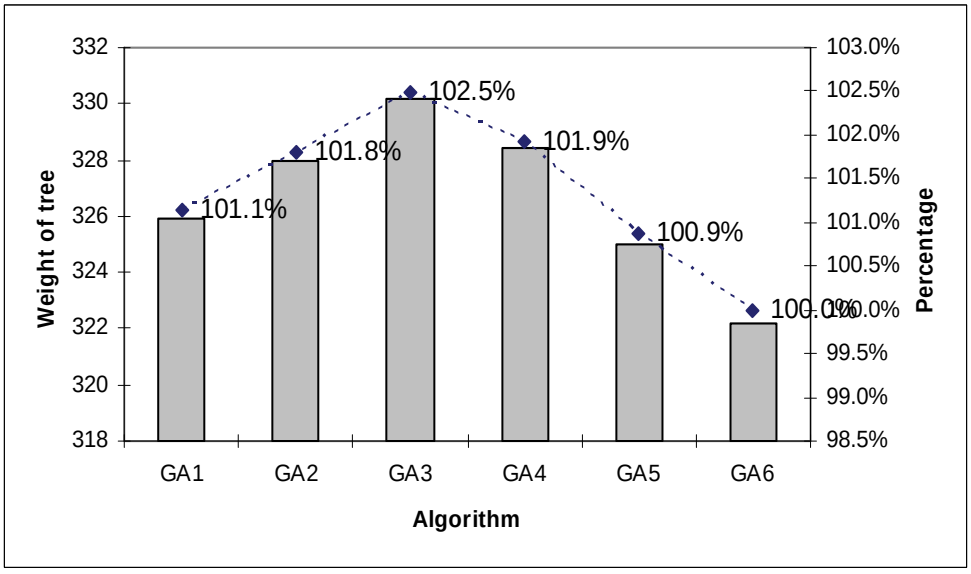


Fig. 18. Comparision between the best solution found by found by GA_1, GA_2, GA_3, GA_4, GA_5, GA_6 on all the problem instance (20 instances)

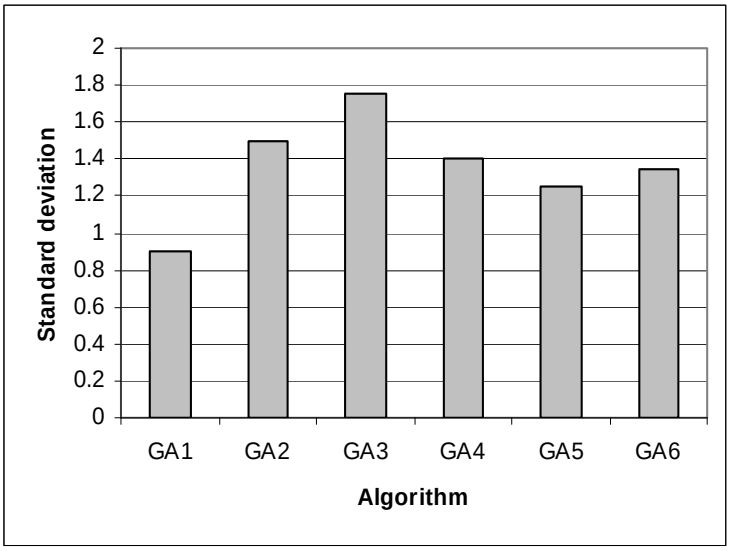


Fig. 19. Comparision between the standard deviation of the solution found by GA_1, GA_2, GA_3, GA_4, GA_5, GA_6 on all the problem instance (20 instances)

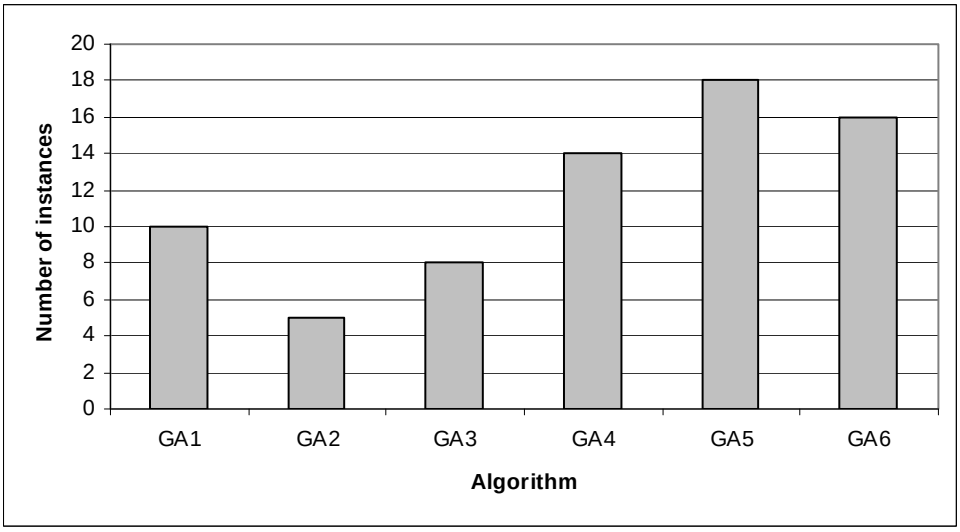


Fig. 20. Number of instances found best result by GA_1, GA_2, GA_3, GA_4, GA_5, GA_6 on all the problem instance (20 instances)

6. Conclusion

We have introduced the heuristic algorithm for solving *BDMST* problem, called CBRC. The experiment shows that *CBRC* have best result than the other known heuristic algorithm for solving *BDMST* prolem on Euclidean instances. The best solution found by the genetic algorithm which uses best heuristic algorithm or only one heuristic algorithm for initialization the population is not better than the best solution found by the genetic algorithm which uses mixed heuristic algorithms (randomized heuristic algorithm and greddy randomized heuristic algorithm). The solution found by the genetic algorithm which uses mixed heuristic algorithm for initialization always is the best result.

7. References

M.R. Garey and D.S.Johnson (1979), Computers and Intractability: A Guide to the Theory of NP-Completeness.

K. Raymond (1989), "A Tree-based Algorithm for Distributed Mutual Exclusion", ACM Transactions on Computer Systems, 7 (1), 1989, pp. 61-77.

K. Bala, K. Petropoulos (1993), and T. E. Stern, "Multicasting in a linear Lightwave Network", in Proceedings of IEEE INFOCOM'93, 1993, pp. 1350–1358

C.C. Palmer and A. Kershenbaum (1994), "Representing Trees in Genetic Algorithms", in Proceedings of The First IEEE Conference on Evolutionary Computation, pp. 379-384

N.R.Achuthan, L.Caccetta, P.Caccetta, and A. Geelen (1994), "Computational Methods for the Diameter Restricted Minimum Weight Spanning Tree Problem", Australian Journal of Combinatorics, 10, pp.51-71.

NA. Bookstein and S. T. Klein (1996), « Compression of Correlated Bit-Vectors", Information Systems, 16 (4), pp. 387-400.

G. Kortsarz and D. Peleg (1997), "Approximating Shallow-light Trees", in Proceedings of the 8th Symposium on Discrete Algorithms, pp. 103-110.

A. Abdalla, N. Deo, and P. Gupta (2000), "Random-tree Diameter and the Diameter Constrained MST", in Proceedings of Congress on Numerantium, pp. 161-182.

J.Gottlieb, B.A.Julstrom, F.Rothlauf, and G.R.Raidl (2001), "Prufer Numbers: A Poor Representation of Spanning Trees for Evolutionary Search", in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001).

A. Abdalla (2001), "Computing a Diameter-constrained Minimum Spanning Tree", PhD Dissertation, The School of Electrical Engineering and Computer Science, University of Central Florida.

G.R. Raidl and B.A. Julstrom (2003), "Edge-sets: An Effective Evolutionary Coding of Spanning Trees", IEEE Transactions on Evolutionary Computation, 7, pp.225-239.

G.R. Raidl and B.A. Julstrom, (2003) "Greedy Heuristics and an Evolutionary Algorithm for the Bounded-Diameter Minimum Spanning Tree Problem", in Proceeding of the ACM Symposium on Applied Computing, pp. 747-752.

B.A. Julstrom, G.R. Raidl (2003), "A Permutation Coded Evolutionary for the Bounded Diameter Minimum Spanning Tree Problem, in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2003), pp.2-7.

B.A. Julstrom (2004), "Encoding Bounded Diameter Minimum Spanning Trees with Permutations and Random Keys", in Proceedings of Genetic and Evolutionary Computational Conference (GECCO'2004).

L Gouveia, T.L. Magnanti and C. Requejo (2004), "A 2-Path Approach for Odd Diameter Constrained Minimum Spanning and Steiner Trees", Network, 44 (4), pp. 254-265.

M. Gruber and G.R. Raidl (2005), "A New 0-1 ILP Approach for the Bounded Diameter Minimum Spanning Tree Problem, in Proceedings of the 2nd International Network Optimization Conference.

M. Gruber and G.R. Raidl (2005), "Variable Neighbourhood Search for the Bounded Diameter Minimum Spanning Tree Problem, in Proceedings of the 18th Mini Euro Conference on Variable Neighborhood Search, Spain.

M. Gruber, J. Hemert, and G.R. Raidl (2006), "Neighbourhood Searches for the Bounded Diameter Minimum Spanning Tree Problem Embedded in a VNS, EA and ACO", in Proceedings of Genetic and Evolutionary Computational Conference (GECCO'2006).

F. Rothlauf (2006), Representations for Genetic and Evolutionary Algorithms, 2nd Edition, Springer-Verlag.

Nguyen Duc Nghia and Huynh Thi Thanh Binh (2007), "A New Recombination Operator for Solving Bouded Diameter Minimum Spanning Tree Problem", in Proceedings of RIVF'2007, LNCS.

A. Singh and A.K. Gupta (2007), "An Impoved Heuristic for the Bounded Diameter Minimum Spanning Tree Problem, Journal of Soft Computing, 11, pp. 911-921.

G. Kortsarz and D. Peleg (1999), "Approximating the Weight of Shallow Steiner Trees", Discrete Application Mathematics, 93, 1999, pp. 265-285.

R. Prim (1957), "Shortest Connection Networks and Some Generalization", Bell System Technical Journal, 36, pp. 1389-1401.

Huynh Thi Thanh Binh, Nguyen Xuan Hoai, R.I Ian McKay (2008a), "A New Hybrid Genetic Algorithm for Solving the Bounded Diameter Minimum Spanning Tree Problem", Proceedings of IEEE World Congress on Computational Intelligence, Hong Kong, LNCS

Huynh Thi Thanh Binh, Nguyen Xuan Hoai, R.I Ian McKay, Nguyen Duc Nghia (2008b), "A new heuristic for the bouded diameter minimum spanning tree problem : some preliminary results", Conference on Artificial Intelligence PRICAI 2008, submitted

21

Opportunistic Scheduling for Next Generation Wireless Local Area Networks

Ertuğrul Necdet Çiftçioğlu[1] and Özgür Gürbüz[2]
[1]*The Pennsylvania State University,*
[2]*Sabancı University,*
[1]*USA,*
[2]*Turkey*

1. Introduction

Wireless access has been increasingly popular recently due to portability and low cost of wireless terminals and equipment. The emerging technologies for wireless local area networks (WLANs) are defined by the IEEE 802.11n standard, where physical layer data rates exceeding 200 Mbps are provisioned with multiple input multiple output antenna techniques. However, actual throughput to be experienced by WLAN users is considerably lower than the provided physical layer data rates, despite the link efficiency is enhanced via the frame aggregation concept of 802.11n.

In a multi user communication system, scheduling is the mechanism that determines which user should transmit/receive data in a given time interval. Opportunistic scheduling algorithms maximize system throughput by making use of the channel variations and multi user diversity. The main idea is favouring users that are experiencing the most desirable channel conditions at each scheduling instant, i.e. riding the peaks. While maximizing capacity, such greedy algorithms may cause some users to experience unacceptable delays and unfairness, unless the users are highly mobile. In order to remedy this problem, we combine aggregation and opportunistic scheduling approaches to further enhance the throughput of next generation WLANs. We argue that aggregation can dramatically change the scheduling scenario: A user with a good channel and a long queue may offer a higher throughput than a user with better channel conditions but shorter queue. Hence, the statement that always selecting the user with the best channel maximizes throughput is not valid anymore.

In this work, we first present our queue aware scheduling scheme that take into account the instantaneous channel capacities and queue sizes simultaneously, named as Aggregate Opportunistic Scheduling (AOS). Detailed simulations results indicate that our proposed algorithm offers significant gains in total system throughput, by up to 53%, as compared to opportunistic schedulers while permitting relatively fair access. We also improve AOS with the principle of relayed transmissions and show the improvements of opportunistic relaying. Later on, we propose another scheduler, which aims to maximize the network throughput over a long time scale. For this purpose, we estimate the statistical evolution of queue states and model the 802.11n MAC transmissions using queuing theory by extending the bulk service model. Utilizing the outcomes of the queuing model, we design Predictive Scheduling with time-domain Waterfilling (P-WF) algorithm. P-WF further improves the

performance of our queue aware schedulers, as the throughput is maximized by applying the water filling solution to time allocations.
This chapter includes an overview of existing literature on opportunistic scheduling for wireless networks in general and presents our proposed algorithms with comparative detailed performance analysis as they are applied into the next generation WLANs.

2. Scheduling approaches for wireless networks

In a multi user communication system scheduling is an essential feature due to its effect on the overall behavior of the network. In this section, we briefly present the prominent scheduling disciplines for wireless networks. In this text, the terms *user* and *station* are used interchangeably.

2.1 Maximum Rate Scheduling (MRS) algorithm

Spatially greedy scheduling schemes, often denoted as Maximum Rate Scheduling (MRS) exploit variations in the time varying wireless channel. The selection metric is the channel capacity, allowing the user with the best channel conditions to transmit at a given time instant [Knopp & Humblet, 1995]. In other words, the selected user k_i^* at the i^{th} transmission opportunity is determined as:

$$k_i^* = \arg\max_k C_i^k \,, \tag{1}$$

where C_i^k denotes the channel capacity of the k^{th} user at the i^{th} transmission opportunity.

Scheduling users according to the channel state can provide significant performance gain due to the independence of fading statistics across users. This phenomenon is called multi user diversity. Although MRS method is shown to be optimal for capacity maximization, an important issue is unfairness in throughput distribution between the users, since the users subject to poor channel conditions may never get a chance to transmit.

2.2 Proportional Fair Queuing (PFQ) algorithm

In Proportional Fair Queuing (PFQ) algorithm, the user with the best channel condition (capacity) relative to its own average capacity is selected [Jalali et al., 2000]. The main aim of PFQ is to maximize throughput while satisfying fair resource allocation. If the users of all channels deviate from their mean capacities in similar ways, all users will gain access to the medium for similar proportions. Note that, being selected for similar proportions does not imply that the users have identical temporal share, since transmission to users with low data rates take longer time durations for the same amount of data. In PFQ, the selected user k_i^* can be found as:

$$k_i^* = \arg\max_k \frac{C_i^k}{\overline{C}_i^k} \tag{2}$$

where $\overline{C}_i^k$ denotes the average channel capacity of the k^{th} user up to the i^{th} transmission opportunity.

2.3 Capacity Queue Scheduler (CQS)

When the above opportunistic schemes are employed, users with high capacity links tend to have small queues, while users subject to poor channel conditions suffer from queue

overflows and long delays. In [Neely et al., 2002], a scheduler is applied which maximizes the link rates weighted by queue backlog differentials for each link. In this downlink setting, the queue-weighted rate metric tries to select user k_i* as

$$k_i^* = \arg\max_k C_i^k Q_i^k \,, \tag{3}$$

where Q_i^k denotes the queue size of the k^{th} user at the i^{th} transmission opportunity. The inclusion of queue length in this scheme provides important insights for fairness. For instance, assume initially that the queue sizes are similar for all users, except for one user whose channel is superior to others. The user with the best channel will be selected and served so its queue size will be reduced; however, in the next scheduling instant, the advantage of better channel quality will be alleviated by the smaller queue size, yielding transmission to other users. The algorithm guarantees stability whenever the arrival rate vector lies within the stability region of the network.

2.4 Shortest Remaining Processing Time First (SRPT) algorithm

Another scheduling algorithm that considers queue size together with capacity is Shortest Remaining Processing Time First (SRPT) method, where the metric is defined as the amount of time it takes to serve all the packets from a given queue [Schrage & Miller, 1966]. Here, the scheduler tries to choose the queue, which can be emptied in the shortest amount of time, i.e., the selected user k_i* at the i^{th} transmission opportunity is determined as:

$$k_i^* = \arg\min_k \frac{Q_i^k}{C_i^k} \cdot \tag{4}$$

2.6 Opportunistic Autorate (OAR) algorithm

Opportunistic Autorate protocol (OAR) is an opportunistic scheduler which takes into account the effect of aggregation, as the users are served in a round-robin fashion [Sadeghi et al., 2002]. While serving each user, the number of packets transmitted for the user depends on the ratio of the user rate to basic rate, hence operating with larger aggregate sizes for users with better channel conditions. It is worthwhile to note that OAR provides temporal fairness since the packet transmission times for each user are equal.

2.7 Longest Queue (LQ) algorithm

Longest Queue (LQ) algorithm, which is also one of the considered schemes for 802.11n [Mujtaba, 2004], is a non-opportunistic scheduling scheme. Using LQ, the scheduler simply selects the station with the largest number of packets in its queue and the channel states are not taken into account. In LQ, the selected user k_i* is found as

$$k_i^* = \arg\max_k Q_i^k \tag{5}$$

The queues of users which have not been served for a long time duration are likely to be long, increasing the scheduling metrics and eventually causing the assocaited user to be served. The reasoning behind the LQ algorithm is to maximize the aggregate size for maximizing the throughput, with the basic assumption that users are experiencing similar

channels with equal data rates. However, the channel quality of stations can vary notably due to time-varying wireless channel and mobility [Rappaport, 2002].
In all of these approaches, the scheduler operates at the physical layer, considering the channel quality and/or queue level for the decision of the selected user. Once the user is selected, the implicit assumption is that a single physical layer data unit is transmitted and the link is fully utilized. With the frame aggregation feature of 802.11n, a number of packets are combined before transmission, so that WLAN overhead is reduced and link efficiency is improved [Tinnirello & Choi, 2005], [Liu & Stephens, 2005]. However, with aggregation, the advantages of opportunism and the statement that selecting the user with the highest channel capacity maximizes the throughput is not valid anymore. For instance, the MRS algorithm with frame aggregation may starve since specific stations are to be served more frequently, their queues will be drained, causing their aggregate sizes to be small, resulting in low efficiency and throughput. Algorithms such as SRPT favour users with high capacity and small queue sizes, which is even worse with frame aggregation causing low throughput. OAR considers frame aggregation and provides temporal fairness, but does not aim throughput maximization. When aggregation is employed, a user with a fair channel and long queue may result in a much higher throughput than a user with a high capacity channel but small queue size. In this work, we study all of the aforementioned algorithms with frame aggregation in the setting of next generation IEEE 802.11n WLANs. We also propose new scheduling algorithms that aim to enhance the performance and fill the performance gap between available and observed data rates by jointly considering channel and queue states of users via throughput calculations.

3. System model

We consider the downlink of a Multiple Input Multiple Output (MIMO) [Telatar, 1999] wireless cellular system that consists of a single access point (AP) communicating with multiple WLAN clients (Figure 1). The system is a closed-loop MIMO OFDM system such that the mobile users measure their channel states and send them as feedback to the AP. Based on the channel state, link capacities are calculated and 802.11n data rates are assigned at the AP according to available capacity[1]. The properties of the fading wireless channel are modeled in the channel matrix $\boldsymbol{H}$, considering large-scale path loss, shadowing and small scale multi-path fading affects. In this paper, the log distance path loss model and the Channel B fading channel model defined by the Task Group n (TGn) are considered. The fading characteristics between individual antenna pairs are spatially correlated and the correlation matrices depend on the angular spread. Further details of the channel model can be found in [Erceg et al., 2004]. Due to low speeds of WLAN users, coherence time is large enough so that channel fading is slow, i.e. the channel is assumed stationary within one transmission opportunity.

[1] In MIMO-OFDM based systems, the channel capacity is calculated by partitioning the system into multiple sub-channels that correspond to different sub-carriers as follows [Bolcskei et al., 2002]: $C = \frac{B}{N_c} \sum_{k=0}^{N_c-1} \log_2(\det(\mathbf{I}_{MN_c} + \rho \mathbf{H}(e^{j2\pi\frac{k}{N_c}})\mathbf{H}^H(e^{j2\pi\frac{k}{N_c}})))$, with $\mathbf{H}(e^{j2\pi\theta}) = \sum_{l=0}^{L-1} \mathbf{H}_l e^{-j2\pi l\theta}$ (N_c: Number of subcarriers). The capacity calculation here considers the air interface specified in 802.11n draft standard. However, the scheduling algorithms can be applied to any other air interface with appropriate capacity calculations.

For medium access, we consider a time division system where only one user is served at a given time period, limited by a duration called transmission opportunity (TXOP).

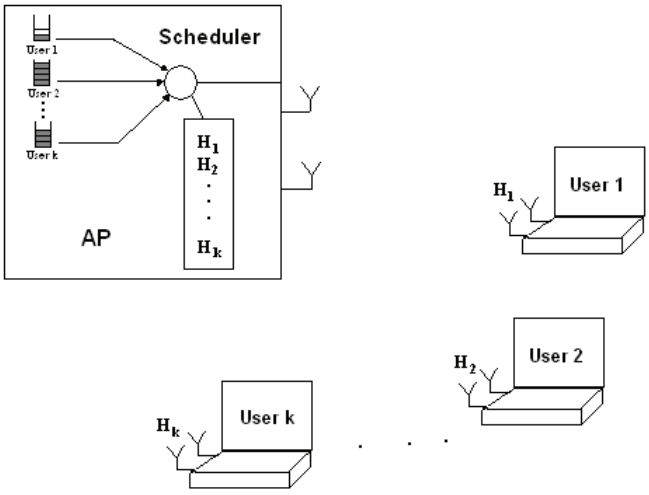


Fig. 1. A typical 802.11n AP and terminals

As defined by 802.11n draft standard, within a TXOP, a two-way handshake with frame aggregation can be performed as shown in Figure 2 [Mujtaba, 2004]. Initiator Aggregation Control (IAC) and Responder Aggregation Control (RAC) are RTS/CTS-like reservation messages, which also involve training sequences to help (MIMO) channel estimation and data rate selection.

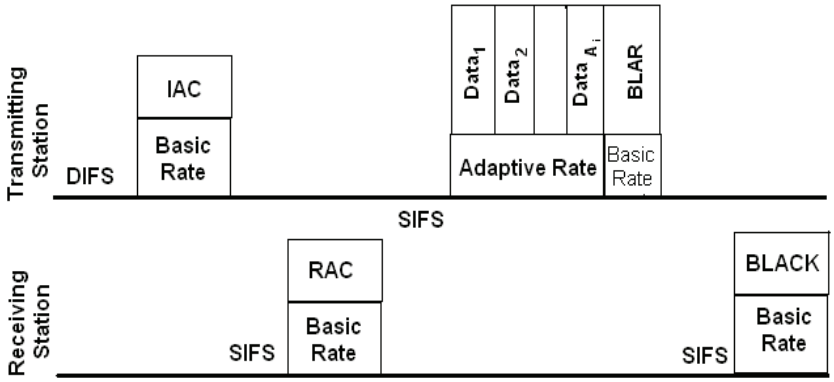


Fig. 2. Example aggregate frame transmission

After IAC/RAC exchange, a number of data packets are aggregated in one frame and an acknowledgement is requested in the end via the Block ACK Request (BLAR) packet. The destination station replies with a Block ACK (BLACK) packet that contains the reception status of packets in the aggregation. The data packets are transmitted at the selected transmission rate, while the control packets (IAC, RAC, BLAR and BLACK) are transmitted at the basic rate, so that all stations can decode these packets. The inter frame spacing (DIFS,

SIFS) values are as in the 802.11 specification. At each TXOP, the AP transmits to a station selected according to the implemented scheduling algorithm.

4. Proposed scheduling algorithms for next generation WLANs

4.1 Aggregate opportunistic scheduling

Despite the performance enhancing techniques introduced by IEEE 802.11n, namely MIMO and frame aggregation, the throughput observed by the system depends on the channel and queue states of the selected user, hence scheduling. Our motivation here is that throughput can essentially shape scheduling, and we propose *Aggregate Opportunistic Scheduling (AOS)* algorithm [Ciftcioglu & Gurbuz, 2006], where the scheduler tries to maximize the instantaneous throughput when the AP is transmitting a number of packets in aggregation to a selected user. In other words, for i^{th} TXOP, the AOS scheduler selects a user k_i^* as

$$k_i^* = \arg\max_k S_i^k \,, \tag{6}$$

where S_i^k is the throughput calculated for i^{th} TXOP and k^{th} user with the actual system overhead and parameters, as shown next. Considering traffic destined to the k^{th} station in the i^{th} TXOP, the point-to-point downlink throughput, S_i^k, can be calculated as

$$S_i^k = \frac{A_i^k \cdot L_P}{\frac{L_{IAC}}{r_0} + \frac{L_{RAC}}{r_0} + 4.T_{PLCP} + DIFS + 4.\tau + 3.SIFS + \frac{L_{BLACK}}{r_0} + \frac{L_{BLAR}}{r_0} + \frac{A_i^k.(L_P + L_{MH})}{C_i^k}}. \tag{7}$$

with $A^k{}_i$ being the instantaneous aggregate size to user k at i^{th} TXOP and L_P, L_{IAC}, L_{RAC}, L_{BLACK}, L_{BLAR} are the length of the data, reservation, ACK and ACK request packets, respectively. L_{MH} is the MAC header in bits, T_{PLCP} is duration of physical layer training header, τ is the one way propagation delay and DIFS, SIFS are inter frame spacing times specified in 802.11 [Mujtaba, 2004]. Finally, r_0 is the basic data rate at which control packets are transmitted and C_i^k is the instantaneous capacity, i.e., maximum achievable data rate to communicate with user k, which depends on the channel state. Instantaneous aggregate size is determined as the minimum of the user's queue size and the maximum allowable aggregation size, which is set according to limit of transmission opportunity duration. Here, only the downlink traffic is considered, hence there are no collisions and losses are merely due to protocol, packet and physical layer overhead.

Another version of AOS, *Aggregate Discrete Opportunistic Scheduling (ADOS)* is also developed with slight modifications. In ADOS, again the throughput maximizing user is selected, but the throughput values are calculated by substituting one of the specified transmission rates of 802.11n, r_i^k for capacity, C_i^k in throughput calculation in (7). r_i^k is selected from the set, R_d ={12,24,36,48,72,96,108,144,192,216} Mbps through a rate matching mechanism, as defined in [Mujtaba, 2004].

4.2 Scheduling with relaying

In this section, we try to take advantage of relaying in our schedulers through increased data rates due to reduced path loss. Relaying offers improvements in throughput and range

extension in wireless networks, making use of multihop communication [Boyer et al.,2004], [Sreng et al., 2002]. Using intermediate relaying stations enables the communication to be carried out through shorter distances where the path loss much is lower as compared to direct transmission. The reduced path loss results in range extension or improved reliability over the same range, which enables transmitters to use lower transmission powers or using higher data rates.

Our aim is to exploit relaying when it offers throughput enhancement with the information available at the AP. For simplicity, we consider only one relaying station. Figure 3 below shows the relaying scenario, where the end station is located at d_f meters from the AP operating at data rate r_f, and the relay station is located at d_1 (operating at data rate, r_1). The distance between the relay and end stations is d_2, and the data rate of the corresponding link is r_2. Figure 4 depicts our modifcitions to 802.11n transmission sequence so as to allow frame aggregation in relaying mode.

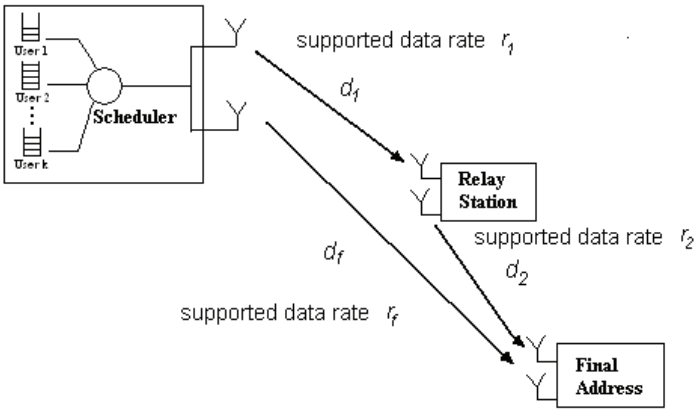


Fig. 3. Relaying with one relay station.

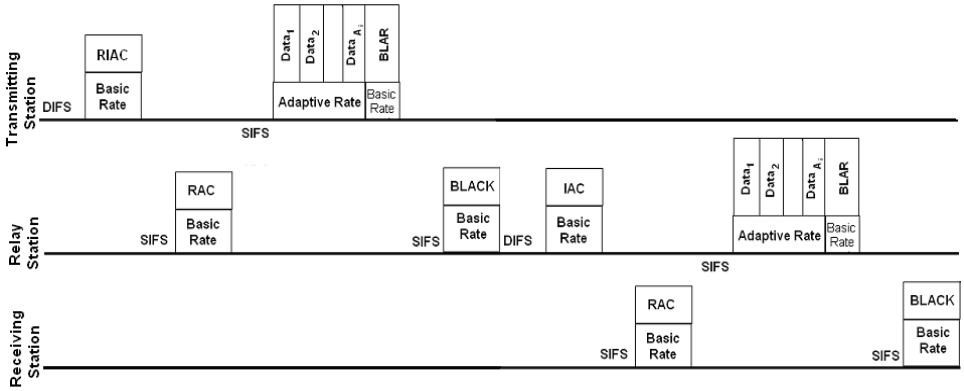


Fig. 4. Frame aggregation with one relay station in IEEE 802.11n.

In order to implement relaying over the 802.11n protocol, the first IAC packet is modified as Relayed IAC (RIAC) by adding fields to the packet which indicate whether relaying is

required or not and the adddress of the relaying station. The relay station initiates another contention-free transmission sequence to the destination. A new transmission sequence will not be initiated at the AP unless the Block ACK is received from the end station. The principle of relaying structure can also be applied to wireless mesh networks [Bicket et al., 2005] ,[Navda et al., 2005] using the IEEE 802.11n interface.

In order to determine whether relaying is beneficial for transmitting data to an end user or not, we compare the transmission durations. Without relaying, the total transmission duration to the end station is given by:

$$T_{direct} = \frac{L_{IAC}}{r_0} + \frac{L_{RAC}}{r_0} + 4.T_{PLCP} + DIFS + 4.\tau + 3.SIFS + \frac{L_{BLACK}}{r_0} + \frac{L_{BLAR}}{r_0} + \frac{A.L_P}{r_f} \quad (8)$$

$$= T_{overhead} + \frac{A.L_P}{r_f} \quad (9)$$

When relaying is employed, the resulting transmission duration is found as:

$$T_{relay} = 2.T_{overhead} + \frac{A.L_P}{r_1} + \frac{A.L_P}{r_2} \quad (10)$$

Clearly, relaying is beneficial if relaying offers a shorter transmission duration than direct transmission, i.e., when $T_{relay} < T_{direct}$. An alternative approach to determine whether relaying is beneficial is to define an *equivalent relaying rate*. For this, we decompose equation (10) as follows:

$$T_{relay} = T_{overhead} + \frac{A.L_P}{r_1} + \frac{A.L_P}{r_2} + T_{overhead}\ , \quad (11)$$

$$T_{relay} = \left(\frac{1}{r_1} + \frac{1}{r_2} + \frac{T_{overhead}}{A.L_P} \right) A.L_P + T_{overhead}\ . \quad (12)$$

Note that the form of (12) is similar to (9) with the total duration as the sum of overhead delay and a rate-dependent term multiplied by the aggregated frame size, in bits. We define the equivalent relaying rate as

$$r_{equivalent} = \left(\frac{1}{r_1} + \frac{1}{r_2} + \frac{T_{overhead}}{A_i.L_P} \right)^{-1}, \quad (13)$$

and rewrite (12) as

$$T_{relay} = T_{overhead} + \frac{A.L_P}{r_{equivalent}} \quad (14)$$

$r_{equivalent}$ not only consists of rate-dependent terms, but it also depends on the aggregate size, A which in turn depends on the queue state for the final station. Hence, increasing aggregate size increases the equivalent relaying rate.

Considering relaying, the queue aware schedulers have been modified as follows: For each destination station, the effective *relaying rate*, $r_{equivalent}$ is calculated using (13) considering each possible intermediate station as a relay station. Then, the best relaying station is selected as the station which enables the maximum effective relaying rate to the destination station. Next, the selected maximum effective relaying rate is compared to the direct rate. If relaying rate is larger than direct transmission rate, relaying is to be preferred, hence the corresponding metric, η_k of the scheduler (AOS, CQS, LQ etc.) is computed for user k using effective relaying rate. If the relaying rate is smaller than direct rate, the metric η_k is computed according to direct transmission. In the end, user scheduling is performed by selecting the user that maximizes the selection metric according to, $k^* = \arg\max_k \eta_k$.

Typically, relaying will improve the rates of stations with poor channel conditions which are located far away from the AP, equivalently increasing their metrics, increasing their chances for being served by the AP. As a result, we expect relaying to improve the fairness peformance of the schedulers. In addition, since higher effective data rates are used, relaying should improve throughput of the non-opportunistic scheduler LQ. For opportunistic schedulers, both effective data rates and the proportion of service for users with poor channels are expected to increase.

4.3 Predictive scheduling with time waterfilling

Selecting the user that maximizes the instantaneous throughput at a specific transmission opportunity may lower the throughput in the subsequent transmission opportunities. Likewise, increasing the participation of low capacity users can later enable the higher capacity users to transmit with larger aggregate sizes and hence result in higher efficiency and throughput. Our aim in this section is to design *block* scheduling algorithms that perform allocation of multiple users, so as to maximize the overall throughput over a long term, the duration of which is set as an external parameter. Hence, we propose an algorithm where the access privileges and proportions of users are determined based on *predicted* per user aggregate size and throughput values. A queuing model is first developed for analyzing packet queueing after transmissions with frame aggregation in 802.11n downlink channel and then the outcomes of the queuing model are used to calculate long term average aggregate size and average throughput, which are then utilized in designing the heuristics of *Predictive Scheduling with Time Water-filling (P-WF)*.

4.3.1 Queuing formulation

Here, we devise a queuing model for aggregate frame transmissions of the 802.11n MAC by extending the bulk service model in [Kleinrock, 1975]. From this queuing model, we compute the state probabilities, where each state corresponds to the number of packets included in the bulk that is an aggregate frame. By using the obtained state probabilities, we compute the expected aggregate size and throughput per user, and then the long term overall system throughput and accordingly design the metrics of the block schedulers. Figure 5 shows the bulk service model, where the packets are served collectively in groups and incoming packets are enqueued. Packets arrive one by one with an average rate, λ packets/second. All of the packets in the queue are served together if the number of packets is less than the bulk size, L. If the queue length exceeds L, only the first L packets are served.

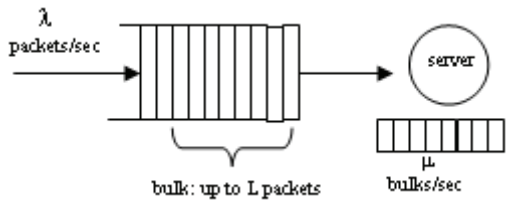


Fig. 5. Bulk service system

The bulk service rate, μ, is defined as the rate of serving bulks, which is assumed constant for all states [Kleinrock, 1975].
The assumption of constant bulk service rate implies that the processing rate in bits per second is to be increased proportionally with the bulk size. For transmissions over a wireless link, the channel data rate can vary due to variations in channel conditions, but in a given rate setting data transmission rate does not change with bulk size. Moreover, in realistic aggregate frame transmissions MAC and physical layer overhead should also be taken into account in determining the service rates. Therefore, for our queing model of aggregate transmission, the service rate μ_j is variable and is obtained as:

$$\mu_j = \begin{cases} \frac{\mu}{j}.\left(\frac{j.L_P}{j.(L_P+L_{MH})+L_{overhead}+r.T_{IFS}}\right) & 1 \le j < L, \\ \frac{\mu}{L}.\left(\frac{L.L_P}{L.(L_P+L_{MH})+L_{overhead}+r.T_{IFS}}\right) & j \ge L, \end{cases} \quad \text{bulks/sec} \tag{15}$$

where j is the number of packets involved in the aggregation; μ is the rate of serving bulks; $L_{overhead}$ accounts for the total overhead including PHY ad MAC headers; T_{IFS} is the sum of interframe durations; r is the channel data rate determined according to the channel conditions which vary over time due to fading.
Assuming Poisson packet arrivals, i.e., exponential inter arrival times, helps us to model the queuing system in terms of a Markov chain, due to the memoryless property of exponential distribution [Kleinrock, 1975]. Although Poisson distribution may exactly model the data traffic applications, it provides an adequate reference for comparing the evolution of different user queues in the AP, hence a relative performance can be obtained for scheduling purposes. Similar assumptions have been made in previous work on modeling WLAN traffic [Bianchi, 2000] as well as scheduler design [Zafer & Modiano, 2005]. Figure 6 depicts the Markov chain representation of the queueing model of aggregate frame transmissions, defining the state as the number of packets in the queue. Packets arrive at average rate λ, and bulks are served at rate μ_j, given by Eq.(15).
Using this model, we derive the state probabilities, $p_1, p_2,...,p_L$, at steady state by solving the balance equations:

$$\lambda p_0 = \mu_1 p_1 + \mu_2 p_2 + ... + \mu_L p_L \Rightarrow p_0 = (1/\lambda)\sum_{j=1}^{L} \mu_j p_j \tag{16a}$$

$$(\lambda+\mu_j)p_j=\mu_L p_{j+L}+\lambda p_{j-1} \qquad 1\le j\le L \tag{16b}$$

$$(\lambda+\mu_L)p_j=\mu_L p_{j+L}+\lambda p_{j-1} \qquad j\ge L \tag{16c}$$

Converting the balance equations into the alternative form by taking the *z*-transform, we obtain *P(z)* in rational form as follows:

$$P(z)=\frac{\sum_{j=1}^{L}[z^{L+j}(\mu_j-\mu_L)-z^L(\mu_j+\frac{\mu_L\mu_j}{\lambda})+\mu_L z^j+\frac{\mu_L\mu_j}{\lambda}]p_j}{\lambda z^{L+1}-(\lambda+\mu_L)z^L+\mu_L} \text{ i.e.,} \tag{17}$$

$$P(z)=\frac{N(z)}{D(z)} \tag{18}$$

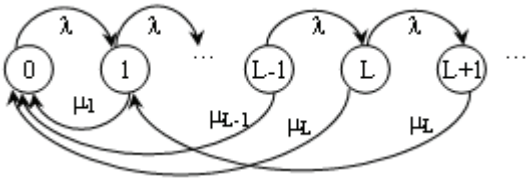


Fig. 6. Markov-chain representation of aggregate frame transmission

The global sum of probabilities should be equal to 1, requiring *P(1)*=1 to be satisfied. Since both *N(1)*=0 and *D(1)*=0, we need to utilize the L'Hospital rule and solve $\lim_{z\to 1}\frac{N'(z)}{D'(z)}=1$. The next step is to obtain state probabilities by taking the inverse transform of *P(z)*. The fact that the bulk service rates are state-dependent has caused the order of *N(z)* to be greater than the order of *D(z)*, so *P(z)* cannot be simplified. We take an alternative approach as follows: Similar to the bulk service model solution in [Kleinrock, 1975], out of the *(L+1)* roots of *D(z)*, *(L-1)* roots are located within the unit circle. Due to the fact that the *z*-transform of a probability distribution is analytical inside the unit circle, *P(z)* should be bounded, which implies that *(L-1)* zeros of *P(z)* must also be the roots of the numerator *N(z)*. *N(z)* must also vanish at each of the (*L*-1) roots of *D(z)* inside the unit circle. This constraint results in a set of *(L-1)* equations. Including the equation provided by the L'Hospital rule, we obtain *L* equations for probabilities p_1, p_2, ..., p_L, and Eq. (16) provides the solution for p_0. The set of equations is solved via numerical computations, obtaining the steady-state probabilities of the system for all the states up to the aggregation limit *L*. The expected aggregate size, $\overline{A}$, and expected throughput, $\overline{S}$, are found as the ensemble average, via

$$\overline{A}=\sum_{j=1}^{L} j.p_j+L.(1-\sum_{j=0}^{L}p_j)\text{,} \tag{19}$$

$$\overline{S}=\sum_{j=0}^{L} p_j S(A_j)+(1-\sum_{j=0}^{L}p_j)S(L)\text{,} \tag{20}$$

where $S(A_j)$ is the throughput achieved with aggregate size A_j.
The queuing model provides us the *expected aggregate size* and *expected throughput* for a single queue (user) given the service rate and applied load. Considering the multi user scenario with time-division multiplexed traffic, the parameters for the queuing model need to be modified by taking the temporal access proportions into account. Given the temporal access proportion of a user as π_n, where $\pi_n \in [0,1]$, the effective channel service rate of that user is to be computed by scaling its link rate by π_n. From Eq. (17), it can be verified that, scaling the service rate by π_n with a given load level has the same effect as keeping service rate and scaling the load level by a factor of $1/\pi_n$. Hence, the effective load at the nth user queue is obtained as λ_n/π_n, and the bulk service rate μ_j is found from Eq. (15) as a function of the data rate of the served user's wireless channel (r_n) and the aggregate size j_n. After computing the state probabilities, the expected throughput per user n, $\overline{S_n}$, is obtained as:

$$\overline{S_n} = f(\pi_n) = \begin{cases} \dfrac{\lambda_n}{\pi_n} & , \dfrac{\lambda_n}{\pi_n} < S(L) \\ S(L), & \dfrac{\lambda_n}{\pi_n} > S(L) \end{cases}, \tag{21}$$

where $S(L)$ is the maximum throughput that can be achieved with the maximum allowed aggregate size, L. The overall network throughput is obtained as the weighted average of the per user throughput values:

$$S_{total} = \sum_{n=1}^{N} \pi_n \overline{S_n}, \tag{22}$$

with N being the total number of users to be scheduled.
The calculation of the state probabilities and estimation of queue size and throughput are to be implemented the AP. The AP has the per user information of traffic load, channel (service) rates and queue states available. Channel states are assumed to be stationary within a scheduling duration, as fading is assumed to be slow due to low mobility in indoor WLANs.

4.3.2 Algorithm description

In order to maximize the total throughput, S_{total} obtained in (22) we propose *Predictive Scheduling with Time-domain Water-filling (P-WF)* [Ciftcioglu & Gurbuz, 2007] as a block scheduling solution that optimizes temporal access proportions, π_n for a given number of users, N. The scheduling problem is described as:

$$\arg\max_{\pi_n} \sum_{n=1}^{N} \pi_n \overline{S_n} \quad \text{such that} \quad \sum_{n=1}^{N} \pi_n = 1. \tag{23}$$

The above problem resembles the power allocation problem among users or multiple transmit antennas for maximizing capacity of multi user or multi antenna fading channels, solved by water-filling. In a water filling problem in general, the aim is to maximize the weighted average of a quantity in the form:

$$\max \sum_{n=1}^{N} (\beta + \gamma_n x_n) \text{ with the constraint } \sum_{n=1}^{N} x_n = 1. \quad (24)$$

The solution for (x_1 ,x_2, ... x_N) is given as [Cover & Thomas, 1991]:

$$x_n^{opt} = (\zeta - \frac{\beta}{\gamma_n})_+ , \ n = 1, ..., N , \quad (25)$$

where $(\theta)_+$ denotes $max(\theta,0)$. For the power allocation problem, the solution, x_n^{opt} is the optimal transmission power level for each channel n with SNR value γ_n and the power cut-off value, ζ is a function of receiver's acceptable threshold SNR. We exploit the mathematical analogy between equations (23) and (24), where power level is analogous to temporal access proportion. Then, we apply the concept of waterfilling for determining the time proportions π_n that maximize S_{total} and we name this method as *time-domain waterfilling*. In order to achieve a full analogy between the equation pairs, we add a constant into the summation term on the left in Eq. (23) and obtain:

$$S' = \sum_{n=1}^{N} (\beta + \pi_n \overline{S}_n) . \quad (26)$$

Maximizing S' is equivalent to maximizing S_{total}, so the waterfilling solution is found as:

$$\pi_n = \left(\varsigma - \frac{\beta}{\overline{S}_n} \right)_+ , \ n=1, ..., N \quad (27)$$

Unlike traditional waterfilling, the solution cannot be computed directly due to the coupling between the waterfilling terms, $\overline{S_n}$ and π_n. At this point, we propose the following heuristic algorithm to find best π_n values:

1. Initialize all temporal proportions equally, as $\pi_n^0 = 1/N$ for n=1...N.
2. For iteration i,
   - Compute the effective load values, $\lambda_n^i = \frac{\lambda_n^0}{\pi_n^i}$, for each user, $\forall n$.
   - Calculate the per user average aggeragate size, $A^i(\lambda_n^i)$ and per user throughput, $S^i(\lambda_n^i)$ from the analytical model.
   - Find access proportions from water filling solution as $\pi_n^{i+1} = \left(\varsigma - \frac{\beta}{S^i(\lambda_n^i)} \right)_+$ also solving for cut off value, ζ using $\sum_{i=0}^{N} \left(\varsigma - \frac{\beta}{S^i(\lambda_n^i)} \right)_+ = 1$. Initially, all of the access proportions are assumed to be greater than zero, and cut off is obtained as: $\varsigma = \frac{1}{N} + \frac{1}{N} \sum_{i=0}^{N} \left(\frac{\beta}{S^i(\lambda_n^i)} \right)$. If $\frac{\beta}{S^i(\lambda_n^i)} > \varsigma$ is satisfied for all users, the iteration is completed. Otherwise, cutoff is calculated by eliminating users with low throughput, until the number of users surpassing ζ is consistent with the number of terms in the summation.

Step 2 with its sub steps is repeated until, after a finite number of iterations, the access proportions (π_ns) converge. The resulting proportions indicate optimal transmission durations of the users relative to the total transmission sequence in which scheduling is applied. Users below the threshold ratio are not served, similar to waterfilling schemes for power allocation, where poor channels are not allowed to transmit when their Signal to Noise Ratio (SNR) fall below the cutoff value.

Having determined the temporal access proportions, next, we need to determine the sequence of transmissions for the selected active users. For this purpose we use an approach that is similar to calculation of finish tags in fluid fair queuing [Leon Garcia & Widjaja, 2004]. Each active user is assigned a turn number, which indicates the number of times the user will be given access throughout the total scheduling duration. The turn number, t_n for user n is determined in two steps: First, the ratio of the access proportion of the user to the transmission duration of serving that user is calculated, then all calculated turn numbers are scaled with respect to the minimum turn number. In other words,

$$t_n = \frac{\pi_n^*}{T_n} = \frac{\pi_n^*}{((\overline{A_n}.L_P)/r_n + T_{overhead})}, \tag{28}$$

$$t' = \min_{\pi_n^*>0}\left(\frac{\pi_1^*}{T_1}, \frac{\pi_2^*}{T_2}, ..., \frac{\pi_{N_{Active}}^*}{T_{N_{Active}}}\right),\ t_1 = \frac{t_1}{t'}, t_2 = \frac{t_2}{t'}, ..., t_{N_{Active}} = \frac{t_{N_{Active}}}{t'} \tag{29}$$

where T_n is the transmission duration of serving user n, $\overline{A_n}$ is the average aggeragate size calculated from the queuing model for user n, $T_{overhead}$ refers to the sum of all the overhead terms in Eq. (7). The optimal solution can yield some of the users with a zero access proportion, so N_{active} is the total number of users with a non-zero access proportion. The transmissions of those active users are scheduled in ascending order of their turn numbers. This ordering makes sure that the users with the smaller access proportions get their allocation before the others.

5. Performance evaluation

5.1 Performance of scheduling algorithms

In this section, the performance of proposed *Aggregate Opportunistic Scheduling (AOS* and *ADOS)* and *Predictive Block Scheduling with Time-Waterfilling (P-WF)* schemes are evaluated in comparison to the scheduling disciplines from the literature namely LQ [Mujtaba, 2004], MRS [Knopp & Humblet, 1995], PFQ [Jalali et al., 2000], CQS [Neely et al., 2002], SRPT [Schrage & Miller, 1966] and OAR[2] [Sadeghi et al., 2002]. The simulations are carried out in the OPNET simulation environment, modeling the wireless channel, physical layer parameters, 802.11 MAC layer with 802.11n enhancements and the scheduling algorithms. For the wireless channel, the log-normal path loss model is simulated with path loss exponent of 2 and log-normal shadowing deviation of 3 dB within a distance of 5 meters

[2] The OAR algorithm defines the aggregate size as the ratio of the data rate of the station over basic rate. Here, we have considered two versions of OAR, where the algorithm is applied with a basic rate of 12 Mbps (OAR-12) and with a basic rate of 24 Mbps (OAR-24).

from the transmitter, and path loss exponent of 3.5 and shadowing variation of 5 dB for distances larger than 5 meters. For the fading model, the Channel B model developed for small office environments and non line-of-sight conditions by TGnSync group is implemented with an rms delay spread of 15 ns and Doppler frequency of 5 Hz. In the physical layer, a practical, 2x2 MIMO configuration is assumed. OFDM parameters such as guard interval, number of subcarriers etc., are chosen according to the 802.11n specifications in [Mujtaba, 2004]. Further details of the MIMO channel can be found in [Erceg et al., 2004]. IEEE 802.11n data rates are adaptively selected from the set {24, 36, 48, 72, 96, 108, 144, 192, 216} Mbps according to the instantaneous channel conditions as explained in [Mujtaba, 2004], [Erceg et al., 2004]. The basic rate, i.e. the common rate for control packet transmission is selected as 24 Mbps. Finally, some of the MAC related parameters of the simulation model are given in Table I. The maximum number of packets allowed in frame aggregation, *L*, is assumed as 63. The downlink traffic is modeled by fixed size (1024 bytes) packets that arrive due to the Poisson distribution. Similar load level is assumed for all stations and increased until the network is brought to saturation. Random topologies are simulated with an AP in the middle and 12 stations uniformly distributed within a radius of 25 m.

In Figure 7 the effect of aggregation on scheduling is illustrated by comparing the throughput of three existing scheduling algorithms MRS, PFQ and LQ. Without frame aggregation, MRS shows the best performance, since the users with the better channel conditions are selected, providing the highest throughput. When frame aggregation is applied however, MRS shows the poorest performance, while LQ has the highest throughput. This is because of the fact that in MRS, the users with better channel capacities are served frequently so their queues do not fill up, resulting in small aggregate size and low throughput. With frame aggregation, the simplest queue aware scheduling scheme, LQ leverages the advantage of frame aggregation.

Parameter	Value
SIFS	16 μ sec= 16 X 10^{-6} sec.
DIFS	34 μ sec= 34 X 10^{-6} sec.
PLCP overhead	44.8 μ sec= 448 X 10^{-7} sec.
T_{IAC}	11.2 μ sec = 112 X 10^{-7} sec.
T_{RAC}	8.7 μ sec = 87 X 10^{-7} sec.
T_{BLACK}	48.7 μ sec = 487 X 10^{-7} sec.
T_{BLAR}	9 μ sec = 90 X 10-7 sec.

Table 1. Some MAC Related Parameters

In the following, we provide the performance analysis when frame aggregation is applied, considering our proposed queue aware throughput opportunistic schedulers AOS, ADOS and P-WF in comparison to existing algorithms LQ, MRS, PFQ, CQS, SRPT and OAR.

As depicted in Figure 8, where simulations are repeated with different topologies and the presented results are average values over ten topologies, proposed algorithms AOS and ADOS significantly outperform all the existing algorithms, e.g., by 53 % over SRPT, by 35 % over MRS, PFQ and by 21% over LQ, as they both maximize the instantaneous throughput. Our predictive block scheduler P-WF provides a further improvement of 4-5% over

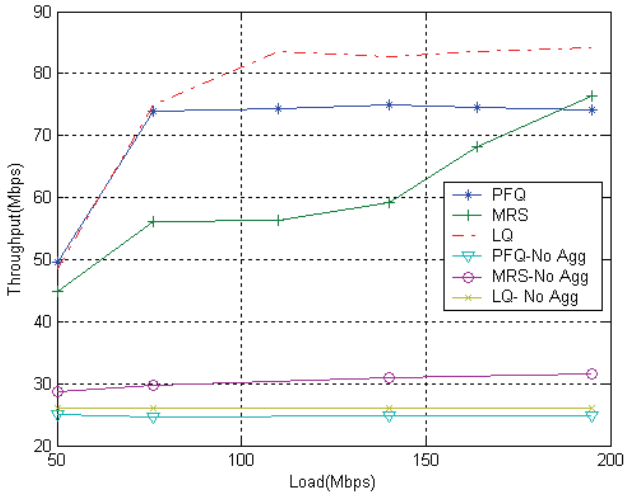


Fig. 7. Throughput of existing schedulers with and without frame aggregation

AOS/ADOS schemes, since it maximizes the throughput in the long term. Among the previous schemes, the CQS algorithm provides the highest throughput. This is followed by OAR and MRS algorithms and the SRPT algorithm exhibits the lowest throughput. In summary, proposed algorithms AOS, ADOS and P-WF provides the highest throughput as they possesses the most explicit insight about the system behavior, considering the effects of the physical medium, MAC efficiency and queue states jointly. It is worthwhile to note that throughput performance of ADOS is close to AOS, implying that the algorithm can be applied after rate matching.

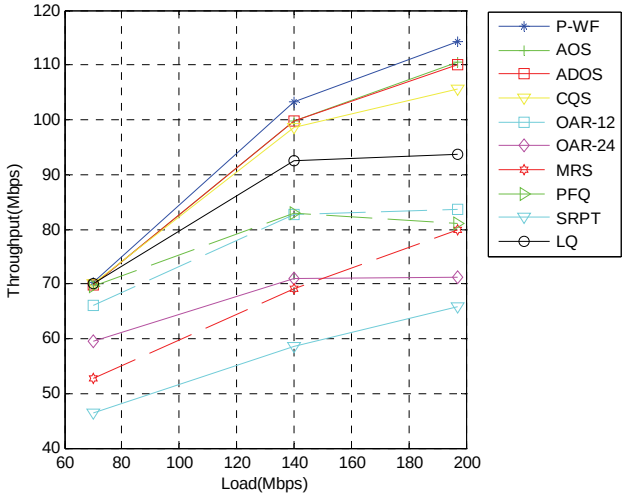


Fig. 8. Throughput of proposed and existing schedulers with frame aggregation

In order to evaluate fairness, we define an unfairness index as the ratio of the standard deviation of station throughputs to the mean throughput, i.e., $UF = \sigma / S_{av}$. It is obvious that the larger *UF* gets, the distribution of throughput among stations becomes more *unfair*. Using the definition of this unfairness index, a picture of the fairness performance of all algorithms under varying load has been obtained as depicted in Figure 9. SRPT and MRS algorithms show the poorest performance in terms of fairness, since they aggressively favour users with high channel capacities. The LQ algorithm is the fairest scheme as it operates like the round robin scheme providing equal access to each station. The CQS algorithm follows the LQ algorithm.

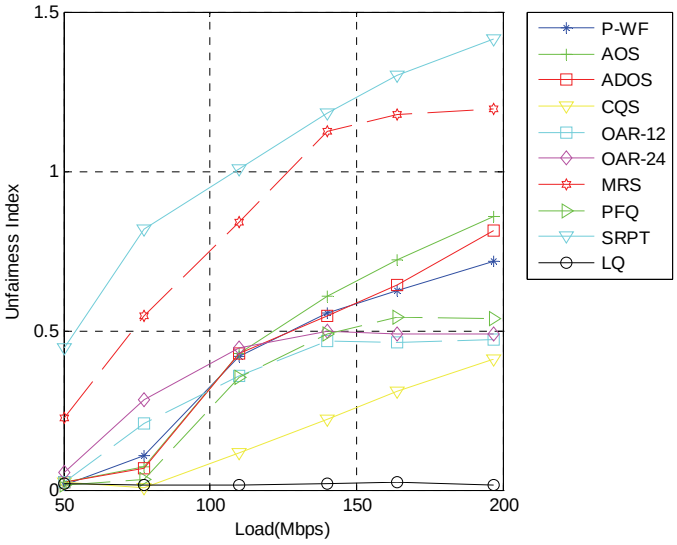


Fig. 9. Fairness performance under varying load

Fairness of our proposed algorithms remain between CQS and MRS. AOS is the most unfair among proposed schemes, since instantaneous throughput is maximized, in an opportunistic fashion. The ADOS algorithm offers slightly more fair distribution than AOS, due to the fact that quantized data rates results in increased emphasis on queue sizes, enhancing fairness. Our predictive block scheduler P-WF improves fairness further, since it considers allocation of multiple users to maximize the long term throughput.

Finally, Figure 10 depicts the MAC efficency of each scheduler, where the actual throughput and time averaged data rates are plotted together as a function of load, again averaged over ten toplogies. LQ and CQS algorithms operate with highest eficiencies, where the average throughput is close to average of physical data rates. SRPT and MRS are the most inefficient schemes, since the achieved throughput levels half or less than half of the average of selected user data rates, which are considerably high.

All our proposed algorithms provide a very good compromise between selected physical layer data rates and efficiency and our predictive block scheduler, P-WF provides the highest throughput with highest efficiency due to the main objective of long term throughput maximization.

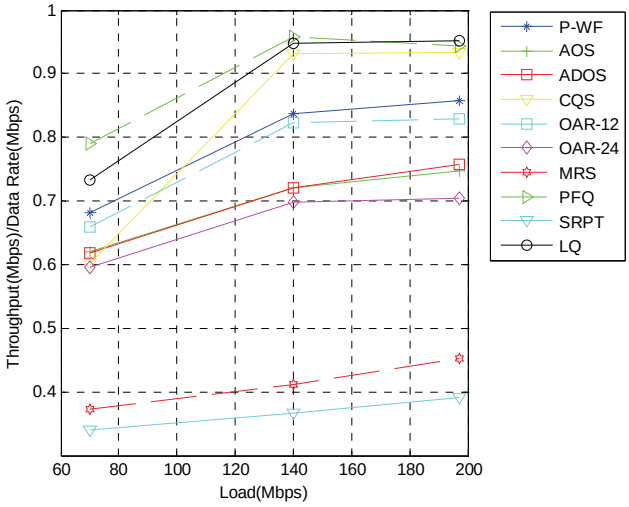


Fig. 10. Average utilization performance

5.2 Performance with relaying

In this section we analyze the effect of incorporating relaying with opportunistic scheduling and frame aggregation. Distances between the AP and destination stations, the AP and intermediate stations, and intermediate stations to destination stations define the respective data rates to be supported in between, so we analyzed relaying by varying the distances between the stations. Before presenting the results, we first demonstrate how the average supported data rates vary by the distance.

In Figure 11, for illustration purposes, we present the average physical data rates achieved for varying distances considering direct transmission and relaying (considering a relay in the middle), neglecting overhead terms. Relaying does not offer improvement for short distances since the maximum data rates are already realized by transmitting over one hop. On the other hand, as distance is increased, the direct transmission rate reduces significantly and improvement of relaying can be observed.

In our next set of simulations we have considered topologies as shown in an example configuration in Figure 12 and we have varied the inner radius, d_1 and outer radius, d_2 together, while keeping $d_2/\ d_1 = 2$. In the figure we show the variation of d_1 only, but the network radius, d_2 is also varied to keep the same ratio. Accordingly, both the good and bad positioned stations are effected in a similar manner in terms of the increasing or decreasing of supported data rates. Yet, the fact that they both increase or decrease does not imply that their ratio remains fixed, hence the probability of opting for relaying is expected to differ with varying distance. We again consider a network with 12 stations and one AP. The total load is set as 200 Mbps, which is evenly distributed between the stations. The algorithms AOS, CQS and LQ are compared with their counterparts aided by relaying. As explained in Section 4, the schedulers are applied using the equivalent relaying rate given by Eq. (13) for a user if it exceeds the rate of direct transmission for that user. The basic rate is selected as 12 Mbps.

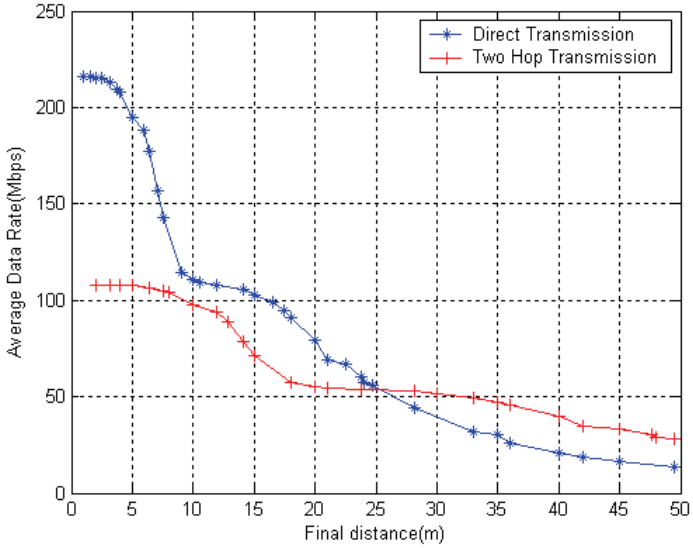


Fig. 11 Comparison of data rates for direct and two-hop neglecing overhead

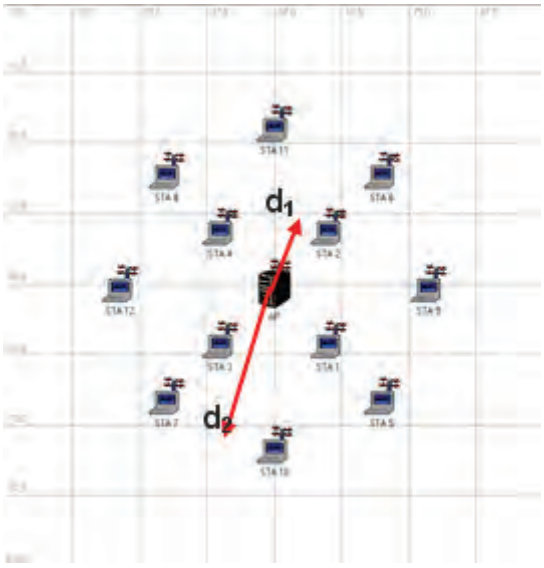


Fig. 12 Relaying topology $d_2/d_1 = 2$

As depicted in Figure 13, without relaying, the total throughput is decreased as the network radius is increased, since the supported data rates are likely to decrease for all stations. When the radius is small, the algorithms perform very similarly, since the topology is close to a uniform topology. However, the effect of distance on total throughput largely depends on the scheduling algorithm used. As the network radius is increased, AOS and CQS outperform LQ since better positioned stations are preferred more frequently. LQ yields very low throughputs, for the algorithm cannot avoid transmitting to the farther stations, even if the supported data rates are very low. This result is due to the fact that although LQ serves each user equally in terms of amount of data, the actual temporal shares of the users are significantly different. Users with very low transmission rates are served for a very long duration, reducing total network throughput drastically.

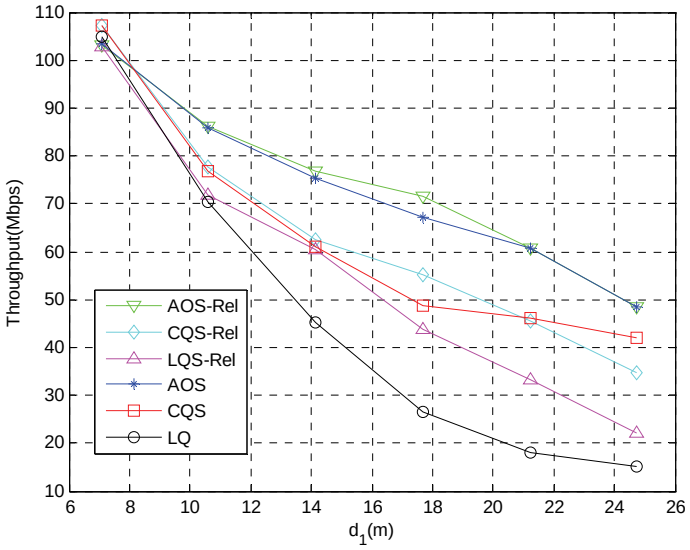


Fig. 13 Throughput with both inner and outer radius varying

As shown in Figure 13, with relaying, the performance of the LQ algorithm is significantly improved. This is because the effective service rates of the farther stations are increased, leading to an increase in throughput. For this setting, we observe that the behavior and performance of the AOS algoritm is not drastically effected by relaying: Initially, all users are supported with high rates and no relaying is selected. When network radius is increased, we see that relaying is employed since direct user rates are reduced, and relaying offers advantage for farther stations. However, as the distance is further increased, the inner stations start to be scheduled more since their queues grow, resulting in larger scheduler metrics than the farther stations. Therefore, in AOS, relaying is not exploited at all for large distances. The behaviour of the CQS algorithm is different than both LQ and AOS. With CQS, initially relaying offers an improvement for throughput, but afterwards as the network radius is increased, the increase in the relative proportion of

outer stations being selected results in an overall decrease in the throughput. Therefore, the frequency of employing relaying in CQS is not as high as LQ, but is still much higher compared with AOS.

In Figure 14 the unfairness index is plotted as a function of the distance. We see that relaying yields an increase in the capacity and enables the outer stations to gain access without growing their queue sizes (as much as the direct case), since their related scheduling metrics are increased. For AOS, fairness is slightly improved when relaying is employed, since the outer users are selected more frequently. For the CQS algorithm, we see that relaying significantly improves fairness preformance since it yields increase in the capacity terms of outer stations in the scheduling metric, enabling the outer stations to gain access without having to grow their queue sizes.

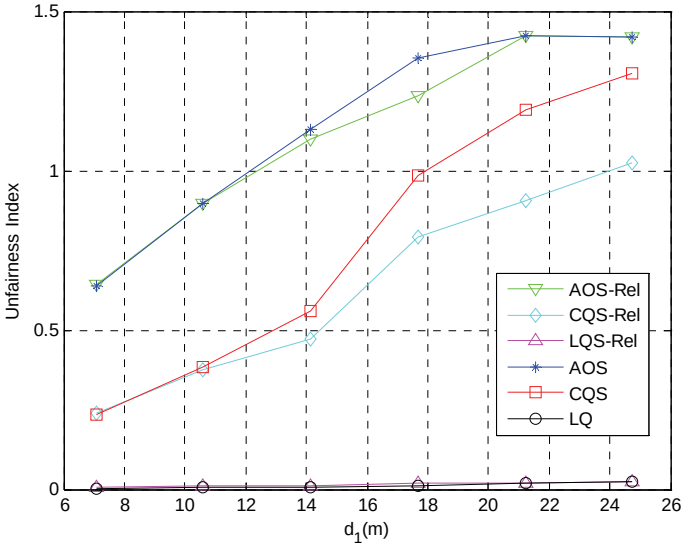


Fig. 14. Unfairness with both inner and outer radius varying

In essence, applying relaying for IEEE 802.11n improves throughput, fairness or even both simultanesously, yet the extent of improvement for opportunistic schedulers is limited by the fact that schedulers tend to give priority to users already with relatively good channel conditions.

6. Conclusions

In this work, we propose a family of scheduling algorithms for IEEE 802.11n, where scheduling decisions are based on throughput, calculated instantaously or considering the long term evolution of user queues. We provide a performance comparison of our schemes with all outstanding algorithms from the literature considering all in the same air interface.

We show that with frame aggregation, spatially greedy scheduling algortihms such as MRS are no longer optimal for maximizing throughput performance. Even though these algorithms yield the maximum physical data rates and they would have provided the highest throughput values in an infinitely backlogged setting if there were no overhead, they all fail considerably under the 802.11n model. This is because of the fact that the observed throughput higly depends on the transmission duration as well as the overhead, especially in WLAN systems which provides improved, 802.11n rates.

Our proposed AOS and ADOS algorithms improve the throughput of such greedy opportunistic scheems, by up to 53% when aggregation is applied. Our block scheduling algorithm P-WF further improves the performance, since the statistical evolution of the queue states are considered and hence the average aggregate size and the throughput are predicted in the long term, justifying the concept that selecting the user which maximizes the instantaneous scheduling metric may not provide maximum performance throughout the entire time duration. This algorithm offers temporal shares of access, in addition to scheduling order, with allocations that provide maximized long term throughput while at the same time providing better fairness. When throughput and fairness performance are considered together, our predictive block scheduler, P-WF stands out as the best scheduling scheme that provides the highest throughput without fairness penalty.

Applying the concept of relaying is slightly differentiated from convential relaying due to overhead. We have shown that for networks which have users located far away from the AP, relaying improves either throughput or fairness or both. Our queue aware scheduler AOS is not improved through relaying in terms of throughput as much as non-opportunsitic schedulers since poor channel users are not selected frequently, but yet the performance is enhanced.

The practical implementation of our scheduling algorithms requires monitoring of the load at each user queue and the channel conditions. Hence, scheduling decisions can adapt to varying channel and traffic conditions as long as channel state information and queue states can be easily, continuously observed.

7. References

Bianchi, G. (2000) . Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal Selected Areas in Communications*, 18,3, March 2000, 535 – 547.

Bicket, J.; Aguayo, D. ;Biswas, S. & Morris, R. (2005) Architecture and Evaluation of an Unplanned 802.11b Mesh Network. *MobiCom'05*, Cologne, Germany, August 2005

Bolcskei, H.; Gesbert, D. & Paulraj, A. J. (2002). On the Capacity of OFDM-Based Spatial Multiplexing Systems. *IEEE Transactions on Communications*, 50, 2, February 2002, 225-234.

Boyer, J.;Falconer, D.D. & Yanikomeroglu, H., (2004). Multihop diversity in wireless relaying Channels. *IEEE Transactions on Communications*, 52, 10, October 2004, 1820-1830.

Ciftcioglu, E.N. & Gurbuz, O. (2006). Opportunistic Scheduling with Frame Aggregation for Next Generation Wireless LANs. *IEEE International Conference on Communications (ICC) 2006*, pp. 5228-5233, Istanbul, Turkey, June 2006.

Ciftcioglu, E.N. & Gurbuz, O. (2007). Access Scheduling Based on Time Water-Filling for Next Generation Wireless LANs. *IEEE Vechicular Technology Conference (VTC)* Spring 2007, pp.2966-2970, Dublin, Ireland, April 2007.

Cover, T.M. & Thomas, J.A. (1991). *Elements of Information Theory.* Wiley, New York.

Erceg, V. et al. (2004). TGn Channel Models. *IEEE 802.11 - 03/940r4*, May 2004.

Jalali, A.; Padovani, R. & Pankaj, R. (2000). Data Throughput of CDMA-HDR: A High Efficiency-High Data Rate Personal Communication Wireless System. *Proceedings of IEEE Vehicular Technology Conference (VTC) Spring 2000*, pp. 1854-1858, Tokyo, Japan, May 2000.

Kleinrock, L. (1975) *Queuing Systems, Volume I: Theory*. Wiley-Interscience, New York.

Knopp, R. & Humblet, P. (1995). Information capacity and Power Control in Single Cell Multi-user Communications. *Proceedings of IEEEInternational Conference on Communications (ICC) 1995*, pp. 331-335, Seattle, USA. June 1995.

Leon Garcia, A. & Widjaja, I. (2004) *Communication Networks: Fundamental Concepts and Key Architectures. Second Edition*, McGraw Hill.

Liu, C. & Stephens, A. P. (2005). An analytic model for infrastructure WLAN capacity with bidirectional frame aggregation. *IEEE Wireless Communications and Networking Conference (WCNC) 2005, pp. 113-119, New Orleans,, USA,* March 2005.

Mujtaba, S.A. (2004). TGn Sync Proposal Technical Specification 3. *TGn Sync Technical Proposal R00*, August 13, 2004.

Navda, V.; Kashyap, A. & Das, S. R. (2005) Design and Evaluation of iMesh: an Infrastructure-mode Wireless Mesh Network. *IEEE Symposium on a World of Wireless, Mobile ans Multimedia(WOWMOM),* pp. 164-170, Taormina, Italy, June 2005.

Neely, M.; Modiano, E. & Rohrs, C.(2002) .Power and server allocation in a multi-beam satellite with time varying channels," *Proceedings of IEEE Infocom 2002*, pp. 1451–1460. New York, USA, June 2002.

Rappaport, T.S. (2002). *Wireless Communications: Principles and Practice, Second Edition*, Prentice Hall, Upper Saddle River, NJ.

Sadeghi, B.; Kanodia, V.; Sabharwal, A. & Knightly, E. (2002). Opportunistic Media Access for Multirate Ad Hoc Networks. *Proceedings of ACM MOBICOM 2002*, 24-35, Atlanta, USA, September 2002.

Schrage, L.E. & Miller, L.W. (1966), The queue M/G/1 with the shortest remaining processing time discipline. *Operations Research*,14, August 1966, 670-684.

Sreng, V.; Yanikomeroglu, H., & Falconer, D.D. (2002). Coverage enhancement through two-hop relaying in cellular radio systems. *IEEE Wireless Communications and Networking Conference(WCNC) 2002,* pp. 881-885, Orlando, USA, March 2002.

Telatar, I. E. (1999). Capacity of multi-antenna Gaussian Channels. *European Transactions on Telecommunications,* 10, 6, November 1999, 585-595.

Tinnirello, I. & Choi, S. (2005). Efficiency Analysis of Burst Transmissions with Block ACK in Contention-Based 802.11e WLANs. *Proceedings of IEEE International Conference on Communications(ICC) 2005*, pp. 3455-3460, Seoul, Korea, May 2005.

Zafer, M. & Modiano, E. (2005). A calculus approach to minimum energy transmission policies with quality of service guarantees. *Proceedings of IEEE Infocom 2005*, pp. 548-559 , Miami, USA, March 2005

22

Parallel Greedy Approximation on Large-Scale Combinatorial Auctions

Naoki Fukuta[1] and Takayuki Ito[2,3]
[1]*Shizuoka University,*
[2]*Nagoya Institute of Technology*
[3]*Massachusetts Institute of Technology*
[1,2]*Japan*
[3]*United States*

1. Introduction

Combinatorial auctions (Cramton et al., 2006) are auctions that allow bidders to place bids for a set of items. Combinatorial auctions provide suitable mechanisms for efficient allocation of resources to self-interested attendees (Cramton et al., 2006). Therefore, many works have been done to utilize combinatorial auction mechanisms for efficient resource allocation. For example, the FCC tried to employ combinatorial auction mechanisms for assigning spectrums to companies (McMillan, 1994).

On the other hand, efficient resource allocation is also becoming crucial in many computer systems that should manage resources efficiently, and combinatorial auction mechanisms are suitable for this situation. For example, considering a ubiquitous computing scenario, there is typically a limited amount of resources (sensors, devices, etc.) that may not cover all needs for all users. Due to certain reasons (physical limitations, privacy, etc.), most of the resources cannot be shared with other users. Furthermore, software agents will use two or more resources at a time to achieve desirable services for users. Of course, each software agent provides services to its own user, and the agent may be self-interested.

Tremendous research efforts have been done to improve many parts of combinatorial auctions. An example is recent efforts for winner determination problem. In general, the optimal winner determination problem of a combinatorial auction is NP-hard (Cramton et al., 2006) for the number of bids. Thus, much work focuses on tackling the computational costs for winner determination (Fujishima et al., 1999); (Cramton et al., 2006); (Sandholm et al., 2005). Also many efforts have been done for generic problem solvers that can be applied to solve winner determination problems.

However, in such ubiquitous computing scenarios, there is strong demand for completing an auction within a fine-grained time period without loss of allocation efficiency. In a ubiquitous computing scenario, the physical location of users may always be changing and that could be handled by the system. Also, each user may have multiple goals with different contexts, and those contexts are also dynamically changing. Therefore, resources should be re-allocated in a certain fine-grained period to keep up with those changes in a timely manner. For better usability, the time period of resource reallocation will be 0.1 to several

seconds depending on services provided there. Otherwise, resources will remain assigned to users who no longer need them while other users are waiting for allocation.
Also, in the above scenarios, it is very important to handle a large number of bids in an auction. Consider that if there are 256 resources and 100 agents, and each agent has 200 to 1000 bids, then there will be 20,000 to 100,000 bids for 256 items in an auction. However, it has been difficult to complete such a large-scale combinatorial auction within a very short time. Such hard time constraint even prevents algorithms to prepare a rich pre-processing to reach optimal results in (not very) short time.
Since greedy algorithm is so simple, it can be applied to such situations. However, a pure greedy algorithm typically provides lower optimality of results that are not satisfiable for applications. When we solve this issue, parallel greedy approach can be a good solution for this kind of problems. Furthermore, a simple greedy algorithm can be used to enforce results to satisfy desirable properties that are very important for both theoretical and practical reasons.
In this chapter, we describe how greedy algorithms can be effectively used in mechanism design, especially, on designing and implementing combinatorial auction mechanisms.

2. Combinatorial auctions and winner determination problem

2.1 Mechanism design and combinatorial auctions

An auction mechanism is an economic mechanism for efficient allocations of items to self-interested buyers with agreeable prices. When the auction mechanism is truthful, i.e., it guarantees incentive compatibility, the mechanism enforces the bidders to locate their bids with true valuations. In such auctions, since we have an expectation of obtaining bids with true valuations, we can allocate items to buyers efficiently even though some buyers may try to cheat the mechanisms out of gaining sufficient incomes from them. For example, Vickrey proposed an auction mechanism that has incentive compatibility (Vickrey, 1961). That is a basic difference from ordinary resource allocation mechanisms that have implicit assumptions of truth-telling attendees.
Combinatorial auction is an auction mechanism that allows bidders to locate bids for a bundle of items rather than single item (Cramton et al., 2006). Combinatorial auction has been applied for various resource allocation problems. For example, McMillan et al. reported a trial on an FCC spectrum auction (McMillan, 1994). Rassenti et al. reported a mechanism for an airport time slot allocation problem (Rassenti et al., 1982). Ball et al. discussed applicability of combinatorial auctions to airspace system resource allocations (Ball et al., 2006). Caplice et al. proposed a bidding language for optimization of procurement on freight transportation services (Caplice et al., 2004). Estelle et al. proposed a formalization on auctioning London Bus Routes (Cantillon & Pesendorfer, 2004). Hohner et al. presented an experience on procurement auctions at a software company (Hohner et al., 2003).
However, on emerging applications with such resource allocation problems, their problem spaces are larger, more complex, and much harder to solve compared to previously proposed applications. For example, Orthogonal Frequency Division Multiple Access (OFDMA) technology enables us to use a physically identical frequency bandwidth as virtually multiplied channels at the same time, and this causes the channel allocation problem to become more difficult (Yang & Manivannan, 2005). Also some recent wireless technologies allow us to use multiple channels on the same, or different physical layers (i.e, WiFi, WiMax, and Bluetooth at the same time) for attaining both peak speed and robust connectivity (Salem et al., 2006); (Niyato and Hossain, 2008). Furthermore, such resource

allocation should be done for many ordinary users rather than a fixed limited number of flights or companies. Also the contexts of users, which are dynamically changing through the time, should be considered in the allocation.

In this chapter, to maintain simplicity of discussion, we focus on utility-based resource allocation problems such as (Thomadakis & Liu, 1999), rather than generic resource allocation problems with numerous complex constraints. The utility-based resource allocation problem is a problem that aims to maximize the sum of utilities of users for each allocation period, but does not consider other factors and constraints (i.e., fair allocation (Sabrina et al., 2007); (Andrew et al., 2008), security and privacy concerns (Xie & Qin, 2007), uncertainty (Xiao et al., 2004), etc).

Also, throughout this chapter, we only consider auctions that are single-sided, with a single seller and multiple buyers to maintain simplicity of discussion. It can be extended to the reverse situation with a single buyer and multiple sellers, and the two-sided case. The two-sided case is known as the combinatorial exchange. In the combinatorial exchange mechanisms, multiple sellers and multiple buyers are trading on a single trading mechanism. About this mechanism, the process of determining winners is almost the same as single-sided combinatorial auctions. However, it is reported that the revenue division among sellers can be a problem. There are a lot of interesting studies on combinatorial exchange (Parkes et al, 2005).

2.2 Winner determination problem

An important issue on combinatorial auction is representation of bids. In this chapter, we use OR bid representation(Lehmann et al., 2006), a simplest one in major formalisms.

On OR bid representation, the winner determination problem on combinatorial auction WDP_{OR} is defined as follows (Cramton et al., 2006): The set of bidders is denoted by $N=\{1,...,n\}$, and the set of items by $M=\{m_1,...,m_k\}$. $|M|=k$. Bundle S is a set of items:$S \subseteq M$. We denote by $v_i(S)$, bidder i's valuation of the combinatorial bid for bundle S. An allocation of the items is described by variables $x_i(S) \in \{0,1\}$, where $x_i(S)=1$ if and only if bidder i wins bundle S. An allocation, $x_i(S)$, is feasible if it allocates no item more than once,

$$\sum_{i \in N} \sum_{S \ni j} x_i(S) \leq 1$$

for all $j \in M$.

The winner determination problem is the problem to maximize total revenue

$$\max_{X} \sum_{i \in N, S \subseteq M} v_i(S) x_i(S)$$

for feasible allocations $X \ni x_i(S)$.

Fig. 1 shows an example of WDP_{OR}. Consider there are three items *a*, *b*, and *c*, and three bidders *Alice*, *Bob*, and *Charles*. *Alice* bids 10 for *a*. *Bob* bids 20 for {*b, c*}. *Charles* bids 18 for {*a, b*}. The problem is to choose winners of this auction from those three bids. Here, to choose *Alice*'s and *Charles*'s, or *Bob*'s and *Charles*'s are infeasible allocation, since both *Alice*'s and *Charles*'s include item *a*, and both *Bob*'s and *Charles*'s include item *b*. The optimal allocation is *a* for *Alice*, and *b* and *c* for *Bob*.

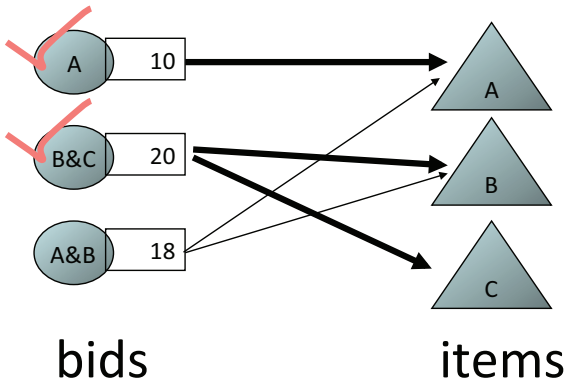


Fig. 1. Winner Determination Problem

Since the winner determination problem WDP_{OR} is a combinatorial optimization problem, it is generally NP-hard(Cramton et al., 2006). Furthermore, winner determination also plays important roles in other parts of combinatorial auction mechanism. For example, some combinatorial auction mechanisms (e.g., VCG, etc.) require many times of winner determination for slightly different bids for pricing mechanism. Therefore, it is strongly demanded to solve the problem in tractable way. In this chapter, we focus on solving this problem.

2.3 Lehmann's greedy winner determination

Lehmann et al. proposed a combinatorial auction mechanism that preserves truthfulness, a very important desirable property, while it uses a greedy approximation algorithm for its winner determination(Lehmann et al., 2002).

Lehmann's greedy algorithm (Lehmann et al., 2002) is a very simple but powerful linear algorithm for winner determination in combinatorial auctions. Here, we denote a bid b=<s,a>, such that $S \subseteq M$ and $a \in \mathcal{R}_+$. Two bids b=<s,a> and b'=<s',a'> conflict if and only if $s \cap s' \neq \emptyset$. The greedy algorithm can be described as follows. (1) The bids are sorted by some criterion. In (Lehmann et al., 2002), Lehmann et al. proposed sorting list L by descending average amount per item. More generally, they proposed sorting L by a criterion of the form $a/|s|^c$ for some number $c \geq 0$, possibly depending on the number of items, k. (2) A greedy algorithm generates an allocation. L is the sorted list in the first phase. Walk down the list L, allocates items to bids whose items are still unallocated.

Example: Assume there are three items a, b, and c, and three bidders *Alice*, *Bob*, and *Charles*. *Alice* bids 10 for a. *Bob* bids 20 for {b,c}. *Charles* bids 18 for {a,b} (Fig. 2 Step1). We sort the bids by the criterion of the form $a/|s|^{0.5}$ (Fig. 2 Step2). *Alice*'s bid is calculated as $10/1^{0.5}$=10. *Bob*'s bid is calculated as $20/2^{0.5}$=14 (approximately). *Charles*'s bid is calculated as $18/2^{0.5}$=13 (approximately). The sorted list is now *Bob*'s bid <{b,c},20>, *Charles*'s bid <{a,b},18>, and *Alice*'s bid <{a}, 10>. The algorithm walks down the list (Fig. 2 Step3). At first, *Bob* wins {b,c} for 20. Then, *Charles* cannot get the item because his bid conflicts with *Bob*'s bid. Finally, *Alice* gets {a} for 10.

Lehmann's greedy algorithm provides a computationally tractable combinatorial auction. However, it has two remaining issues: (1)efficiency of item assignment, and (2)adjustment of good bid weighting parameter c. In the next section, we describe possible approaches for these issues.

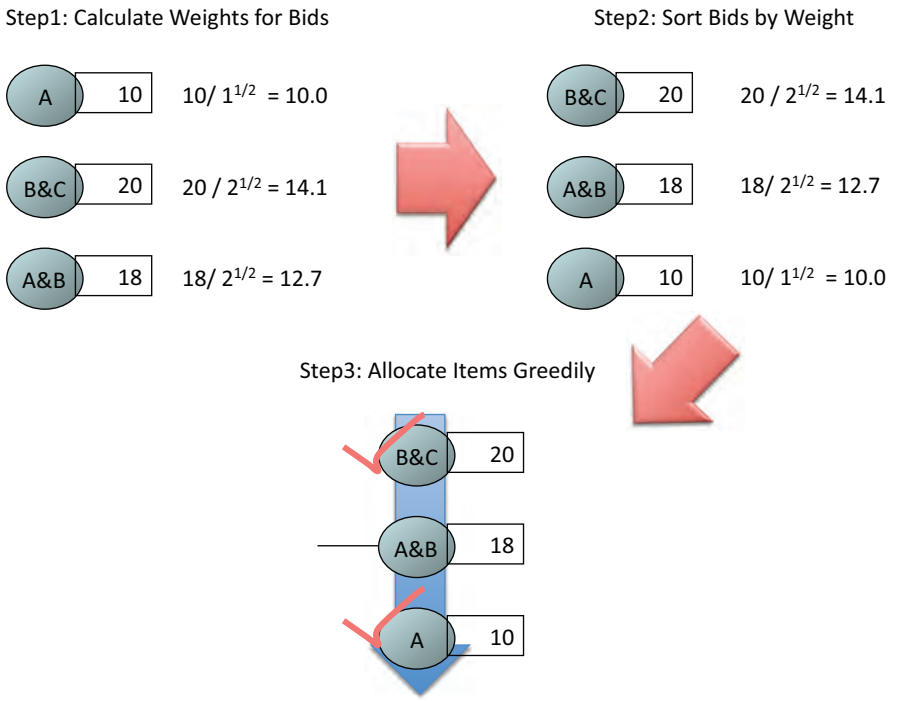


Fig. 2. Lehmann's Greedy Allocation

3. Parallel greedy approximation

3.1 Incremental updating

In (Fukuta & Ito, 2006), we have shown that the hill-climbing approach performs well when an auction has a massively large number of bids. In this section, we summarize our proposed algorithms for incremental updating solutions.

Lehmann's greedy winner determination could succeed in specifying the lower bound of the optimality in its allocation (Lehmann et al., 2002). A straightforward extension of the greedy algorithm is to construct a local search algorithm that continuously updates the allocation so that the optimality is increased. Intuitively, one allocation corresponds to one state of a local search.

List 1 shows the algorithm. The inputs are *Alloc* and *L*. *L* is the bid list of an auction. *Alloc* is the initial greedy allocation of items for the bid list.

The function ***consistentBids*** finds consistent bids for the set ***NewAlloc*** by walking down the list ***RemainBids***. Here, a new inserted bid will wipe out some bids that conflict with the inserted bid. So there will be free items to allocate after the insertion. The function ***consistentBids*** tries to insert the other bids greedily for selling as many of the items as possible. When the total price for ***NewAlloc*** is higher than ***Alloc***, current allocation is updated to ***NewAlloc*** and the function continues updating from ***NewAlloc***. We call this as *Greedy Hill Climbing*(GHC) in this chapter.

```
1: function GreedyHillClimbingSearch(Alloc, L)
2:   RemainBids:= L - Alloc;
3:   for each b ∈ RemainBids as sorted order
4:     if b conflicts Alloc then
5:       Conflicted:=Alloc - consistentBids({b}, Alloc);
6:       NewAlloc:= Alloc - Conflicted + {b};
7:       ConsBids:=
8:          consistentBids(NewAlloc, RemainBids);
9:       NewAlloc:=NewAlloc+ConsBids;
10:    if price(Alloc) < price(NewAlloc) then
11:      return GreedyHillClimbingSearch(NewAlloc,L);
12:  end for each
13:  return Alloc
```

List. 1. Greedy Hill Climbing Algorithm

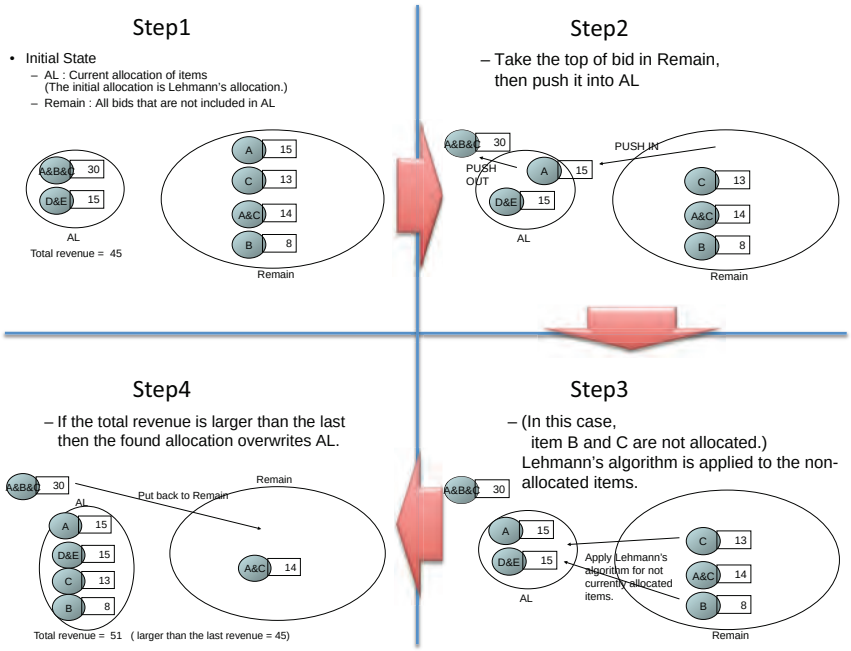


Fig. 3. Example of Greedy Hill Climbing

Example: Assume there are five items *a*, *b*, *c*, *d*, and *e*, and there are six bids, <{*a*,*b*,*c*},30>, <{*a*},15>, <{*c*},13>, <{*d*,*e*},15>, <{*a*,*c*},14>, and <{*b*},8>. We can calculate the values of Lehmann's criterion $a/|s|^{0.5}$ as 17.6, 15, 13, 10.7, 10, and 8, respectively. In this case, the initial allocation is Lehmann's greedy allocation <{*a*,*b*,*c*},30>, <{*d*,*e*},15> and the total revenue is 45. Here, the remaining list contains <{*a*},15>, <{*c*},13>, <{*a*,*c*},14>, and <{*b*},8> (Fig. 3,

Step1). In this algorithm, we pick <{*a*},15> since it is the top of the remaining list. Then we insert <{*a*},15> into the allocation and remove <{*a,b,c*},30>. The allocation is now <{*a*},15>, <{*d,e*},15> (Fig. 3, Step2). We then try to insert the other bids that do not conflict with the allocation (Fig. 3, Step3). Then, the allocation becomes <{*a*},15>, <{*b*},8>, <{*c*},13>,<{*d,e*},15>. The total revenue is 51, and is increased. Thus, the allocation is updated to it (Fig. 3, Step4). Our local algorithm continues to update the allocation until there is no allocation that has greater revenue. This could improve the revenue that Lehmann's greedy allocation can achieve.

To show the advantages of greedy incremental updating, we also prepared an ordinary Hill-Climbing local search algorithm. List.2. shows the algorithm. The difference to above is to choose *best* alternatives in each climbing step, instead of choosing it greedily. We call this as *Best Hill Climbing*(BHC) in this chapter.

```
1:  function BestHillClimbingSearch(Alloc, L)
2:    MaxAlloc := φ
3:    RemainBids:= L - Alloc;
4:    for each b ∈ RemainBids as sorted order
5:      if b conflicts Alloc then
6:        Conflicted:=Alloc - consistentBids({b}, Alloc);
7:        NewAlloc:= Alloc - Conflicted + {b};
8:        ConsBids:=
9:          consistentBids(NewAlloc, RemainBids);
10:       NewAlloc:=NewAlloc+ConsBids;
11:     if price(MaxAlloc) < price(NewAlloc) then
12:       MaxAlloc := NewAlloc;
13:   end for each
14:   if price(Alloc) < price(MaxAlloc) then
15:     return BestHillClimbingSearch(MaxAlloc,L);
16:   return Alloc
```

List. 2. Best Hill Climbing Algorithm

3.2 Parallel search for multiple weighting strategies

The optimality of allocations got by Lehmann's algorithm (and the following hill-climbing) deeply depends on which value was set to *c* in the bid weighting function. Again, in (Lehmann et al., 2002), Lehmann et al. argued that *c=1/2* is the best parameter for approximation when the norm of the worst case performance is considered. However, optimal value for approximating an auction is varied from 0 to 1 depending on the auction problem.

For example, when we choose *c=1* in the example in section 3.1, we can get better results directly at the time of initial Lehmann's greedy allocation (Fig. 4).

In (Fukuta & Ito, 2006), we presented an initial idea of an enhancement for our incremental updating algorithm to parallel search for different bid weighting strategies (e.g, doing the same algorithm for both *c=0* and *c=1*).

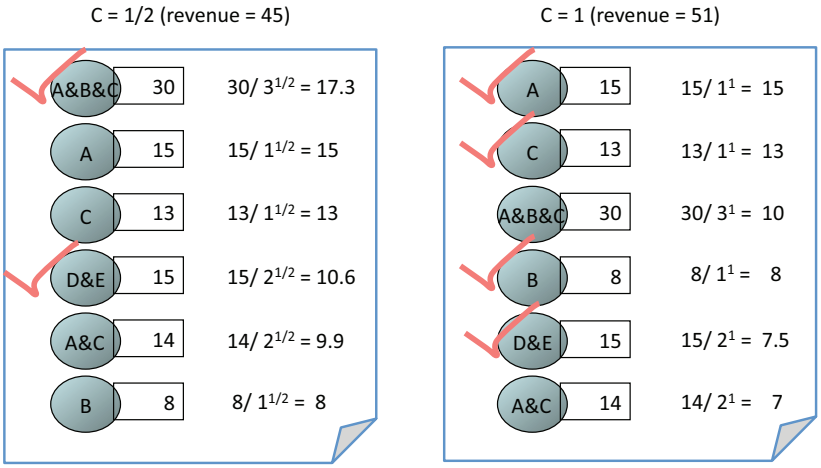


Fig. 4. Effects of Bid Weighting Strategy

3.3 Simulated annealing search

We also prepared a small extension of the shown algorithm to the simulated annealing local search(Fukuta & Ito, 2006). The algorithm is a combination of the presented hill-climbing approach and a random search based on the standard simulated annealing algorithm. We use a parameter that represents the temperature. The temperature is set at a high value at the beginning and continuously decreased until it reaches 0. For each cycle, a neighbour is randomly selected and its value may be less than the current value in some cases. Even in such a case, if a probability value based on the temperature is larger than 0, the state is moved to the new allocation that has less value. This could make us get off the local minimum.

We prepared this algorithm only for investigating how random search capability will improve the performance. Note that the proposed SA search may not satisfy our proposed features discussed later.

4. Experimental analysis

4.1 Experiment settings

In this section, we compare our algorithms to other approaches in various datasets. Details about other approaches are presented in section 5.

We implemented our algorithms in a C program for the following experiments. We also implemented the Casanova algorithm(Hoos & Boutilier, 2000) in a C program. However, for the following experiments, for Zurel's algorithm we used Zurel's C++ based implementation that is shown in (Zurel & Nisan, 2001). Also we used CPLEX Interactive Optimizer 11.0.0 (32bit) in our experiments.

The experiments were done with the above implementations to examine the performance differences among algorithms. The programs were employed on a Mac with Mac OS X 10.4, CoreDuo 2.0GHz CPU, and 2GBytes of memory. Thus, actual computation time will be much smaller when we employ parallel processor systems in a distributed execution environment.

We conducted several experiments. In each experiment, we compared the following search algorithms. greedy(c=0.5) uses Lehmann's greedy allocation algorithm with parameter (*c=0.5*). greedy-N uses the best results of Lehmann's greedy allocation algorithm for N different weighting parameters ($0 \leq c \leq 1$). *HC(c=0.5) uses a local search in which the initial allocation is Lehmann's allocation with *c=0.5* and conducts one of hill-climbing searchs (e.g., GHC or BHC) shown in the previous section. Similarly, *HC-N uses the best results of a hill-climbing search (e.g., GHC or BHC) for N different weighting parameters ($0 \leq c \leq 1$). For example, GHC-11 means the best result of greedy hill-climbing(GHC) with parameter c = {0, 0.1,...,0.9, 1}. SA uses the simulated annealing algorithm presented in (Fukuta & Ito, 2006). Also, we denote the Casanova algorithm as casanova and Zurel's algorithm as Zurel.

In the following experiments, we used 0.2 for the epsilon value of Zurel's algorithm in our experiments. This value appears in (Zurel & Nisan, 2001). Also, we used 0.5 for *np* and 0.15 for *wp* on Casanova, which appear in (Hoos & Boutilier, 2000). Note that we set *maxTrial* to 1 but *maxSteps* to ten times the number of bids in the auction.

4.2 Evaluation on basic auction dataset

In (Zurel & Nisan, 2001), Zurel et al. evaluated the performance of their presented algorithm with the data set presented in (de Vries & Vohra, 2003), compared with CPLEX and other existing implementations.

In (Fukuta & Ito, 2007a), we presented comparison of our algorithms, Casanova, and Zurel's algorithm with the dataset provided in (de Vries & Vohra, 2003). This dataset contains 2240 auctions with optimal values, ranging from 25 to 40 items and from 50 to 2000 bids. Since the data set is small, we omit details in this chapter.

We conducted detailed comparisons with common datasets from CATS benchmark(Leyton-Brown et al., 2000). Compared to deVries' dataset shown in (de Vries & Vohra, 2003), the CATS benchmark is very common and it contains more complex and larger datasets.

Fig. 5 shows the comparison of our algorithms, Casanova, and Zurel's algorithm with a dataset provided in the CATS benchmark (Leyton-Brown et al., 2000). The dataset has numerous auctions with optimal values in several distributions. Here we used ***varsize*** which contains a total of 7452 auctions with reliable optimal values in 9 different distributions[1]. Numbers of items range from 40 to 400 and numbers of bids range from 50 to 2000.

Since problems in the dataset have relatively small size of bids and items, we omitted the execution time since all algorithms run in very short time. Here, we can see that the performances of GHC-11 and SA are better than Zurel's on average optimality.

Note that those differences come from the differences of the termination condition on each algorithm. In particular, Casanova spent much more time compared with the other two algorithms. However, we do not show the time performance here since the total execution time is relatively too small to be compared.

---

[1] Since some of the original data seems corrupted or failed to obtain optimal values, we excluded such auction problems from our dataset. Also, we excluded a whole dataset of a specific bid distribution when the number of valid optimal values is smaller than the other half of the data. The original dataset provides optimal values of auction problems by two independent methods, CASS and CPLEX. Therefore, it is easy to find out such corrupted data from the dataset.

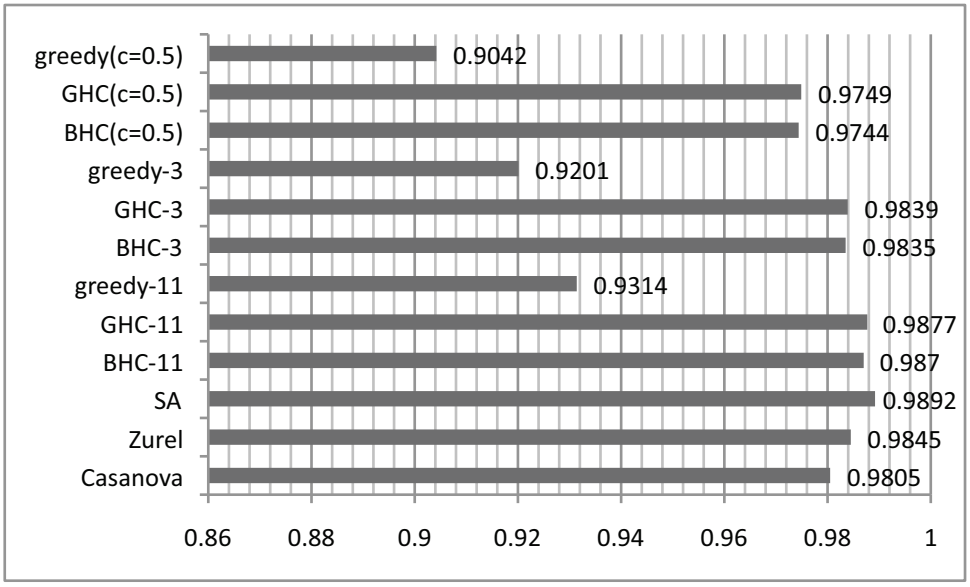


Fig. 5. Optimality on CATS-VARSIZE dataset

Here, we can see the performance of both greedy, GHC, and BHC increases when we use more threads to parallel search for multiple weightings. For example, the result of GHC-3 is better than GHC(c=0.5) and GHC-11 is slightly better in the average. It shows that our parallel approximation approach will increase the performance effectively even when the number of parallel executions is small.

Also we compared the performance on our greedy local updating approach (GHC) with ordinary best updating approach (BHC). Surprisingly, the average performances of GHC are slightly better than BHC, regardless of using parallel search. This is because the BHC approach is still heuristic one so it does not guarantee the choice is best for global optimization. Also we think we found a very good heuristic bid weighting function for our greedy updating.

4.3 Evaluation on large auction dataset

The CATS common datasets we used in Section 4.2 have a relatively smaller number of bids than we expected. We conducted additional experiments with much greater numbers of bids. We prepared additional datasets having 20,000 non-dominated bids in an auction. The datasets were produced by CATS (Leyton-Brown et al., 2000) with default parameters in 5 different distributions. In the datasets, we prepared 100 trials for each distribution. Each trial is an auction problem with 256 items and 20,000 bids[2].

[2] Due to the difficulty of preparing the dataset, we only prepared 5 distributions. For more details about the bid generation problem, see (Leyton-Brown et al., 2000). A preliminary result of this experiment was shown in (Fukuta & Ito, 2007b).

Fig. 6 (6a and 6b) shows the experimental result on the datasets with 20,000 bids in an auction focused on execution time of approximation. Due to the difficulty of attaining optimal values, we normalized all values as Zurel's results equaling 1 as follows.
Let A be a set of algorithms, $z \in A$ be the Zurel's approximation algorithm, L be a dataset generated for this experiment, and $revenue_a(p)$ such that $a \in A$ be the revenue obtained by algorithm a for a problem p such that $p \in L$, the average revenue ratio $ratioA_a(L)$ for algorithm $a \in A$ for dataset L is defined as follows:

$$ratioA_a(L) = \frac{\sum_{p \in L} revenue_a(p)}{\sum_{p \in L} revenue_z(p)}$$

Here, we use $ratioA_a(L)$ for our comparison of algorithms.

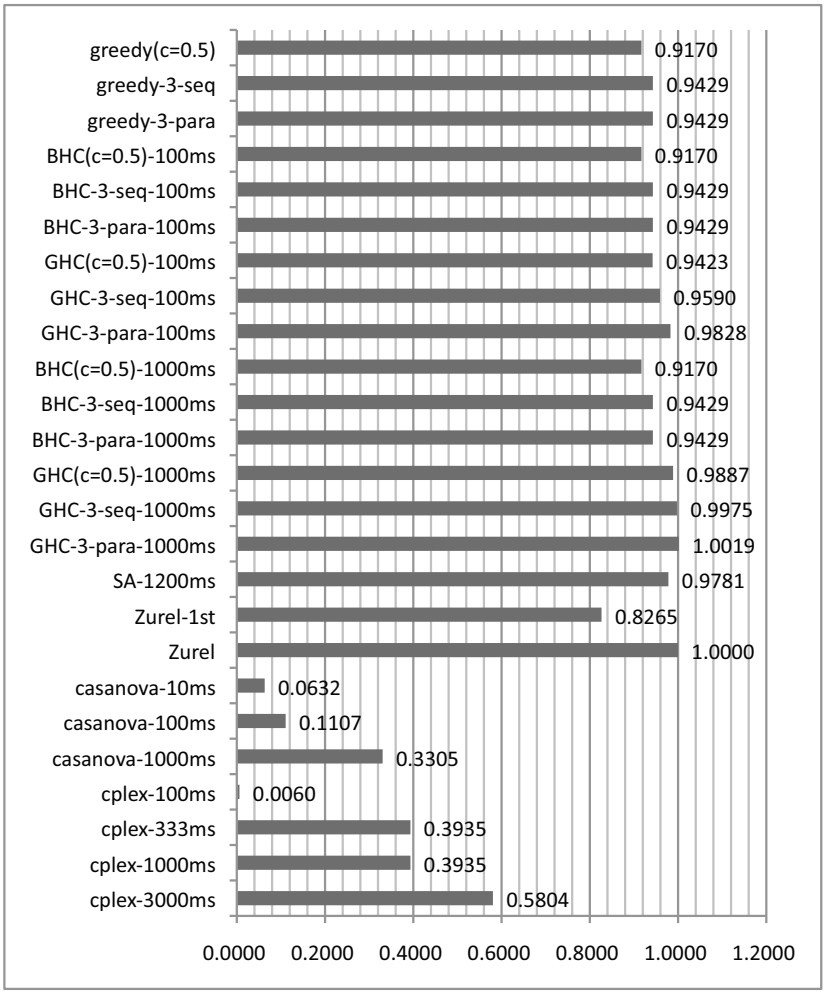


Fig. 6a. Time Performance on 20,000 bids- 256 items (Optimality Ratio)

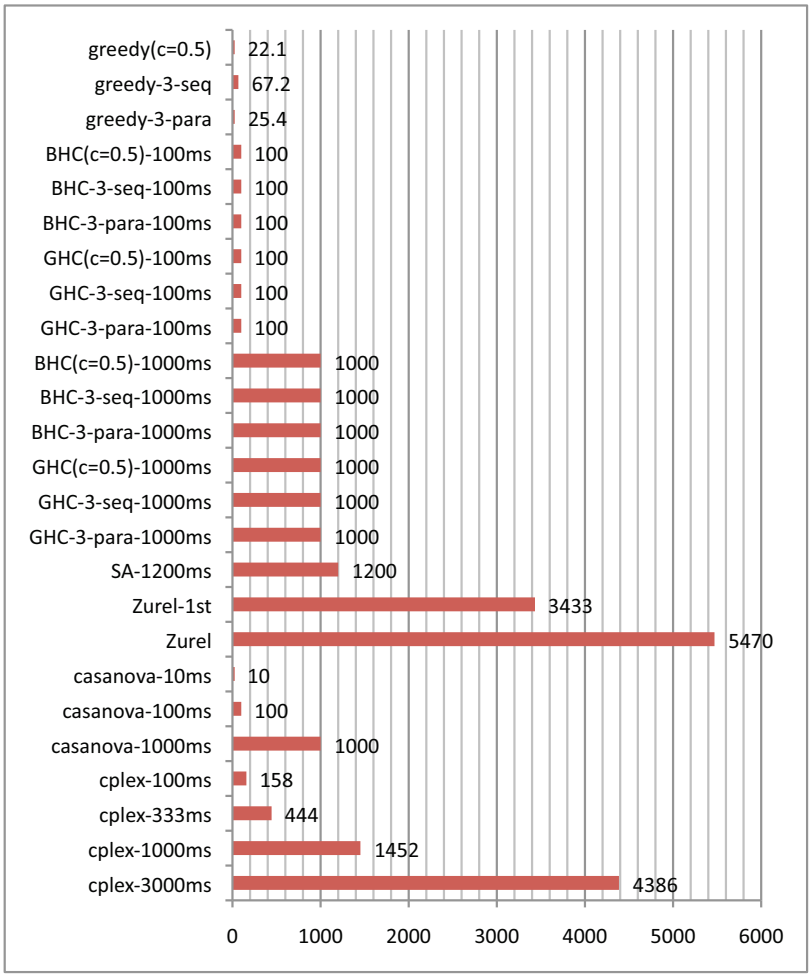


Fig. 6b. Time Performance on 20,000 bids - 256 items (Elapsed Time[msec])

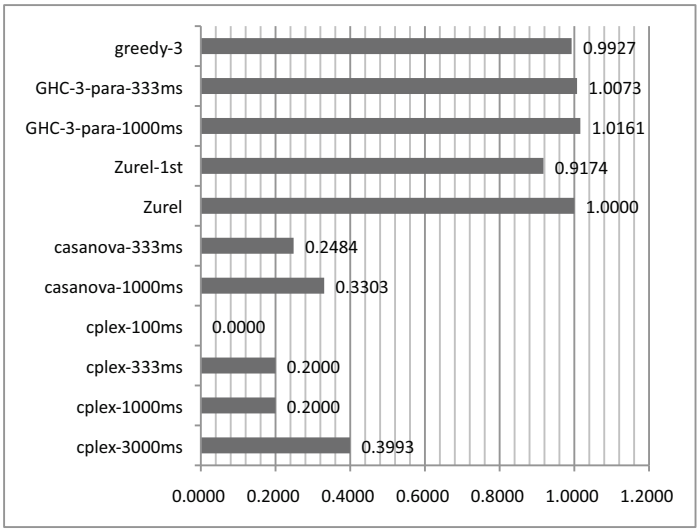
We prepared cut-off results for Casanova and HC. For example, casanova-10ms denotes the result of Casanova within 10 milliseconds. Here, for faster approximation, we used greedy-3, GHC-3, and BHC-3 but did not use greedy-11, GHC-11, and BHC-11. Here, greedy-3 uses the best results of Lehmann's greedy allocation algorithm with parameter ($0 \leq c \leq 1$ in 0.5 steps). GHC-3 and BHC-3 use the best results of the local updating with parameter ($0 \leq c \leq 1$ in 0.5 steps). Also, we prepared a variant of our algorithm that has a suffix of -seq or -para. The suffix -seq denotes the algorithm is completely executed in a sequence that is equal to one that can be executed on a single CPU computer. For example, greedy-3-seq denotes that the execution time is just the sum of execution times of three threads. The suffix -para denotes the algorithm is completely executed in a parallel manner, and the three independent threads are completely executed in parallel. Here, we used the ideal value for -para since our computer has only two cores in the CPU. The actual execution performance

will be between -seq and -para. Also, we denote the initial performance of Zurel's algorithm as Zurel-1st. Here, Zurel-1st is the result at the end of its first phase and no winners will be approximately assigned before it. cplex is the result of CPLEX with the specified time limit.
On most distributions in Fig. 6, Zurel-1st takes more than 1 second but the obtained *ratioA* is lower than greedy-3-seq. Furthermore, the average *ratioA* of GHC-3-para-1000ms is higher than Zurel while its computation time is less than both Zurel and Zurel-1st.
In Fig. 6, BHC could not get any update within the time limit so there is no update from greedy. Here, although SA performs better than greedy(C=0.5), it could not outperform GHC(C=0.5) in any case. Therefore, we can see that both *best-updating* and *random-updating* approaches are not sufficient enough for extremely short time approximation, although the *greedy-updating* approach makes a good performance in the same situation.
In many settings of CPLEX, the values are 0. This is because CPLEX could not generate initial approximation result within the provided time limit. Only datasets for two bid distributions have non-zero results for CPLEX. However, CPLEX spends around 400 msec for the computation but the results are still lower than greedy-3. On a dataset for another bid distribution, CPLEX could prepare results in 3.8 sec of computation, however, the result is still lower than greedy-3. This is because the condition we set up gave extremely short time limit so therefore CPLEX could not generate sufficient approximation results in such hard time constraint.
Fig. 7 shows the experimental result on the dataset with 100,000 bids in an auction focused on the early anytime performance. While GHC-3 and Zurel's algorithm are competitive in Fig. 6, it is clear that our proposed GHC-3 outperforms Zurel's algorithm in any time performance in Fig. 7. Note that, for Zurel's algorithm, the time needed to attain initial allocations increased approx. six times when the number of bids becomes five times larger than that of Fig. 6. However, while our GHC-3-para-1000ms only takes the same execution time (i.e, 1000 msec) for larger dataset, its average *ratioA* is higher than Zurel. Note that the GHC-3-para-333ms has still higher *ratioA* value than Zurel while its average computation time is 100 times less. We argue that our algorithm has an advantage when the number of bids increases.

5. Related work

5.1 Approaches for optimization problems

There are really many approaches to optimization problems. Linear programming is one of the well-known approaches in this area. The winner determination problem on combinatorial auctions can be transformed into a linear programming problem. Therefore, it is possible to use a linear programming solver for the winner determination problem.
CPLEX is a well-known, very fast linear programming solver system. In (Zurel & Nisan, 2001), Zurel et al. evaluated the performance of their presented algorithm with many data sets, compared with CPLEX and other existing implementations. While the version of CPLEX used in (Zurel & Nisan, 2001) is not up-to-date, the shown performance of Zurel's algorithm is approximately 10 to 100 times faster than CPLEX. In this chapter, we showed direct comparisons to the latest version of CPLEX we could prepare. Our approach is far better than latest version of CPLEX for large-scale winner determination problems. Therefore, the performance of our approach is competitive enough with CPLEX or other similar solver systems. This is natural since Zurel's and our approaches are specialized for combinatorial auctions, and also focus only on faster approximation but do not seek optimal



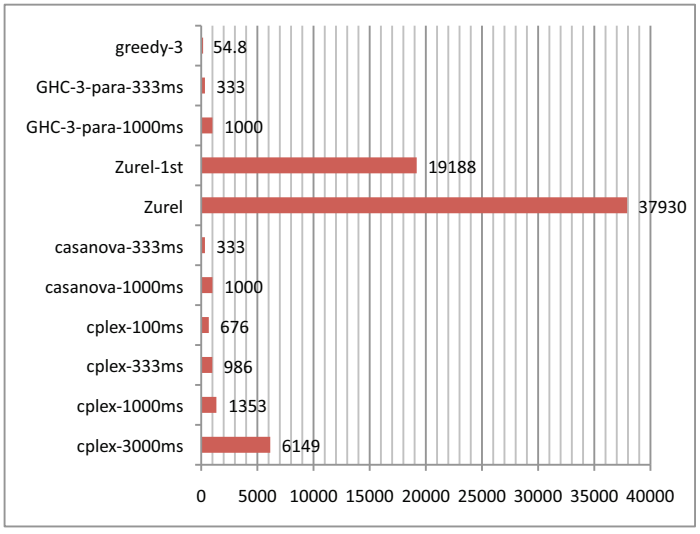


Fig. 7. Time Performance on 100,000bids - 256items

solutions. In case we need optimal solutions, it is good choice to solve the same problem by both our approach and CPLEX in parallel. This could improve anytime performance but guarantee obtaining optimal solutions. Even in such case, our approach should spend very small computation overhead.

Random-walk search is also a strong approach for approximating combinatorial optimization problems. There have been many algorithms proposed based on random-walk search mechanisms. In (Hoos & Boutilier, 2000), Casanova was proposed, which applies a

random walk SAT approach for approximating the winner determination problem in combinatorial auctions. In this chapter, we showed that our approach outperforms Casanova when the time constraint is very hard but the problem space is really large.
Simulated Annealing (SA) is another similar approach. We prepared an SA-based extension for our approach and we confirmed it increases the performance when the problem size is relatively small. However, SA needs random-walk in the early stage of its search and it decreases performance on short-time approximation.
Genetic Algorithm is another similar approach. In (Avasarala et al., 2006), Avasarala et al. proposed an approach for the winner determination problem on combinatorial auctions. However, in (Avasarala et al., 2006), they noticed that their algorithm is not effective for approximation in short time but is effective for obtaining higher optimal solutions with enough computation time. Random-walk searching is really effective approximation approach for combinatorial optimization problems. However, it is not effective when there are such hard time constraints. We focused on solving problems that are hard for such random-walk search approaches.

5.2 Approaches to obtain optimal solutions

There have been a lot of works on obtaining optimal solutions for winner determination in combinatorial auctions (de Vries & Vohra, 2003). For example, CABOB (Sandholm et al., 2005) and CASS (Fujishima et al., 1999) have been proposed by aiming to get the optimal allocations.
In (Hoos & Boutilier, 2000), it is shown that the Casanova algorithm outperforms approximation performance of CASS on winner determination. In this chapter, we showed that our approach outperforms Casanova in settings of a very large number of bids in an auction. Therefore, our approach should also outperform CASS in the same settings.
In (Sandholm et al., 2005), Sandholm et al. showed that CABOB outperforms CPLEX in several settings. However, according to our comparison, our algorithm should outperform CABOB in our settings. We argue that our approach is rather complementary to those algorithms that are seeking exact optimal solutions. It is not fair to compare their approximation performances when one guarantees obtaining optimal solutions but the other does not. Our approximation approach only covers large size problem settings that can only be handled by specialized approximation algorithms. Our approach does not contribute to advances in developing algorithms to obtain optimal solutions directly.

5.3 Other greedy approaches

Some researchers have noticed the better performance of simple greedy and incremental approaches for very large-scale problems. For example, (Sandholm, 2002) noticed the ease of approximation on very large auction problems. In (Lehmann et al., 2002), Lehmann et al. mentioned that a simple greedy approach obtains very high results when the auction problem is rather huge.
Also in (Kastner et al., 2002), Kastner et al. mentioned a potential capability of a simple incremental search approach to apply to very large auction problems and discussed the sensitivity for the number of bids in an auction. However, there is little mentioned about a detailed comparison of actual performances for several different types of datasets. In (Kastner et al., 2002), they only presented their preliminary experimental results on a dataset that is based on a single bid distribution.

Guo et al. (Guo et al., 2005) proposed similar local-search based algorithms and they argued that their approach is good for the settings of a large number of bids in a combinatorial auction problem. However, in (Guo et al., 2005), they presented very limited experimental results and little analysis or comparison to other high performance algorithms. Also in (Guo et al., 2005), they did not propose an idea that is similar to our multiple bid-weighting search. We argue that this multiple weighting search approach is very effective and that it distinguishes our approach from others. Also, we showed a detailed analysis of our experiments based on datasets generated by possible different bid distributions. We also showed direct comparisons to Zurel's approach presented in (Zurel & Nisan, 2001).

5.4 Other approaches

When we have some assumptions about models for valuation of bids, we can utilize those assumptions for better approximation. Dobzinski et al. proposed improved approximation algorithms for auctions with submodular bidders (Dobzinski & Schapira, 2006). Lavi et al, reported an LP-based algorithm that can be extended to support the classic VCG (Lavi & Swamy, 2005). Those studies mainly focused on theoretical aspects. In contrast to those papers, we rather focus on experimental analysis and implementation issues. Those papers did not present experimental analysis of the settings with a large number of bids as we presented in this chapter.

Using sequential auctions (Boutiler et al., 1999) is another approach to overcome the communication cost problem. Koenig et al. proposed a multiple-round auction mechanism that guarantees the upper bound of communication cost as fixed size *k*, that is independent from the number of agents or items in the auction (Koenig et al., 2007). Although our algorithm itself can approximate winners within a very short time with a huge number of updated bids, the communication cost problem remains.

6. Discussion

Lehmann's mechanism preserves truthfulness of the auction. However, since greedy incremental updating approach breaks *monotonicity*, an important property to provide truthfulness of auctions, the resulting auction will not be truthful. Detailed discussions and a counter example for *monotonicity* is presented in (Fukuta & Ito, 2007c). Therefore, another *monotonicity* has been proposed to approach this issue.

In real world auctions, often we open the winners and their bidding prices after the auction is finished. When we employ an approximated algorithm for winner determination, a loser who might be a winner in the optimal allocation could know the winner's bidding price in an approximate allocation after the auction finishes. In some cases, this loser had placed a higher price than the winner's for the same or a subset of the bundle. This would result in *unacceptable* allocations for bidders.

We believe that the above issue should be considered to make our mechanism acceptable by participants in the real world. Therefore, Winner-Price-Monotonicity and Weak-Winner-Price-Monotonicity are proposed to avoid *unacceptable* allocations(Fukuta & Ito, 2007a).

Definition 1. (Winner-Price-Monotonicity: WPM) *For two non-empty bundles B and B', if $B \subseteq B'$ and $v_i(B) > v_j(B')$, then j must not win bundle B'.*

Definition 2. (Weak-Winner-Price-Monotonicity: Weak-WPM) *For non-empty bundle B, if $v_i(B) > v_j(B)$, then j must not win bundle B.*

Here, proofs for following propositions are shown in (Fukuta & Ito, 2007a).

Proposition 1. *Our proposed winner determination algorithms, except for the simulated annealing-based algorithm, produce allocation W_{fin} that satisfies WPM when the algorithm reaches an end.*

Proposition 2. *In terms of any allocations that are achieved during computation (as an anytime algorithm), our proposed winner determination algorithms, except for the simulated annealing-based algorithm, satisfy Weak-WPM.*

It is a big merit to guarantee WPM and/or Weak-WPM at the algorithm level when we use it where slightly different combinatorial auctions are conducted iteratively. It seems easy to satisfy WPM and/or Weak-WPM by using any approximated winner determination algorithms by adding a pre-processing that removes all dominated bids from the bidset before starting the approximation. However, we should consider its computational overhead. For simplicity, consider a case $B = B'$ instead of $B \subseteq B'$. Let n be the number of items and m be the number of items in an auction. When m is very small, it is easy to look up the highest bids of each bundle by using a hash algorithm. In this case, the computational order is $O(n)$. However, it consumes a great deal of memory (of course it can be smaller than 2^m but at least additional $O(n)$ of working space), and it is actually very difficult to determine good hash functions for a smaller hash table size without loss of computational speed. It is a serious problem when the memory is almost completely used up for storing the data of a large number of bids. Sometimes its computational order might reach $O(n^2)$, which is greater than that of typical good approximation algorithms. For example, the computational order of Lehmann's greedy algorithm is $O(n \log n)$ when we use one of the $O(n \log n)$ sorting algorithms on it. Furthermore, when we consider the deletion of a bid, we have to determine the highest price bid that has been made obsolete by the deleted bid, or recalculate such pre-processing for all bids again. Considering a case $B \subseteq B'$ will make the problem more difficult. Since our algorithms guarantee Weak-WPM and WPM for the produced results, there is no need to prepare such additional pre-processing.

7. Conclusions

In this chapter, we presented how greedy approach can be used in combinatorial auctions. When we have hard time constraint and a large scale problem, greedy approach works very well compared to other approaches. Two different greedy approaches can be combined to improve performance. Also it is good idea to combine parallel search approach for greedy approximation algorithm. Furthermore, greedy-based approach is also helpful to keep the result of algorithm a certain desirable property, while other random search algorithms could not.

For further reading about combinatorial auctions, (Cramton et al., 2006) is a best book for both researchers and practitioners. For further reading about the shown approach, see (Fukuta & Ito, 2007a); (Fukuta & Ito, 2007b) for detailed performance analysis, and see (Fukuta & Ito, 2006); (Fukuta & Ito, 2007c); (Fukuta & Ito, 2007a) for theoretical issues and further discussions.

8. References

Andrew, L. L.H.; Hanly, S. V. & Mukhtar, R. G. (2008). Active queue management for fair resource allocation in wireless networks. *IEEE Transactions on Mobile Computing*, pages 231-246, Feb. 2008.

Avasarala, V.; Polavarapu, H.; & Mullen, T. (2006). An approximate algorithm for resource allocation using combinatorial auctions. *In Proc. of The 2006 WIC/IEEE/ACM International Conference on Intelligent AgentTechnology (IAT2006)*, pages 571-578, 2006.

Ball, M. O.; Donohue, G. L. & Hoffman, K. (2006). Auctions for allocation of airspace system resources. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions,* chapter 20, pages 507-538. The MIT Press, 2006.

Boutiler, C.; Goldszmidt, M.; & Sabata, B. (1999). Sequential auctions for the allocation of resources with complementarities. *In Proc. of International Joint Conference on Artificial Intelligence(IJCAI1999)*, pages 527-534, 1999.

Cantillon, E. & Pesendorfer, M. (2004). Combination bidding in multi-unit auctions. *Working Paper of Harvard Business School and London School of Economics*, 2004.

Caplice, C.; Plummer, C. & Sheffi, Y. (2004). Bidder behavior in combinatorial auctions for transportation services. *Working Paper of Massachusetts Institute of Technology Center for Transportation and Logistics*, 2004.

Cramton, P.; Shoham, Y. & Steinberg, R. (2006). *Combinatorial Auctions*. The MIT Press, 2006.

de Vries, S. & Vohra, R. V. (2003). Combinatorial auctions: A survey. *International Transactions in Operational Research*, 15(3):284-309, 2003.

Dobzinski, S. & Schapira, M. (2006). An improved approximation algorithm for combinatorial auctions with submodular bidders. I*n SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1064-1073. ACM Press, 2006.

Fujishima, Y.; Leyton-Brown, K. & Shoham, Y. (1999). Taming the computational complexity of combinatorial auctions: Optimal and approximate approarches. *In Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI99)*, pages 548-553, 1999.

Fukuta, N. & Ito, T. (2007a). Periodical resource allocation using approximated combinatorial auctions. *In Proc. of The 2007 WIC/IEEE/ACM International Conference on Intelligent Agent Technology (IAT2007)*, pages 434-441, 2007.

Fukuta, N. & Ito, T. (2007b). Short-time approximation on combinatorial auctions – a comparison on approximated winner determination algorithms. *In Proc. of The 3rd International Workshop on Data Engineering Issues in E-Commerce and Services (DEECS2007)*, pages 42-55, 2007.

Fukuta, N. & Ito, T. (2007c). Toward a large scale e-market: A greedy and local search based winner determination. *In Proc. of The 20th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE2007)*, pages 354-363, 2007.

Fukuta, N. & Ito, T. (2006). Towards better approximation of winner determination for combinatorial auctions with large number of bids. *In Proc. of The 2006 WIC/IEEE/ACM International Conference on Intelligent Agent Technology (IAT2006)*, pages 618-621, 2006.

Guo, Y. ; Lim, A. ; Rodrigues, B. & Zhu, Y. (2005). A non-exact approach and experiment studies on the combinatorial auction problem. *In Proc. of HICSS2005*, page 82.1, 2005.

Hohner, G.; Rich, J.; Ng, E.; Reid, G.; Davenport, A.; Kalagnanam, J.; Lee, H. S. & An, C. (2003). Combinatorial and quantity discount procurement auctions with mutual benefits at mars, incorporated. *Interfaces*, 33:23-35, 2003.

Hoos, H. H. & Boutilier, C. (2000). Solving combinatorial auctions using stochastic local search. *In Proc. of the AAAI2000*, pages 22-29, 2000.

Kastner, R.; Hsieh, C.; Potkonjak, M. & Sarrafzadeh, M. (2002). On the sensitivity of incremental algorithms for combinatorial auctions. *In Proc. International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS2002)*, pages 81-88, 2002.

Koenig, S.; Tovey, C.; Zheng, X. & Sungur, I. (2007). Sequential bundle-bid single-sale auction algorithms for decentralized control. *In Proc. of International Joint Conference on Artificial Intelligence(IJCAI2007)*, pages 1359-1365, 2007.

Lavi, R. & Swamy, C. (2005). Truthful and near-optimal mechanism design via linear programming. I*n 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 595-604, 2005.

Lehmann, D.; Mu¨ller, R. & Sandholm T. (2006). The winner determination problem. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*, chapter 20, pages 507-538. The MIT Press, 2006.

Lehmann, D.; O'Callaghan, L. I. & Shoham, Y. (2002). Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM,* 49:577-602, 2002.

Leyton-Brown, K.; Pearson, M. & Shoham, Y. (2000). Towards a universal test suite for combinatorial auction algorithms. *In Proc. of EC 2000*, pages 66-76, 2000.

McMillan, J. (1994). Selling spectrum rights. *The Journal of Economic Perspectives*, 1994.

Niyato, D. & Hossain, E. (2008). A noncooperative gametheoretic framework for radio resource management in 4g heterogeneous wireless access networks. *IEEE Transactions on Mobile Computing*, pages 332-345, March 2008.

Parkes, D. C.; Cavallo, R.; Elprin, N. ; Juda, A.; Lahaie, S.; Lubin, B.; Michael, L.; Shneidman, J. & Sultan, H. (2005). Ice: An iterative combinatorial exchange. *In The Proc. 6th ACM Conf. on Electronic Commerce (EC'05)*, 2005.

Rassenti, S. J.; Smith, V. L. & Bulfin, R. L. (1982). A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402-417, 1982.

Sabrina, F.; Kanhere, S. S. & Jha, S. K. (2007). Design, analysis, and implementation of a novel low complexity scheduler for joint resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, pages 749-762, June 2007.

Salem, N. B.; Buttyan, L.; Hubaux, J.-P. & Jakobsson, M. (2006). Node cooperation in hybrid ad hoc networks. *IEEE Transactions on Mobile Computing*, pages 365-376, April 2006.

Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1-54, 2002.

Sandholm, T.; Suri, S.; Gilpin, A.; & Levine, D. (2005). Cabob: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51(3):374-390, March 2005.

Thomadakis, M. E. & Liu, J.-C. (1999). On the efficient scheduling of non-periodic tasks in hard real-time systems. *In Proc. of IEEE Real-Time Systems Symp.*, pages 148-151, 1999.

Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, XVI:8-37, 1961.

Xiao, L.; Chen, S.; & Zhang, X. (2004). Adaptive memory allocations in clusters to handle unexpectedly large data-intensive jobs. *IEEE Transactions on Parallel and Distributed Systems*, pages 577-592, July 2004.

Xie, T. & Qin, X. (2007). Security-aware resource allocation for real-time parallel jobs on homogeneous and heterogeneous clusters. *IEEE Transactions on Parallel and Distributed Systems*, Sep. 2007.

Yang, J. & Manivannan, D. (2005). An efficient fault-tolerant distributed channel allocation algorithm for cellular networks. *IEEE Transactions on Mobile Computing*, pages 578-587, Nov. 2005.

Zurel, E. & Nisan, N. (2001). An efficient approximate allocation algorithm for combinatorial auctions. *In Proc. of the Third ACM Conference on Electronic Commerce (EC2001)*, pages 125-136, 2001.

23

Parallel Search Strategies for TSPs using a Greedy Genetic Algorithm

Yingzi Wei[1] and Kanfeng Gu[2]
[1]*School of Information Science and Engineering, Shenyang Ligong University,*
[2]*Shenyang Institute of Automation, Chinese Academy of Science,*
China

1. Introduction

The Genetic Algorithm (GA) is an optimizing algorithm modelled after the evolution of natural organisms. GA was not originally intended for highly constrained optimization problems but were soon adapted to order-based problems like the TSP (Goldberg, D.E. etc. 1985, 1989). It has also been applied to a variety of combinatorial optimization problems. GA is an iterative procedure which maintains a population of candidate solutions. These solutions (instances or chromosomes) are encoded into strings of symbols. The initial population of instances, represented by their chromosomes, can be chosen heuristically or at random. During each iteration step, called a generation, a number of individuals selected from population solutions implement genetic operations. Some of the GA's merits are that it can be easily developed. GA does not require detailed knowledge about the problem, can search globally, and also adapt to the changing conditions in the problem. The traveling salesman problem (TSP) is defined as a very difficult task that seeks a shortest tour of N cities in such a way, that to visit all cities only once and return to the starting city. The TSP was chosen for many reasons: (i) it can be used to model many practical problems, (ii) it is a standard test-bed for new algorithmic ideas and a good performance on the TSP is often taken as a proof of their usefulness or effectiveness, and (iii) it is easily understandable, so that the algorithm behavior is not obscured by too many technicalities.

Despite of these merits, GA is often slower than conventional methods, such as heuristic searches. This is because GA does not utilize explicitly the knowledge of how to search for the solutions. Therefore, hybrid methods that combine GA with other techniques have been attempted (G. Andal Jayalakshmi etc, 2001). The TSP solver we suggested is one of the hybrid methods. It combines GA and greedy principles to construct the TSP solver. With the TSP, we can study the effect of using information about distances of the cities in genetic operators. We improved the genetic operator to guide the generation of new offspring genotypes. Owing to heuristics of greed, it is much faster than other TSP solvers based on GA alone.

This paper begins with a brief description of TSP and GA in general, followed by a review of key to design the GA for permutation problems and analysis of the probable difficulties therein. Then, the greedy selection principle is introduced. In the next a few sections, we present the greedy genetic algorithm (GGA), how we modify a genetic algorithm to solve TSP, our methodology, results, and conclusions.

2. Population initialization

2.1 Encoding scheme

We use a path representation where the cities are listed in the order in which they are visited. In this technique, the N cities are represented by a permutation of the integers from 1 to N. For example, assuming there are 5 cities 1, 2, 3, 4 and 5, if a salesman goes from city 4, through city 1, city 2, city 5, city 3 and returns back to city 4, the chromosome will be {4 1 2 5 3}. For an *N* cities TSP, we initialize the population by randomly placing *1* to *N* into *N* length chromosomes and guaranteeing that each city appears exactly once. Thus chromosomes stand for legal tours.

When using the GA to solve TSPs, the absolute position of a city in a string is less important than the relative position of a city with respect to a tour. So the important information in a chromosome or city sequence is the relative positions of the cities, not the absolute position. Changing the relative positions of the cities may increase or decrease the amount of building blocks and thus result in greater or lesser fitness. For example, for a 5 cities tour, {4 1 2 5 3} and {3 4 1 2 5} mean the same tour. However, pairs of cities are now important. Shortly, highly fit subsets of strings (building blocks) play an important role in the action of genetic algorithms because they combine to form better strings (Goldberg, D.E. etc. 1985, 1989).

2.2 Initial population generation from gene bank

The initial solution plays a critical role in determining the quality of final solution in any local search. However, since the initial population has been produced randomly in most GA researches, it not only requires longer search time to obtain an optimal solution but also decreases the search possibility for an optimal solution. Evolution burden on the GA is especially obvious for TSP when GA starting from an original population with poor quality. For overcoming the difficulties forementioned, we use a gene bank to generate the initial population with good and diverse individuals in this paper.

The N cities are permuted and assembled to build a gene bank. For a TSP of *N* cities, *C* cities that are closer to the city *i* are encoded to construct a gene bank, where *C* is a number less than *N-1*. For simplification, *C* equals 3 in GGA. Gene bank is a matrix $A_{N\times C}$ whose size is $N \times C$. The element of A[*i*][*j*] is the jth closest city to city *i*. For example, A[*i*][*1*] and A[*i*][*2*] are the first and second cities closest to city *i*, respectively. The *C* closest cities constitute the whole ith row of gene bank for the city *i*.

When initializing the population, the first city code *i* is generated randomly. From the *i*th row of gene bank, city code *j* is then generated where *j* is the closest one in the unselected elements of the ith row. Then, city code *h* is selected from the *j*th row of gene bank. If all the city codes of the *j*th row have been selected, GGA produce randomly a city code not traveled before as the next traveling city. Following this method, city codes not traveled are generated to form a complete chromosome. The algorithm repeats the forgoing procedures multiple times. Many such chromosomes form the initial population of GGA.

Our algorithm always makes the choice that looks best when selecting a gene to assemble a chromosome based on the gene bank. This strategy for generating initial population is of a greedy method. The substring assembled based on gene bank is of above-average fitness and short defining length. These schemata with above-average fitness, low-order and short defining length tend to produce more offspring than others. For brevity, such schemata are called building blocks. As we known, building block hypothesis is that a genetic algorithm creates stepwise better solutions by recombining, crossing and mutating short, high-fitness

schemata(Goldberg, D.E. etc. 1985, 1989). So using these substrings is of great benefit to GGA getting an effective solver.

3. Operators of greedy genetic algorithm

A simple class of GAs always guides the algorithm to the solution by preferring individuals with high fitness over low-fitted ones. It can be deterministic, but in most implementation that it has random components. Greedy algorithms are introduced to our genetic operations. After genetic operation, such as crossover and mutation, only the better offspring will replace the parents. This policy is mainly to maintain its respective evolution direction of an individual and deduce the error of random operations.

3.1 Double-directional greedy crossover

Different crossover acts like the different environmental condition impacting on an individual. A different crossover operation changes the domain and procedure of search in order to enhance the possibility of finding a new solution. We adopt multiple crossover operators in this algorithm.

Crossover is a very powerful tool for introducing new genetic material and maintaining genetic diversity, but with the outstanding property that good parents also produce well-performing children or even better ones. Traditionally, combination has been viewed as the primary mechanism and advantage of crossover. However, there is no guarantee that crossover combines the correct schemata.

For crossover operation after several tests and researching, we use the double-directional greedy crossover which is similar to the greedy crossover invented. Greedy crossover selects the first city of one parent, compares the cities leaving that city in both parents, and chooses the closer one to extend the tour (Grefenstette 1985). If one city has already appeared in the tour, we choose the other city. If both cities have already appeared, we randomly select a non-selected city. Greedy crossover guides the searching direction by using local information. The TSP, we chose, is symmetric and its tour is a Hamiltonian cycle. So we propose an effective strategy to improve the greedy crossover operation aforementioned. The gene crossing of a double-directional greedy crossover is applied twice to a chromosome (e.g. to select from the first gene to the last and from the last gene to the first, respectively). This double-directional greedy crossover provides equivalent chances for gene segments located in different positions to reach a local optimum. The method is developed to form a suboptimal cycle based on more effective local searches.

3.2 Greedy mutation

In a GA, the mutation is the random deformation of one or more genes that occurs infrequently during the evolutionary process. The purpose of the mutation is to provide a mechanism to increase coverage of the search space and help prevent premature convergence into a local optimum. Given a permutation based individual of TSP, the mutation operator modifies the related traveling sequence. There are a lot of manners for doing sequence swapping operation. Easiest way is in using random swap. Unfortunately, such strategy unable to achieve an optimum quickly but can prevent convergence into a local optimum.

We use a new mutation operator, greedy-swap of two cities positions. The basic idea of greedy-swap is to randomly select two adjacent cities from one chromosome and swap them

if the new (swapped) tour length is shorter than the elder. For the use of the gene bank when initializing the population, the neighboring coding is often constituted of building block. This strategy is mainly to decrease the possibility of breaking the building block. GGA keep the new tour only when getting a shorter-length tour after not more than 3 trials of swap. So the greedy mutation operation is a procedure of local adjustment and improvement for the chromosome.

3.3 Immigration

The "goodness" of the genetic population depends both on the average fitness (that is corresponding to the objective function value) of individuals and the diversity in the population. Losing on either count tends to produce a poor GA. In the beginning, the potentially good individuals sometimes fill the population so fast that can lead to premature convergence into local maxima. Mutation means to increase diversity in the population by introducing random variations in the members of the population. However, the mutation in the end phase can be too slow to improve population since the individuals have similar fitness values. These problems can be overcome by using the immigration in place of mutation.

Immigration refers to the process of replacing poor members of the current population by bringing in new individuals. For our implementation of the immigration, the population is doped with immigrant individuals for a few of generations. After the midterm phase of evolution, we use the same method to generate immigrants as the method we adopt to generate the initial population. We found that these immigrants not only introduce new genetic material into the population but also bring an open competition plaza for GGA and hence force the algorithm to search newer regions of solution space. Immigrants can also remedy the shortage of small population because the population size is limited for too heavy computation. Figure 1 illustrates the transitional process between consecutive generations of GGA.

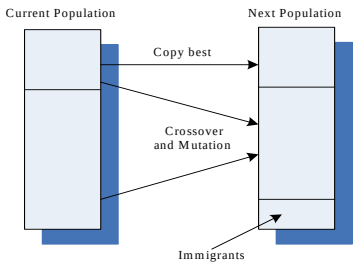


Fig. 1. Transitional process between consecutive generations

4. Evolutionary dynamics of GGA

Genetic algorithms mimic nature evolution using the principles of survival of the fittest. Reproduction operation, or called selection, is the impulsion of GA evolution. A simple GA selects the better individuals from the population into the next generation based on the roulette wheel selection. This always affects the diversities of population for that super-individual(s) will take over most of the population in a few generations.

By comparing the qualities of parents to their offspring's, the individuals of GGA realize the population evolution. Only better offspring will replace its parent place. We abandon the traditional selection operator in our GGA so that the population's diversity is kept very well all along. Each individual runs in its own evolution direction, respectively. Different individuals search different domains. The greedy genetic algorithm takes on the parallelization nature due to its parallel searches.
We haven't employed special fitness function for TSP problems. The length of tour is calculated and directly used for evaluating the fitness of each individual. We leave out the transformation procedure between the objective function and fitness function so as to deduce the computation amount.
The genetic operators of GGA make the most of the heuristic information to achieve local optima. The evolution of whole population fulfils the distributed and parallelized search. So the GGA search is a perfect combination of local and global search for optimal solution keeping from the premature convergence.
By comparing the qualities of parents to their offspring's, the individuals of GGA realize the population evolution. Only better offspring will replace its parent place. We abandon the traditional selection operator in our GGA so that the population's diversity is kept very well all along. Each individual runs in its own evolution direction, respectively. Different individuals search different domains. The greedy genetic algorithm takes on the parallelization nature due to its parallel searches.
We haven't employed special fitness function for TSP problems. The length of tour is calculated and directly used for evaluating the fitness of each individual. We leave out the transformation procedure between the objective function and fitness function so as to deduce the computation amount.
The genetic operators of GGA make the most of the heuristic information to achieve local optima. The evolution of whole population fulfills the distributed and parallelized search. So the GGA search is a perfect combination of local and global search for optimal solution keeping from the premature convergence.

5. Experimental results

We used standard TSP benchmarks (G. Reinelt, 1996) whose optimal solutions (or the current best solutions) are compiled, too. For all the problems, we use the same double-directional greedy crossover and greedy mutation possibilities of 0.8 and 0.02, respectively, but use different population sizes, immigrant possibilities and various number of generations for different problems, as Table 1 shows. Because the template based crossover operation is of the random operation, low possibility of template based crossover is adopted.
We run the GGA 10 times with 10 different random seeds contrast with GA so as to compare the average performance between GGA and GA. For comparison, we also experiment on the different effects between the greedy crossover operator (G. Andal Jayalakshmi, 2001) known and our double-directional greedy crossover operator. We list real number solutions, not the integral ones. From figure 2 to figure 7, we illustrate the best tour routes provided and the best solution that we calculate out with GGA for problem eil51, eil76, eil101, respectively. For problem eil51, we get a new better solution, shown in figure 3, than the provided one (G. Reinelt, 1996). We try to use a higher immigration possibility and less population size for problem eil101 in order to decrease the computation amount, where we get a solution shown in figure 6.
As illustrated in figure 8, for problem eil76, the average tour length of initial population generated from gene bank is 1012 in GGA. However, the average tour length of initial

population generated randomly is 2561 in GA. From figure 8, the curve of average tour length declines straightly with the increase of evolution generation, especially in the start phase. But we notice the occurrence that the curve of GGA fluctuates slightly after the midterm phase of evolution. That is because, after evolving half of the whole generation number, the population is mixed with immigrants that lead the average population fitness to decrease. With sacrificing the high fitness a little, the population retrieves its diversities to some extent. However, the curve of GGA is still in the decline tendency generally. The immigration operation manifests its effect of inhibiting from premature convergence.

Problem instance	Control parameter of GGA			Solutions of GGA and solutions provided			
	Population size	Immigrant possibilities	Number of generations	Average tour length	Best tour length	Best tour length (G. Reinelt, 1996)	Quality of tour
eil51	150	0.15	2000	433.05	428.98	429.98	0.9977
eil76	200	0.15	2000	562.93	553.70	545.39	1.0152
eil101	105	0.2	5000	689.67	665.50	642.30	1.0353

Table 1. Empirical results

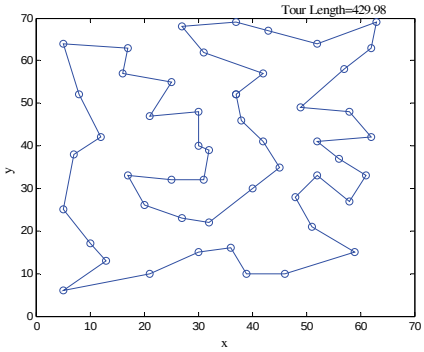


Fig. 2. Best solution of problem eil51 (G. Reinelt, 1996). Tour Length=429.98

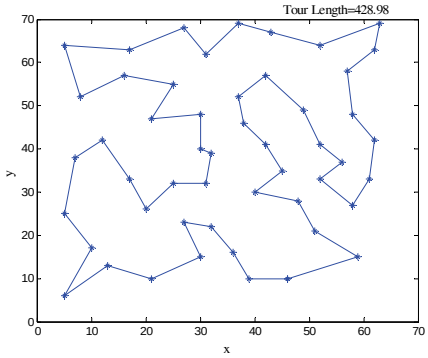


Fig. 3. Our best solution of problem eil51. Tour Length=428.98

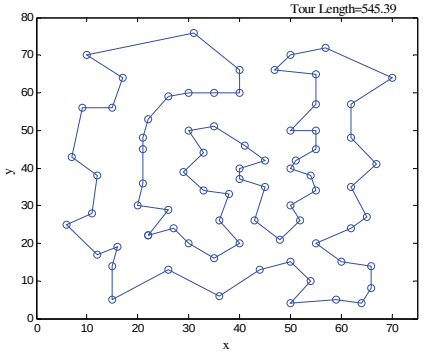


Fig. 4. Best solution of problem eil76 (G. Reinelt, 1996). Tour Length=545.39

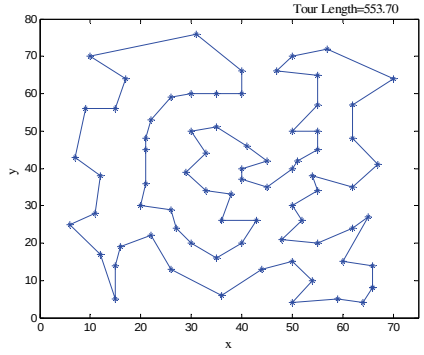


Fig. 5. Our best solution of problem eil76. Tour Length=553.70

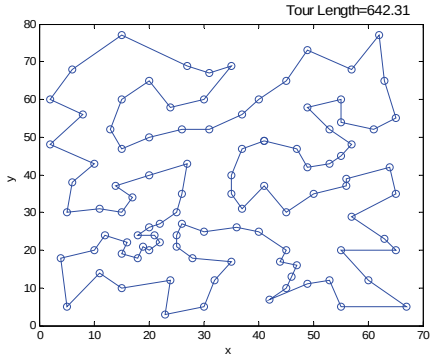


Fig. 6. Best solution of problem eil101 (G. Reinelt, 1996). Tour Length=642.31

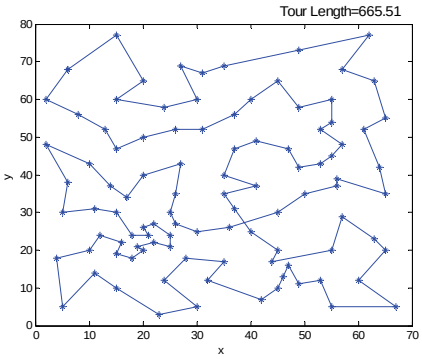


Fig. 7. Our best solution of problem eil101. Tour Length=665.51

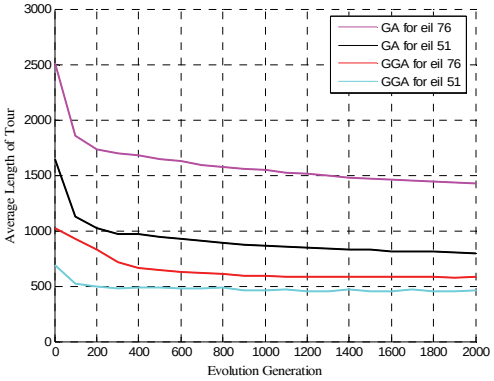


Fig. 8. Performance comparison of different algorithms

6. Conclusion

GA is unable to guarantee to achieve the optimal solution of problems. Compared to the GA, the greedy genetic algorithm with improved genetic operations has been presented for the global optimization of TSPs. The GGA is a parallel-searching algorithm based on TSP-oriented methodologies. Powerful heuristics developed in the corresponding field of TSPs can significantly increase the performance of the genetic algorithm. It is vital for GGA application to engineering practice that GGA works very efficiently in the start phase. A suit of benchmark test has been used to illustrate the merits of the modified genetic operations in GGA. Both the solution quality and stability are improved. GGA demonstrates its promising performance.

7. References

Andrzej Jaszkiewicz, (2002) Genetic local search for multi-objective combinatorial optimization, *European Journal of Operational Research*, Vol. 137, No. 1, pp. 50-71.

Bryan A. Norman & James C. Bean, (1999) A genetic algorithm methodology for complex scheduling problems, *Naval Research Logistics*, Vol. 46, No. 2, pp. 199-211.

Chatterjee S., Carrera C. & Lynch L., (1996) "Genetic algorithms and traveling salesman problems," *European Journal of Operational Research* vol. 93, No. 3, pp. 490-510.

Forbes J. Burkowski, (2004) "Proximity and priority: applying a gene expression algorithm to the Traveling Salesperson Problem," *Parallel Computing*, Vol. 30, No. 5-6, pp. 803-816.

G. Andal Jayalakshmi & S. Sathiamoorthy. (2001) "A Hybrid Genetic Algorithm: A New Approach to Solve Traveling Salesman Problem," *International Journal of Computational Engineering Science* Vol. 2, No. 2, pp. 339-355.

G. Reinelt, (1996) TSPLIB, University of Heidelberg, http://www. iwr. uni-heidelberg.de/iwr/comopt/soft/ TSPLIB95/TSPLIB.html.

Goldberg, D.E. & Lingle, R.J. (1985) Alleles, loci, and the traveling salesman problem, *Proceedings of the International Conference on Genetic Algorithms*, London, pp. 154-159.

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

J. Grefenstette, R. Gopal, R. Rosmaita, & D. Gucht, (1985) Genetic algorithms for the traveling salesman problem, *Proceedings of the Second International Conference on Genetic Algorithms*. Lawrence Eribaum Associates, Mahwah, NJ.

Paul K. Bergey & Cliff Ragsdale, (2005) "Modified differential evolution: a greedy random strategy for genetic recombination," *Omega*, Vol. 33, No. 3, pp. 255-265.

Whitley D., Starkweather, T. & Fuquay, D., (1989) "Scheduling problems and traveling salesmen: the genetic edge recombination operator," *Proceedings of the Third International Conference on Genetic Algorithms*, Los Altos, CA, pp. 133-140.

24

Provably-Efficient Online Adaptive Scheduling of Parallel Jobs Based on Simple Greedy Rules

Yuxiong He[1,2,3] and Wen-Jing Hsu[1,2]
[1]*Singapore-MIT Alliance*
[2]*Nanyang Technological University*
[3]*Sun Microsystems*
Singapore

1. Introduction

Scheduling competing jobs on multiprocessors has always been an important issue for parallel and distributed systems. The challenge is to ensure overall system efficiency while offering a level of fairness to user jobs. Although various degrees of successes have been achieved over the past decades, few existing schemes address both efficiency and fairness over a wide range of work loads. Moreover, in order to obtain analytical results, many known results [22, 24, 7, 8, 17, 20, 23, 25, 33] require prior information about jobs such as jobs' release time, amount of work, parallelism profile, etc, which may be difficult to obtain in real applications. This chapter describes a scheduling algorithm - GRAD, which offers provable efficiency in terms of makespan and mean response time by allotting each job a fair share of processor resources. Our algorithm is *non-clairvoyant* [10, 6, 18, 12], i.e. it assumes nothing about the release time, the execution time, and the parallelism profile of jobs.

A parallel job can be classified as adaptive or non-adaptive. An *adaptively parallel job* [34] may change its parallelism, and it allows the number of the allotted processors to vary during its execution. A job is *nonadaptive* if it runs on a fixed number of processors over its lifetime. With adaptivity, new jobs can enter the system by simply recruiting processors from the already executing jobs. Moreover, in order to improve the system utilization, schedulers can shift processors from jobs that do not require many processors to the jobs in need. However, since the parallelism of adaptively parallel jobs can change during the execution and the future parallelism is usually unknown, how a scheduler decides the processor allotments for jobs is a challenging problem. We describe GRAD that effectively addresses such an adaptive scheduling problem.

Scheduling parallel jobs on multiprocessors can be implemented in two levels [14]: a kernel-level *job scheduler* which allots processors to jobs, and a user-level *thread scheduler* which maps the threads of a given job to the allotted processors. The processor reallocation occurs periodically between *scheduling quanta*. The thread scheduler provides *parallelism feedback* to the job scheduler. The feedback is an estimation of the number of processors that its job can effectively use during the next quantum. The job scheduler follows some processor allocation policy to determine the *allotment* to the job. It may implement a policy that is either *space-sharing*, where jobs occupy disjoint processor resources, or *time-sharing*, where

different jobs may share the same processor resources at different points in time. Once a job is allotted its processors, the allotment does not change within the quantum.

GRAD is a two-level scheduling algorithm that uses simple, greedy-like rules. The thread-level scheduler called A-GREEDY [1] provides feedback based on two simple indicators acquired from the past quantum, namely, whether its request was satisfied and whether the allotted processors are well utilized. Based on the feedbacks from all jobs, the OS allocator RAD [19] partitions processors as equally as possible. Once given the processors, A-GREEDY greedily maps the ready threads of the job onto its allotted processors. If the number of ready threads is less than or equal to the number of allotted processors, all ready threads are scheduled to execute. Otherwise, each allotted processor is assigned with one ready thread. The thread mapping in greedy manner ensures that the allotted processors always make useful work unless there are insufficient number of ready threads to work on.

Based on the "equalized allotment" scheme for processor allocation, and by using the history-based feedback, we show that GRAD is provably efficient. The performance is measured in terms of both makespan and mean response time. GRAD achieves $O(1)$-competitiveness with respect to makespan for job sets with arbitrary release times, and $O(1)$-competitiveness with respect to mean response time for batched job sets where all jobs are released simultaneously. Unlike many previous results, which either assume clairvoyance [29, 21, 31] or use instantaneous parallelism [10, 6], GRAD removes these restrictive assumptions. Moreover, because the quantum length can be adjusted to amortize the cost of context-switching during processor reallocation, GRAD provides effective control over the scheduling overhead and ensures efficient utilization of processors.

Our simulation results also suggest that GRAD performs well in practice. For job sets with arbitrary release time, their makespan scheduled by GRAD is no more than 1.39 times of the optimal on average (geometric mean). For batched job sets, their mean response time scheduled by GRAD is no more than 2.37 times of the optimal on average.

The remainder of this chapter is organized as follows. Section 2 describes the job model, scheduling model, and objective functions. Section 3 describes the GRAD algorithm. Section 4 analyzes the competitiveness of GRAD with respect to makespan. Section 5 shows the competitiveness of GRAD with respect to mean response time for batched jobs, while its detailed analysis is presented in Appendix A. Section 6 presents the empirical results. Section 7 discusses the related work, and Section 8 gives some concluding remarks.

2. Scheduling and analytical model

Our scheduling input consists of a collection of independent jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_{|\mathcal{J}|}\}$ to be scheduled on a collection of P identical processors. Time is broken into a sequence of equal-sized *scheduling quanta* 1, 2, …, each of length L, where each quantum q includes the interval $[L \cdot q, L \cdot q + 1, \ldots, L(q + 1) - 1]$ of time steps. The quantum length L is a system configuration parameter chosen to be long enough to amortize scheduling overheads. In this section, we formalize the job model, define the scheduling model, and present the optimization criteria of makespan and mean response time.

We model the execution of a multithreaded job J_i as a dynamically unfolding directed acyclic graph (*DAG*, for short). Each vertex of the DAG represents a unit-time instruction. The *work* $T_1(J_i)$ of the job J_i corresponds to the total number of vertices in the dag. Each edge represents a dependency between the two vertices. The *span* $T_\infty(J_i)$ corresponds to the

number of nodes on the longest chain of the precedence dependencies. The *release time* $r(J_i)$ of the job J_i is the time at which J_i becomes first available for processing. Each job is handled by a dedicated thread scheduler, which operates in an online manner, oblivious to the future characteristics of the dynamically unfolding DAG.

The job scheduler and the thread schedulers interact as follows. The job scheduler may reallocate processors between scheduling quanta. Between quantum q - 1 and quantum q, the thread scheduler of a given job J_i determines the job's *desire* $d(J_i, q)$, which is the number of processors J_i wants for quantum q. Based on the desire of all running jobs, the job scheduler follows its processor-allocation policy to determine the *allotment* $a(J_i, q)$ of the job with the constraint that $a(J_i, q) \leq d(J_i, q)$. Once a job is allotted its processors, the allotment does not change during the quantum.

A schedule $\mathcal{X} = (\tau, \pi)$ of a job set $\mathcal{J}$ is defined as two mappings $\tau : \cup_{J_i \in \mathcal{J}} V_i \rightarrow \{1, 2, \ldots, 1\}$, and $\pi : \cup_{J_i \in \mathcal{J}} V_i \rightarrow \{1, 2, \ldots, P\}$, which map the vertices of the jobs in the job set $\mathcal{J}$ to the set of time steps, and the set of processors on the machine respectively. A valid mapping must preserve the precedence relationship of each job. For any two vertices $u, v \in V_i$ of the job J_i, if $u \prec v$, then $\tau(u) < \tau(v)$, i.e. the vertex u must be executed before the vertex v. A valid mapping must also ensure that one processor can only be assigned to one job at any given time. For any two vertices u and v, both $\tau(u) = \tau(v)$ and $\pi(u) = \pi(v)$ are true iff $u = v$.

Our scheduler uses makespan and mean response time as the performance measurement.

Definition 1 The *makespan* of a given job set $\mathcal{J}$ is the time taken to complete all the jobs in $\mathcal{J}$, i.e. $T(\mathcal{J}) = \max_{J_i \in \mathcal{J}} T(J_i)$, where $T(J_i)$ denotes the completion time of job J_i.

Definition 2 The *response time* of a job J_i is $T(J_i) - r(J_i)$, which is the duration between its release time $r(J_i)$ and the completion time $T(J_i)$. The *total response time* of a job set $\mathcal{J}$ is given by $R(\mathcal{J}) = \sum_{J_i \in \mathcal{J}} (T(J_i) - r(J_i))$ and the *mean response time* is $\overline{R}(\mathcal{J}) = R(\mathcal{J}) / |\mathcal{J}|$.

The goal of the chapter is to show that our scheduler optimizes the makespan and mean response time, and we use competitive analysis as a tool to evaluate and compare the scheduling algorithm. The competitive analysis of an online scheduling algorithm is to compare the algorithm against an optimal clairvoyant algorithm. Let $T^*(\mathcal{J})$ denote the makespan of an arbitrary jobset $\mathcal{J}$ scheduled by an optimal scheduler, and $T(\mathcal{J})$ denote the makespan produced by an algorithm A for the job set $\mathcal{J}$. A deterministic algorithm A is said to be *c-competitive* if there exists a constant b such that $T(\mathcal{J}) \leq c \cdot T^*(\mathcal{J}) + b$ holds for the schedule of any job set. We will show that our algorithm is c-competitive in terms of the makespan, where c is a small constant. Similarly, for the mean response time, we will show that our algorithm is also constant-competitive for any batched jobs.

3. Algorithms

This section presents the job scheduler - RAD, and overviews the thread scheduler - A-GREEDY [1].

RAD Job Scheduler

The job scheduler RAD unifies the space-sharing job scheduling algorithm DEQ [35, 27] with the time-sharing RR algorithm. When the number of jobs is greater than the number of processors, GRAD schedules the jobs in a batched, round-robin fashion, which allocates one processor to each job with an equal share of time. When the number of jobs is not more than the number of processors, GRAD uses DEQ as the job scheduler. DEQ gives each job an equal share of spatial allotments unless the job requests for less.

When a batch of jobs are scheduled in the round-robin fashion, RAD maintains a queue of jobs. At the beginning of each quantum, if there are more than P jobs, it pops P jobs from the top of the queue, and allots one processor to each of them during the quantum. At the end of the quantum, RAD pushes the P jobs back to the bottom of the queue if they are uncompleted. The new jobs can be put into the queue once they are released.

DEQ attempts to give each job a fair share of processors. If a job requires less than its fair share, however, DEQ distributes the extra processors to the other jobs. More precisely, upon receiving the desires $\{d(J_i, q)\}$ from the thread schedulers of all jobs $J_i \in \mathcal{J}$, DEQ executes the following *processor-allocation algorithm*:

1. Set $n = |\mathcal{J}|$. If $n = 0$, return.
2. If the desire of every job $J_i \in \mathcal{J}$ satisfies $d(J_i, q) \geq P/n$, assign each job $a(J_i, q) = P/n$ processors.
3. Otherwise, let $\mathcal{J}' = \{J_i \in \mathcal{J}: d(J_i, q) < P/n\}$. Assign $a(J_i, q) = d(J_i, q)$ processors to each $J_i \in \mathcal{J}'$. Update $\mathcal{J} = \mathcal{J} - \mathcal{J}'$, and $P = P - \sum_{J_i \in \mathcal{J}'} d(J_i, q)$. Go to Step 1.

Note that, at any quantum where the number of jobs is equal to the number of processors, DEQ and RR give exactly the same processor allotment, and allocate each of P jobs with one processor.

Adaptive Greedy Thread Scheduler

A-GREEDY [1] is an adaptive greedy thread scheduler with parallelism feedback. Between quanta, it estimates its job's desire, and requests processors from the job scheduler. During the quantum, it schedules the ready threads of the job onto the allotted processors greedily [15, 5]. If there are more than $a(J_i, q)$ ready threads, A-GREEDY schedules any $a(J_i, q)$ of them. Otherwise, it schedules all of them.

A- GREEDY's desire-estimation algorithm is parameterized in terms of a *utilization parameter* $\delta > 0$ and a *responsiveness parameter* $\rho > 1$, both of which can be adjusted for different levels of guarantees for waste and completion time.

Before each quantum, A- GREEDY y for a job $J_i \in \mathcal{J}$ provides parallelism feedback to the job scheduler based on the J_i's history of utilization in the previous quantum. A- GREEDY classifies quanta as "satisfied" versus "deprived" and "eficient" versus "inefficient." A quantum q is *satisfied* if $a(J_i, q) = d(J_i, q)$, in which case J_i's allotment is equal to its desire. Otherwise, the quantum is *deprived*.[1] The quantum q is *efficient* if A- GREEDY utilizes no less than a δ fraction of the total allotted processor cycles during the quantum, where δ is the utilization parameter. Otherwise, the quantum is *inefficient*. Under the four-way classification, however, A- GREEDY only uses three: inefficient, efficient-and-satisfied, and efficient-and-deprived.

Using this three-way classification and the job's desire for the previous quantum, A-GREEDY computes the desire for the next quantum as follows:

- If quantum $q - 1$ was inefficient, decrease the desire, setting $d(J_i, q) = d(J_i, q - 1)/2$, where ρ is the responsiveness parameter.
- If quantum $q - 1$ was efficient-and-satisfied, increase the desire, setting $d(J_i, q) = \rho d(J_i, q - 1)$.
- If quantum $q - 1$ was efficient-and-deprived, keep desire unchanged, setting $d(J_i, q) = d(J_i, q - 1)$.

[1] We can extend the classification of "satisfied" versus "deprived" from quanta to time steps. A job J_i is *satisfied* (or *deprived*) at step $t \in [L \cdot q, L \cdot q + 1, \ldots, L(q + 1) - 1]$ if J_i is satisfied (resp. deprived) at the quantum q.

4. Makespan

This section shows that GRAD is c-competitive with respect to makespan, where c denotes a constant. The exact value of c is related to the choice of A-GREEDY's utilization and responsiveness parameter, as will be explained shortly.

We first review the lower bounds of makespan. Given a job set $\mathcal{J}$ and P processors, lower bounds on the makespan of any job scheduler can be obtained based on release time, work, and span. Recall that, for a job $J_i \in \mathcal{J}$, the quantities $r(J_i)$, $T_1(J_i)$, and $T_\infty(J_i)$ represent the release time, work, and span of J_i, respectively. Let $T^*(\mathcal{J})$ denote the makespan produced by an optimal scheduler on a job set $\mathcal{J}$ on P processors. Let $T_1(\mathcal{J}) = \sum_{J_i \in \mathcal{J}} T_1(J_i)$ denote the total work of the job set. The following two inequalities give two lower bounds on the makespan [6]:

$$T^*(\mathcal{J}) \geq \max_{J_i \in \mathcal{J}} \{r(J_i) + T_\infty(J_i)\} , \tag{1}$$

$$T^*(\mathcal{J}) \geq T_1(\mathcal{J})/P . \tag{2}$$

To facilitate the analysis, we state a lemma from [1] that bounds the satisfied steps and the waste of a single job scheduled by A-GREEDY. Recall that, the parameter $\rho > 1$ denotes A-GREEDY's responsiveness parameter, $\delta > 0$ its utilization parameter, and L the quantum length.

Lemma 1 *[1] For a job J_i with work $T_1(J_i)$ and span $T_\infty(J_i)$ on a machine with P processors,* A- GREEDY *produces at most $2T_\infty(J_i)/(1 - \delta)+L\log_\rho P + L$ satisfied steps, and it wastes at most $(1+\rho - \delta)T_1(J_i)/\delta$ processor cycles in the course of the computation.* □

The following theorem analyzes the makespan of a job set $\mathcal{J}$ scheduled by GRAD.

Theorem 2 *Let ρ denote* A-GREEDY's *responsiveness parameter, δ its utilization parameter, and L the quantum length. Then,* GRAD *completes a job set $\mathcal{J}$ on P processors in*

$$T(\mathcal{J}) \leq \frac{\rho+1}{\delta}\frac{T_1(\mathcal{J})}{P} + \frac{2}{1-\delta}\max_{J_i \in \mathcal{J}}\{T_\infty(J_i) + r(J_i)\} + L\log_\rho P + 2L \tag{3}$$

time steps.

Proof. Suppose job J_k is the last job completed among the jobs in $\mathcal{J}$. Let $S(J_k)$ denote the set of satisfied steps for J_k, and $D(J_k)$ denote its set of deprived steps. The job J_k is scheduled to start its execution at the beginning of the quantum q where $Lq < r(J_k) \leq L(q + 1)$, which is the quantum immediately after J_k's release. Therefore, we have $T(\mathcal{J}) \leq r(J_k) + L + |S(J_k)| + |D(J_k)|$. We now bound $|S(J_k)|$ and $|D(J_k)|$ respectively.

From Lemma 1, we know that the number of satisfied steps attributed to J_k is at most $|S(J_k)| \leq 2T_\infty(J_k)/(1 - \delta) + L\log_\rho P + L$.

We now bound the total number of deprived steps $D(J_k)$ of job J_k. For each step $t \in D(J_k)$, GRAD applies either DEQ or RR as job scheduler. RR always allots all processors to jobs. By definition, DEQ must have allotted all processors to jobs whenever J_k is deprived. Thus, the total allotment on such a step t is always equal to the total number of processors P. Moreover, the total allotment of $\mathcal{J}$ over J_k's deprived steps $D(J_k)$ is $a(\mathcal{J}, D(J_k)) = \sum_{t \in D(J_k)} \sum_{J_i \in \mathcal{J}} a(J_i, t) = P|D(J_k)|$. Since any allotted processor is either working productively or wasted, the total allotment for any job J_i is bounded by the sum of its total

work $T_1(J_i)$ and total waste $w(J_i)$. By Lemma 1, the waste for the job J_i is at most $(\rho - \delta + 1)/\delta$ times of its work. Thus, the total number of allotted processor cycles for job J_i is at most $T_1(J_i) + w(J_i) \le (\rho + 1)T_1(J_i)/\delta$. The total number of allotted processor cycles for all jobs is at most $\sum_{J_i \in \mathcal{J}} (\rho + 1)T_1(J_i)/\delta = ((\rho + 1)/\delta)T_1(\mathcal{J})$. Given $a(\mathcal{J}, D(J_k)) \le ((\rho + 1)/\delta)T_1(\mathcal{J})$ and $a(\mathcal{J}, D(J_k)) = P|D(J_k)|$, we have $|D(J_k)| \le \frac{\rho+1}{\delta}\frac{T_1(\mathcal{J})}{P}$.
Therefore, we can get

$$\begin{aligned} \mathrm{T}(\mathcal{J}) &< r(J_k) + L + |D(J_k)| + |S(J_k)| \\ &\le \frac{\rho+1}{\delta}\frac{T_1(\mathcal{J})}{P} + \frac{2}{1-\delta}\max_{J_i \in \mathcal{J}}(T_\infty(J_i) + r(J_i)) \\ &\quad + L\log_\rho P + 2L\,. \end{aligned}$$

□

Since both $T_1(\mathcal{J}) = P$ and $\max_{J_i \in \mathcal{J}}\{T_\infty(J_i) + r(J_i)\}$ are lower bounds of $\mathrm{T}^*(\mathcal{J})$, we obtain the following corollary.

Corollary 3 GRAD *completes a job set* $\mathcal{J}$ *in*

$$\mathrm{T}(\mathcal{J}) \le \left(\frac{\rho+1}{\delta} + \frac{2}{1-\delta}\right)\mathrm{T}^*(\mathcal{J}) + L\log_\rho P + 2L$$

time steps, where $\mathrm{T}^*(\mathcal{J})$ *denotes the makespan of* $\mathcal{J}$ *produced by an optimal clairvoyant scheduler.* □

Since both the quantum length L and the processor number P are independent variables with respect to any job set $\mathcal{J}$, Corollary 3 shows that GRAD is $O(1)$-competitive with respect to makespan.

To better interpret the bound, let's substitute $\rho = 1.2$ and $\delta = 0.6$, we have $\mathrm{T}(\mathcal{J}) \le 8.67\mathrm{T}^*(\mathcal{J}) + L\lg P/\lg 1.2 + 2L$. Since both the quantum length L and the processor number P are independent variables with respect to any job set $\mathcal{J}$, GRAD is 8.67-competitive given $\rho = 1.2$ and $\delta = 0.6$.

When $\delta = 0.5$ and ρ approaches 1, the competitiveness ratio $(\rho + 1)/\delta + 2/(1 - \delta)$ approaches its minimum value 8. Thus, GRAD is $(8 + \epsilon)$-competitive with respect to makespan for any constant $\epsilon > 0$.

5. Mean response time

Mean response time is an important measure for multiuser environments where we desire as many users as possible to get fast response from the system. In this section, we first introduce the lower bounds. Then, we show that GRAD is $O(1)$-competitive for batched jobs with respect to the mean response time.

Lower Bounds and Preliminaries

We first introduce some definitions.

Definition 3 Given a finite list $\mathcal{A} = \langle \alpha_i \rangle$ of $n = |\mathcal{A}|$ integers, define $f: \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ to be a permutation satisfying $\alpha_{f(1)} \le \alpha_{f(2)} \le \ldots \le \alpha_{f(n)}$. The *squashed sum* of A is defined as

$$\text{sq-sum}(\mathcal{A}) = \sum_{i=1}^{n}(n - i + 1)\alpha_{f(i)}\,.$$

The *squashed work area* of a job set $\mathcal{J}$ on a set of P processors is

$$\mathrm{swa}(\mathcal{J}) = \frac{1}{P}\,\text{sq-sum}(\langle T_1(J_i)\rangle)\ ,$$

where $T_1(J_i)$ is the work of job $J_i \in \mathcal{J}$. The *aggregate span* of $\mathcal{J}$ is

$$T_\infty(\mathcal{J}) = \sum_{J_i \in \mathcal{J}} T_\infty(J_i)\ ,$$

where $T_\infty(J_i)$ is the span of job $J_i \in \mathcal{J}$.
The research in [36, 37, 10] establishes two lower bounds for the mean response time:

$$\overline{\mathrm{R}^*}(\mathcal{J}) \ \geq\ T_\infty(\mathcal{J})/|\mathcal{J}|\ , \tag{4}$$

$$\overline{\mathrm{R}^*}(\mathcal{J}) \ \geq\ \mathrm{swa}(\mathcal{J})/|\mathcal{J}|\ , \tag{5}$$

where $\overline{\mathrm{R}^*}(\mathcal{J})$ denotes the mean response time of $\mathcal{J}$ scheduled by an optimal clairvoyant scheduler. Both the aggregate span $T_\infty(\mathcal{J})$ and the squashed work area swa $(\mathcal{J})$ are lower bounds of the total response time R*($\mathcal{J}$) under an optimal clairvoyant scheduler.

Analysis

The proof is divided into two parts. In the first part where $|\mathcal{J}| \leq P$, GRAD always uses DEQ as job scheduler. In this case, we apply the result in [18], and show that GRAD is $O(1)$-competitive. In the second part where $|\mathcal{J}| > P$, GRAD uses both RR and DEQ. Since we consider batched jobs, the number of incomplete jobs decreases monotonically. When the number of incomplete jobs drops to P, GRAD switches its job scheduler from RR to DEQ. Therefore, we prove the second case based on the properties of round robin scheduling and the results of the first case. The following theorem shows the total response time bound for the batched job sets scheduled by GRAD. Please refer to Appendix A for the complete proof.

Theorem 4 *Let* ρ be A-GREEDY's *responsiveness parameter,* δ *its utilization parameter, and* L *the quantum length. The total response time* $\mathrm{R}(\mathcal{J})$ *of a job set* $\mathcal{J}$ *produced by* GRAD *is at most*

$$\mathrm{R}(\mathcal{J}) = \left(2 - \tfrac{2}{|\mathcal{J}|+1}\right)\left(\tfrac{\rho+1}{\delta}\,\mathrm{swa}(\mathcal{J}) + \tfrac{2}{1-\delta}T_\infty(\mathcal{J})\right) + O(|\mathcal{J}|\,L\log_\rho P)\ , \tag{6}$$

where swa $(\mathcal{J})$ *denotes the squashed work area of* $\mathcal{J}$*, and* $T_\infty(\mathcal{J})$ *denotes the aggregate span of* $\mathcal{J}$*.* □

Since both swa $(\mathcal{J})$ / $|\mathcal{J}|$ and $T_\infty(\mathcal{J})$/ $|\mathcal{J}|$ are lower bounds on $\mathrm{R}(\mathcal{J})$, we obtain the following corollary. It shows that GRAD is $O(1)$-competitive with respect to mean response time for batched jobs.

Corollary 5 *The mean response time* $\overline{\mathrm{R}}(\mathcal{J})$ *of a batched job set* $\mathcal{J}$ *produced by* GRAD *satisfies*

$$\overline{\mathrm{R}}(\mathcal{J}) = \left(2 - \tfrac{2}{|\mathcal{J}|+1}\right)\left(\tfrac{\rho+1}{\delta} + \tfrac{2}{1-\delta}\right)\overline{\mathrm{R}^*}(\mathcal{J}) + O(L\log_\rho P)\ ,$$

where $\overline{\mathrm{R}^*}(\mathcal{J})$ *denotes the mean response time of* $\mathcal{J}$ *scheduled by an optimal clairvoyant scheduler.* □

6. Experimental results

To evaluate the performance of GRAD, we conducted four sets of experiments, which are summarized below.

- The **makespan experiments** compares the makespan produced by GRAD against the theoretical lower bound for over 10000 runs of job sets.
- The **mean response time experiments** investigate how GRAD performs with respect to mean response time for over 8000 batched job sets.
- The **load experiments** investigate how the system load affects the performance of GRAD.
- The **proactive RAD experiments** compare the performance of RAD against its variation - proactive RAD. The proactive RAD always allots all processors to jobs even if the overall desire is less than the total number of processors.

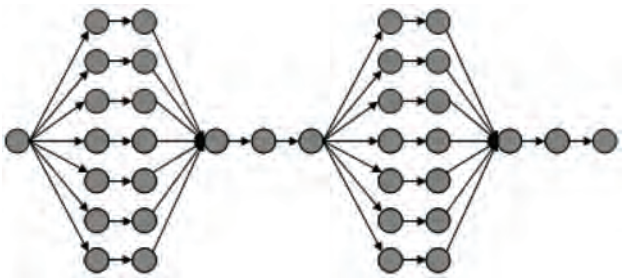


Fig. 1. The DAG of a fork-join job used in the simulation. This job has start-up length $w_0 = 1$, serial phase length $w_1 = 3$, parallel phase length $w_2 = 2$, parallelism $h = 7$, and the number of iterations $iter = 2$.

6.1 Simulation setup

To study GRAD, we build a Java-based discrete-time simulator using DESMO-J [11]. Our simulator models four major entities - processors, jobs, thread schedulers, and job schedulers, and simulates their interactions in a two-level scheduling environment. As described in Section 2, we model the execution of a multithreaded job as a dag. When a job is submitted to the simulated multiprocessor system, an instance of a thread scheduler is created for the job. The job scheduler allots processors to the job, and the thread scheduler executes the job using A-GREEDY. The simulator operates in discrete time steps, and we ignore the overheads incurred in the reallocation of processors.

Our benchmark application is the Fork-Join jobs, whose task graphs are typically as shown in Figure 1. Each job alternates between a *serial phase* of length w_1 and a *parallel phase* (with h-way parallelism) of length w_2, while the initial serial phase has length w_0. The parallelism of job's parallel phase is the *height* h of the job, and the number of iterations is denoted as *iter* . Fork-Join jobs arise naturally in jobs that exhibit "data parallelism", and apply the same computation to a number of different data points. Many computationally intensive applications can be expressed in a data-parallel fashion [30]. The repeated fork-join cycle in the job reflects the often iterative nature of these computations. The average parallelism of the job is approximately $(w_1 + hw_2)=(w_1 + w_2)$. By varying the values of w_0, w_1, w_2, h, and the number of iterations, we can generate jobs with different work, spans, and phase lengths.

GRAD requires some parameters as input. We set the responsiveness parameter to be $\rho = 2.0$, and the utilization parameter $\delta = 0.8$ unless otherwise specified. GRAD is designed for moderate-scale and large-scale multiprocessors, and we set the number of processors to be $P = 128$. The quantum length L represents the time between successive reallocations of

processors by the job scheduler, and is selected to amortize the overheads due to the communication between the job scheduler and the thread scheduler, and the reallocation of processors. In conventional computer systems, a scheduling quantum is typically between 10 and 20 milliseconds. The execution time of a task is decided by the granularity of the job. If a task takes approximately 0.5 to 5 microseconds, then the quantum length L should be set to values between 10^3 and 10^5 time steps. Our theoretical bounds indicate that as long as $T_\infty \gg L\log P$, the length of L should have little effect on our results. In our experiments, we set $L = 1000$.

6.2 Makespan experiments

The competitive ratio of makespan derived in Section 4, though asymptotically strong, has a relatively large constant multiplier. The makespan experiments were designed to evaluate the constants that would occur in practice and compare GRAD to an optimal scheduler. The experiments are conducted on more than 10, 000 runs of job sets using many combinations of jobs and different loads.

Figure 2 shows how GRAD performs compared to an optimal scheduler. The makespan of a job set $\mathcal{J}$ has two lower bounds $\max_{J_i \in \mathcal{J}}(r(J_i) + T_\infty(J_i))$ and $T_1(\mathcal{J}) = P$. The makespan produced by an optimal scheduler is lower-bounded by the larger of these two values. The makespan ratio in Figure 2 is defined as the makespan of a job set scheduled by GRAD divided by the theoretical lower bounds. Its X-axis represents the range of the makespan ratio, while the histogram shows the percentage of the job sets whose makespan ratio falls into the range. Among more than 10, 000 runs, 76.19% of them use less than 1.5 times of the theoretical lower bound, 89.70% use less than 2.0 times, and none uses more than 4.5 times. The average makepsan ratio is 1.39, which suggests that, in practice, GRAD has a small competitive ratio with respect to the makespan.

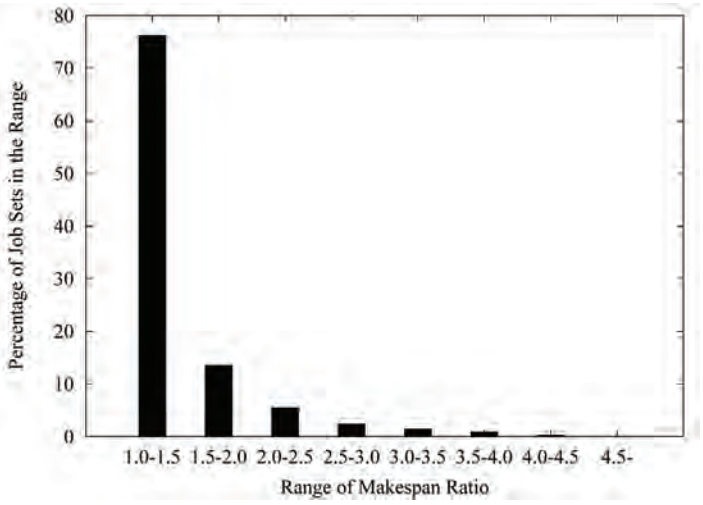


Fig. 2. Comparing the makespan of GRAD with the theoretical lower bound for job sets with arbitrary job release time.

We now interpret the relation between the theoretical bounds and experimental results as follows. When $\rho = 2$ and $\delta = 0.8$, from Theorem 2, GRAD is 13.75-competitive in the worst case. However, we anticipate that GRAD's makespan ratio would be small in practical settings, especially when the jobs have total work much great than the span and with the machine moderately- or highly- loaded. In this case, the term on $T_1(\mathcal{J})/P$ in Inequality (3) of Theorem 2 is much larger than the term $\max_{J_i \in \mathcal{J}} \{T_\infty(i) + r(i)\}$, i.e. the term $T_1(\mathcal{J})/P$ generally dominates the makespan bound. The proof of Theorem 2 calculates the coefficient of $T_1(\mathcal{J})/P$ as the ratio of the total allotment (total work plus total waste) versus the total work. When the job scheduler is RAD, which is not a true adversary, our simulation results indicate that the ratio of the waste versus the total work is only about 1/10 of the total work. Thus, the coefficient of $T_1(\mathcal{J})/P$ in Inequality (3) is about 1.1. It explains why the makespan produced by GRAD is less than 2 times of the lower bound on average as shown in Figure 2.

6.3 Mean response time experiments

This set of experiments is designed to evaluate the mean response time of the batch job sets scheduled by GRAD. Figure 3 shows the distribution of the mean response time normalized w.r.t. the larger of the two lower bounds { the squashed work bound swa $(\mathcal{J})/|\mathcal{J}|$ and the aggregated critical path bound $T_\infty(\mathcal{J})/|\mathcal{J}|$. The histogram in Figure 3 shows that, among more than 8, 000 runs, 94.65% of them use less than 3 times of the theoretical lower bound, and none of them uses more than 5:5 times. The average mean response time ratio is 2.37.

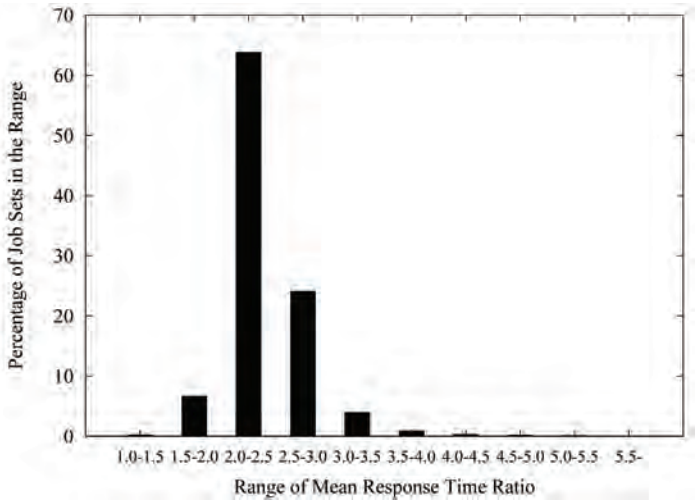


Fig. 3. Comparing the mean response time of GRAD with the theoretical lower bound for batched job sets.

Similar to the discussion in Section 6.2, we can relate the theoretical bounds for mean response time to the experimental results. When $\rho = 2$ and $\rho = 0.8$, from Theorem 4, GRAD is 27.60-competitive. However, we expect that GRAD should perform closer to optimal in practice. In particular, when the job set J exhibits reasonably large total parallelism, we have swa $(\mathcal{J}) \gg T_\infty(\mathcal{J})$, and thus, the term involving swa $(\mathcal{J})$ in Theorem 4 dominates the total response time. More importantly, RAD is not an adversary of A-GREEDY, as mentioned

before, the waste of a job is only about 1/10 of the total work in average for over 100, 000 job runs we tested. Based on this waste, the squashed area bound swa ($\mathcal{J}$) in Inequality (6) of Theorem 4 has a coefficient to be around 2.2. It explains that the mean response time produced by GRAD is less than 3 times of the lower bound as shown in Figure 3.

6.4 Load experiments

This set of experiments is designed to investigate how the load affects the performance of GRAD. The *load* of a job set J on a machine with *P* processors indicates how heavily the jobs compete for processors on the machine, which is calculated as follows

$$load = \frac{T_1(\mathcal{J})}{P \cdot \left(\max_{J_i \in \mathcal{J}} r(J_i) - \min_{J_i \in \mathcal{J}} r(J_i) + T_\infty(\mathcal{J}) / |\mathcal{J}| \right)}.$$

For a batched job set, the load is just the average parallelism of the set divided by the total number of processors.

Figure 4 shows how GRAD performs against the theoretical lower bound with respect to makespan by varying system load. The makespan ratio in this figure is defined as the makespan of a job set scheduled by GRAD divided by the larger of the two lower bounds. Each data point represents the makespan ratio of a job set. The testing results suggest that the makespan ratio becomes smaller when the load gets heavier. Specifically, the makespan generated by GRAD is very close to the lower bound when the load is greater than 4; it never exceeds 1.5 times of the makespan produced when the system load is greater than 3. However, when the load is less than 2, the makespan ratio spreads in the range from 1 to 4.

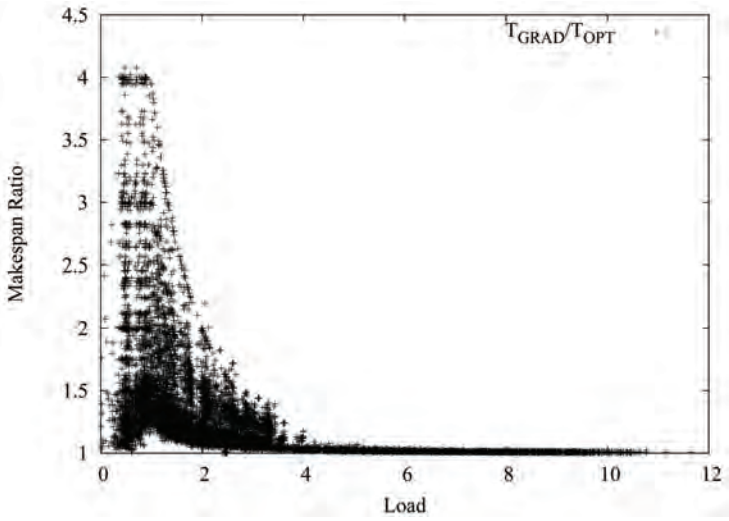


Fig. 4. Comparing GRAD against the theoretical lower bound for makespan with varying load.

Figure 5 shows the performance of GRAD with respect to mean response time for batched jobs by varying system load. It compares the mean response time incurred by GRAD with

the theoretical lower bound. Under heavy load, the mean response time produced by GRAD concentrates on about 2 times of the lower bound, while under light load, the ratio spreads in the range from 1 to 4.

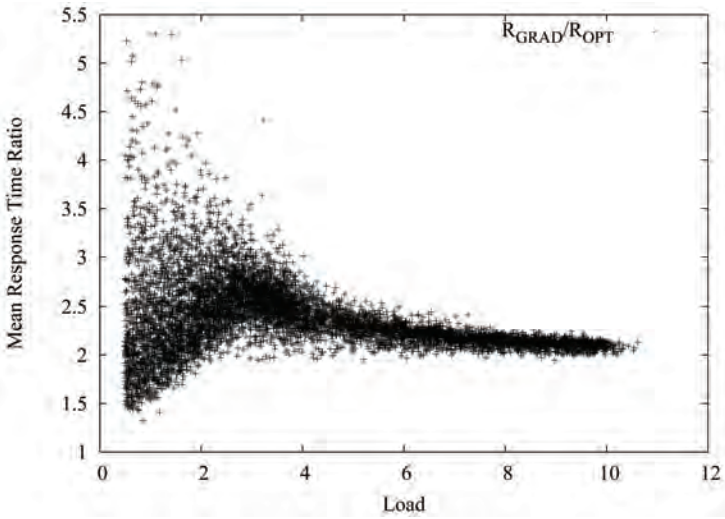


Fig. 5. Comparing GRAD against the theoretical lower bound for mean response time with varying load for batched jobs.

The load experiments bring up a question of how to improve the performance of GRAD under light load. The job scheduler RAD makes conservative decision on the allocation of processors to jobs. When the system is lightly loaded where the total demand is less than the total number of processors, RAD keeps some processors idle without allocating them to any jobs. Since a greedy thread scheduler executes a job faster with more processors allotted, a job scheduler that always allots all processors to jobs should perform better under light load. We will explore such a variation of the job scheduler RAD in the next set of the experiments.

6.5 Proactive RAD experiments

Proactive RAD always allocates all processors to jobs even if the total requests are less than the total number of processors. At a quantum q, when the total requests $d(\mathcal{J}, q) = \sum_{J_i \in \mathcal{J}} d(J_i, q)$ are greater than or equal to the total number P of processors, the proactive RAD works exactly the same as the original one. However, if $d(\mathcal{J}, q) < P$, the proactive RAD evenly allots the remaining $P - d(\mathcal{J}, q)$ processors to all the jobs.

Figure 6 shows the makespan ratio of proactive RAD against its original algorithm by varying system load. Each data point in the figure represents a job set's makespan ratio, defined as the makespan produced by the proactive RAD divided by that of the original. We can see that the makespan ratio is less than 1 for most of the runs, indicating that the proactive RAD out-performs the original one in most of these job sets. Moreover, the difference between them becomes more pronounced under light load, and diminishes with the increase of the system load. The reason is that the proactive RAD generally allocates more processors to jobs, especially when the load is light. The increased allotment allows

faster execution of jobs which shortens the makespan of the job set. Figure 6 gives evidences that the proactive RAD improves the performance of our scheduling algorithm under light load.

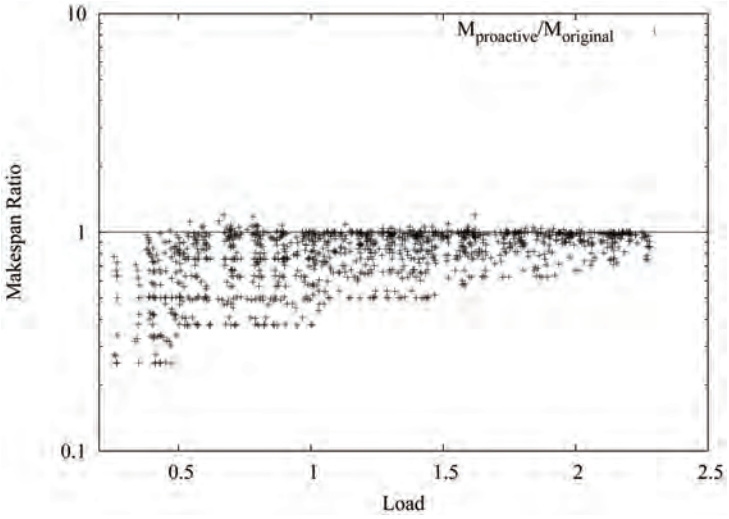


Fig. 6. Comparing the proactive RAD against the original for makespan with varying load. The X-axis represents the load of the system. The Y- axis represents the makespan ratio between the proactive and original RAD.

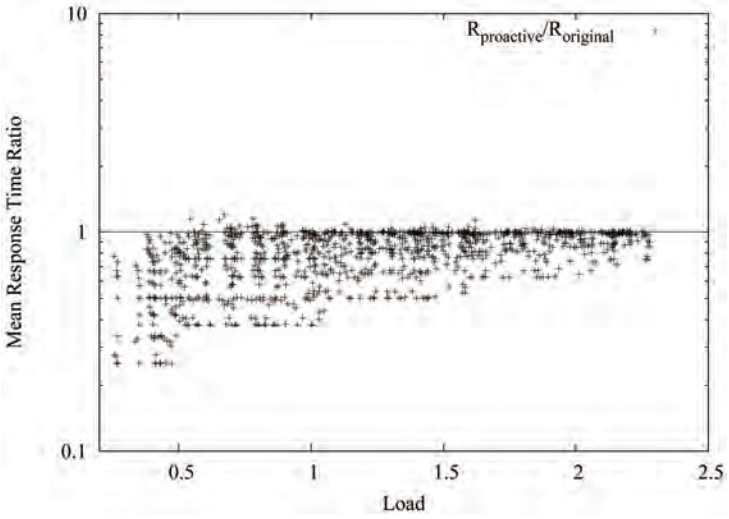


Fig. 7. Comparing the proactive RAD against the original for mean response time with varying load . The X-axis represents the load of the system. The Y-axis represents the mean response time ratio between the proactive and original RAD.

7. Related work

Adaptive parallel job scheduling has been studied both empirically [27, 38, 35, 26] and theoretically [16, 9, 28, 12, 13, 4]. McCann, Vaswani, and Zahorjan [27] introduce the notion of dynamic equipartitioning (DEQ), which gives each job a fair allotment of processors based on the job's request, while allowing processors that cannot be used by a job to be reallocated to the other jobs. Brecht, Deng, and Gu [6] prove that DEQ with instantaneous parallelism as feedback is 2-competitive with respect to the makespan. Later, Deng and Dymond [10] prove that DEQ with instantaneous parallelism is also 4-competitive for batched jobs with respect to the mean response time.

Even though using instantaneous parallelism as feedback is intuitive, it can either cause gross misallocation of processor resources [32] or introduce significant scheduling overhead. For example, the parallelism of a job may change substantially during a scheduling quantum, alternating between parallel and serial phases. Depending on which phase is currently active, the sampling of instantaneous parallelism may lead the task scheduler to request either too many or too few processors. Consequently, the job may either waste processor cycles or take too long to complete. On the other hand, if the quantum length is set to be small enough to capture frequent changes in instantaneous parallelism, the proportion of time spent reallotting processors among the jobs increases, resulting in a high scheduling overhead.

Our previous work in [18] presents a two-level adaptive scheduler AGDEQ, which uses DEQ as the job scheduler, and A-GREEDY as the thread scheduler. Instead of using instantaneous parallelism, AGDEQ uses the job's utilization in the past as feedback. AGDEQ is $O(1)$-competitive for makespan, and in a batched setting, $O(1)$-competitive for mean response time. However, as with other prior work [6, 10] that uses DEQ as the job scheduler, AGDEQ can only be applied to the case where the total number of jobs in the job set is less than or equal to the number of processors.

8. Conclusions

We have presented a non-clairvoyant adaptive scheduling algorithm GRAD that ensures provable efficiency, fairness and minimal overhead.

The history-based feedback mechanism of GRAD can be applied to not only greedy-based thread schedulers, but many other thread schedulers. For example, GRAD using greedy rules to map ready threads to allotted processors is suitable for scheduling jobs in more centralized setting such as data parallel applications. In the centralized setting, the scheduler has the information of all ready threads at any moment such that it can apply greedy rules to make effective assignment of ready threads. However, for applications using many processors and executed with more distributed setting, it can be costly for a scheduler to collect the ready threads information before making each scheduling decision. In this case, other than using a greedy thread scheduler, it is more practical to apply a distributed thread scheduler such as A-STEAL [2, 3] that uses randomized work stealing. A-STEAL performs as well as A-GREEDY asymptotically [3] in terms of both job completion time and waste, however, A-STEAL has slightly larger coefficients because it does not have the complete information on ready threads to make full utilization of the allotted processors. Therefore, a greedy scheduler like A-GREEDY could be a good choice in the centralized setting, while A-STEAL can be applied in the distributed setting where a greedy thread scheduler is no

longer applicable. Analogously, one can develop a two-level scheduler by applying the feedback mechanism in GRAD, and application-specific thread schedulers. Such a two-level scheduler can be developed to provide both system-wide performance guarantees such as minimal makespan and mean response time, and optimization of individual applications.

9. Acknowledgements

The preliminary version of GRAD algorithm was published in our paper [19] coauthored with Charles E. Leiserson. The authors would like to thank Charles for many helpful discussions on formalizing the analysis and advices on revising the write-up.

10. References

[1] K. Agrawal, Y. He, W. J. Hsu, and C. E. Leiserson. Adaptive task scheduling with parallelism feedback. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 100 - 109, New York City, NY, USA, 2006.

[2] K. Agrawal, Y. He, and C. E. Leiserson. An empirical evaluation of work stealing with parallelism feedback. In *Proceedings of the International Conference on Distributed Computing Systems*, pages 19 - 29, Lisboa, Portugal, 2006.

[3] K. Agrawal, Y. He, and C. E. Leiserson. Work stealing with parallelism feedback. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, San Jose, CA, USA, 2007.

[4] N. Bansal, K. Dhamdhere, J. Konemann, and A. Sinha. Non-clairvoyant scheduling for minimizing mean slowdown. *Algorithmica*, 40(4):305-318, 2004.

[5] R. D. Blumofe and C. E. Leiserson. Space-efficient scheduling of multithreaded computations. *SIAM Journal on Computing*, 27(1):202-229, 1998.

[6] T. Brecht, X. Deng, and N. Gu. Competitive dynamic multiprocessor allocation for parallel applications. In *Parallel and Distributed Processing*, pages 448 - 455, San Antonio, TX, 1995.

[7] S. Chakrabarti, C. A. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein. Improved scheduling algorithms for minsum criteria. In *In the Proceedings of Automata, Languages and Programming*, pages 646-657, Paderborn, Germany, 1996.

[8] B. Chen and A. P. A. Vestjens. Scheduling on identical machines: How good is lpt in an on-line setting? *Operations Research Letters*, 21:165-169, 1998.

[9] X. Deng and P. Dymond. On multiprocessor system scheduling. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, pages 82-88, Padua, Italy, 1996.

[10] X. Deng, N. Gu, T. Brecht, and K. Lu. Preemptive scheduling of parallel jobs on multiprocessors. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 159-167, Philadelphia, PA, USA, 1996.

[11] DESMO-J: A framework for discrete-event modelling and simulation. http://asi-www.informatik.uni-hamburg.de/desmoj/.

[12] J. Edmonds. Scheduling in the dark. In *Proceedings of the ACM Symposium on the Theory of Computing*, pages 179-188, Atlanta, Georgia, United States, 1999.

[13] J. Edmonds, D. D. Chinn, T. Brecht, and X. Deng. Non-clairvoyant multiprocessor scheduling of jobs with changing execution characteristics. *Journal of Scheduling*, 6(3):231-250, 2003.

[14] D. G. Feitelson. Job scheduling in multiprogrammed parallel systems (extended version). Technical report, IBM Research Report RC 19790 (87657) 2nd Revision, 1997.

[15] R. L. Graham. Bounds on multiprocessing anomalies. *SIAM Journal on Applied Mathematics*, pages 17(2):416-429, 1969.

[16] N. Gu. Competitive analysis of dynamic processor allocation strategies. Master's thesis, York University, 1995.

[17] L. A. Hall, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: off-line and on-line algorithms. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 142-151, Philadelphia, PA, USA, 1996.

[18] Y. He, W. J. Hsu, and C. E. Leiserson. Provably e±cient two-level adaptive scheduling. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, Saint-Malo, France, 2006.

[19] Y. He, W. J. Hsu, and C. E. Leiserson. Provably efficient online non-clairvoyant scheduling. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, Long Beach, CA, USA, 2007.

[20] K. S. Hong and J. Y. T. Leung. On-line scheduling of real-time tasks. *IEEE Transactions on Computers*, 41(10):1326-1331, 1992.

[21] K. Jansen and H. Zhang. Scheduling malleable tasks with precedence constraints. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, pages 86-95, New York, NY, USA, 2005.

[22] D. Karger, C. Stein, and J. Wein. *Handbook of Algorithms and Theory of Computation*, chapter 35 - Scheduling Algorithms. CRC Press, 1997.

[23] H. Kellerer, T. Tautenhahn, and G. J. Woeginger. Approximability and nonaproximability results for minimizing total flow time on single machine. In *Proceedings of the ACM Symposium on the Theory of Computing*, Philadelphia, Pennsylvania, USA, 1996.

[24] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *Sequencing and Scheduling: Algorithms and Complexity*, pages 445-552. Elsevier Science Publishers, 1997.

[25] S. Leonardi and D. Raz. Approximating total flow time on parallel machines. In *Proceedings of the ACM Symposium on the Theory of Computing*, pages 110-119, El Paso, Texas, USA, 1997.

[26] S. T. Leutenegger and M. K. Vernon. The performance of multiprogrammed multiprocessor scheduling policies. In *SIGMETRICS*, pages 226-236, Boulder, Colorado, United States, 1990.

[27] C. McCann, R. Vaswani, and J. Zahorjan. A dynamic processor allocation policy for multiprogrammed shared-memory multiprocessors. *ACM Transactions on Computer Systems*, 11(2):146-178, 1993.

[28] R. Motwani, S. Phillips, and E. Torng. Non-clairvoyant scheduling. In *Proceedings of the ACM- SIAM Symposium on Discrete Algorithms*, pages 422-431, Austin, Texas, United States, 1993.

[29] G. Mounie, C. Rapine, and D. Trystram. E±cient approximation algorithms for scheduling malleable tasks. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, pages 23-32, New York, NY, USA, 1999.

[30] L. S. Nyland, J. F. Prins, A. Goldberg, and P. H. Mills. A design methodology for data-parallel applications. *IEEE Transactions on Software Engineering*, 26(4):293-314, 2000.

[31] U. Schwiegelshohn, W. Ludwig, J. L.Wolf, J. Turek, and P. S. Yu. Smart smart bounds for weighted response time scheduling. *SIAM Journal of Computing*, 28(1):237-253, 1998.

[32] S. Sen. Dynamic processor allocation for adaptively parallel jobs. Master's thesis, Massachusetts Institute of technology, 2004.
[33] D. B. Shmoys, J. Wein, and D. P. Williamson. Scheduling parallel machines online. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 131-140, San Juan, Puerto Rico, 1991.
[34] B. Song. Scheduling adaptively parallel jobs. Master's thesis, Massachusetts Institute of Technology, 1998.
[35] A. Tucker and A. Gupta. Process control and scheduling issues for multiprogrammed shared-memory multiprocessors. In *Proceedings of the ACM Symposium on Operating Systems Principles*, pages 159-166, New York, NY, USA, 1989.
[36] J. Turek, W. Ludwig, J. L. Wolf, L. Fleischer, P. Tiwari, J. Glasgow, U. Schwiegelshohn, and P. S. Yu. Scheduling parallelizable tasks to minimize average response time. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, pages 200-209, Cape May, New Jersey, United States, 1994.
[37] J. Turek, U. Schwiegelshohn, J. L.Wolf, and P. S. Yu. Scheduling parallel tasks to minimize average response time. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 112-121, Philadelphia, PA, USA, 1994.
[38] K. K. Yue and D. J. Lilja. Implementing a dynamic processor allocation policy for multiprogrammed parallel applications in the Solaris™operating system. *Concurrency and Computation-Practice and Experience*, 13(6):449-464, 2001.

Appendix A. Proof of Theorem 4

The proof is divided into two cases - when $|\mathcal{J}| \le P$ and when $|\mathcal{J}| > P$.

Case 1: when $|\mathcal{J}| \le P$

For the first case where $|\mathcal{J}| \le P$, GRAD always use DEQ as job scheduler. In our previous work [18], we show that AGDEQ (the combination of DEQ and A-GREEDY) is O(1)-competitive with respect to mean response time for batched jobs when $|\mathcal{J}| \le P$. The following lemma from [18] bounds the mean response time of a batched job set with $|\mathcal{J}| \le P$.

Lemma 7 *[18] A job set* $\mathcal{J}$ *is scheduled by* GRAD *on P processors where* $|\mathcal{J}| \le P$. *The total response time* $\mathrm{R}(\mathcal{J})$ *of the schedule is at most*

$$\mathrm{R}(\mathcal{J}) \le c \cdot \left(\frac{\rho+1}{\delta} \mathrm{swa}(\mathcal{J}) + \frac{2}{1-\delta} T_\infty(\mathcal{J}) + |\mathcal{J}| L(\log_\rho P + 1) \right)$$

where $c = 2 - 2/(|\mathcal{J}| + 1)$.

Case 2: when jJ j > *P*

We now derive the mean response time of GRAD for batched jobs for the second case where $|\mathcal{J}| > P$. Since all jobs in the job set J arrive at time step 0, the number of uncompleted jobs decreases monotonically. When the number of uncompleted jobs drops down to *P* or below, GRAD switches its job scheduler from RR to DEQ. We divide the analysis into three parts. In **Part (a)**, we prove two technical lemmas (Lemmas 8 and 9) which show the properties of round robin as the job scheduler. In **Part (b)**, we analyze the completion time of the jobs which are scheduled by RR during their entire execution. In **Part (c)**, we combine results and give response time of GRAD in general.

A batched job set $\mathcal{J}$ can be divided into two subsets - *RR set* and *DEQ set*. The RR set, denoted as $\mathcal{J}_{RR}$, includes all the jobs in $\mathcal{J}$ which are entirely scheduled by RR for their execution. The DEQ set, denoted as $\mathcal{J}_{DEQ}$, includes all the jobs in $\mathcal{J}$ which are scheduled by RR at the beginning, and by DEQ eventually. There exists a unique quantum *q* called the

final RR quantum such that q is the last quantum scheduled by RR, and from quanta q+1 onwards are all scheduled by DEQ. According to RAD, there must be greater than P uncompleted jobs at the beginning of q, and less than or equal to P uncompleted jobs immediately after the execution of q. Let σ denote the total number of uncompleted jobs immediately after the execution of the final RR quantum. We know that $\sigma = |\mathcal{J}_{DEQ}|$, and $\sigma \le P$. Let π denote a permutation that lists the jobs according to the ascending order of their completion time, i.e. $T(J_{\pi(1)}) \le T(\mathcal{J}_{\pi(2)}) \le \ldots \le T(\mathcal{J}_{\pi(|\mathcal{J}|)})$. We have $\mathcal{J}_{RR} = \{\mathcal{J}_{\pi(i)} \mid 1 \le i \le |\mathcal{J}| - \sigma\}$ and $\mathcal{J}_{DEQ} = \{\mathcal{J}_{\pi(i)} \mid i > |\mathcal{J}| - \sigma\}$, i.e. $\mathcal{J}_{DEQ}$ includes the σ jobs that are completed last, and $\mathcal{J}_{RR}$ includes the other $|\mathcal{J}| - \sigma$ jobs.

We define two notations - *t-suffix* and *t-preffix*, and use them to simplify the notations. For any time step t, *t-suffix* denoted as $\overrightarrow{t}$ represents the set of time steps from t to the completion of $\mathcal{J}$ by $\overrightarrow{t} = \{t, t+1, \ldots, T(\mathcal{J})\}$, while *t-preffix* denoted as $\overleftarrow{t}$ represents set of time steps from 1 to t by $\overleftarrow{t} = \{1, 2, \ldots, t\}$. We shall be interested in the suffixes of jobs. Define the *t*-suffix of a job $J_i \in \mathcal{J}$ to be the job $J_i(\overrightarrow{t})$, which is the portion of job J_i that remains after t - 1 number of time steps have been executed. The *t*-suffix of the job set $\mathcal{J}$ is

$$\mathcal{J}(\overrightarrow{t}) = \{J_i(\overrightarrow{t}) : J_i \in \mathcal{J} \text{ and } J_i(\overrightarrow{t}) \neq \emptyset\} .$$

Thus, we have $\mathcal{J} = \mathcal{J}(\overrightarrow{1})$, and the number of uncompleted jobs at time step t is the number $|\mathcal{J}(\overrightarrow{t})|$ of nonempty jobs in $\mathcal{J}(\overrightarrow{t})$. Similarly, we can define the t-prefix of a job J_i as $J_i(\overleftarrow{t})$, and the t-prefix of a job set $\mathcal{J}$ as $\mathcal{J}(\overleftarrow{t})$.

Case 2 - Part (a)

The following two technical lemmas present the properties of round robin as a job scheduler. The first lemma shows that jobs make almost the same progress on the execution of their work when they are scheduled by RR. The second lemma relates the work of jobs to their completion time.

Lemma 8 *A batched job set $\mathcal{J}$ is scheduled by* GRAD *on a machine with P processors where $|\mathcal{J}| > P$. At any time step t scheduled by RR, for any two uncompleted jobs J_i and J_j, we have $|T_1(J_i(\overleftarrow{t})) - T_1(J_j(\overleftarrow{t}))| \le L$, where L is the length of the scheduling quantum.*

Proof. Since RR gives an equal share of processors to all uncompleted jobs, for any two jobs that arrive at the same time, their allotments differ by at most L at any time. When a job's allotment is 1, its allotted processor is always making useful work. Then the work done for any two uncompleted jobs differs by at most L at any time before their completion. □

Lemma 9 *A batched job set $\mathcal{J}$ is scheduled by* GRAD *on a machine with P processors where $|\mathcal{J}| > P$. The following two statements are true*:

1. *If $J_i \in \mathcal{J}_{RR}$, $J_j \in \mathcal{J}_{RR}$, and $T_1(J_i) < T_1(J_j)$, then $T(J_i) \le T(J_j)$.*
2. *If $J_i \in \mathcal{J}_{RR}$, and $J_j \in \mathcal{J}_{DEQ}$, then $T_1(J_i) \le T_1(J_j)$.*

Proof. We now prove the first statement. Let $t = T(J_i)$. At time step t, job J_i completes work $T_1(J_i)$. From Lemma 8, we know that $T_1(J_j(\overleftarrow{t})) \ge T_1(J_i(\overleftarrow{t})) - L = T_1(J_i) - L$. Since job J_j completes after job J_i, job J_j takes at least one more scheduling quantum than J_i to complete its execution. Thus the work done for J_j during the period from t to $T(J_j)$ is at least L. Therefore, we have $T_1(J_j) = T_1(J_i(\overleftarrow{T(J_j)})) \ge T_1(J_i(\overleftarrow{t})) + L \ge T_1(J_i)$.

For any two jobs $J_i \in \mathcal{J}_{RR}$, and $J_j \in \mathcal{J}_{DEQ}$, we have $T(J_i) < T(J_j)$. By using a similar analysis, we can prove the second statement.

Lemma 9 relates the work of jobs to their completion time. Its second statement tells us that only the σ jobs with largest work are scheduled by DEQ eventually, and the other $|\mathcal{J}|$ - σ jobs are scheduled by RR for their overall execution. Moreover, according to its first statement, under the schedule of RR, the jobs with less work are completed more quickly than those with more work. Consider the jobs according to their work such that $T_1(J_1) \le T_1(J_2) \le \ldots \le T_1(\mathcal{J}_{|\mathcal{J}|})$. From Lemma 9, we have $\mathcal{J}_{RR} = \{J_i \mid 1 \le i \le |\mathcal{J}| - \sigma\}$ and $\mathcal{J}_{DEQ} = \{J_i \mid i > |\mathcal{J}| - \sigma\}$.

Case 2 - Part (b)

The following lemma bounds the completion time of the jobs in $\mathcal{J}_{RR}$ where $T_1(J_i)$ denotes the work of a job J_i.

Lemma 10 GRAD *schedules a batched job set* $\mathcal{J}$ *on a machine with P processors where* $|\mathcal{J}| > P$. *Consider the jobs according to their work such that* $T_1(J_1) \le T_1(J_2) \le \ldots \le T_1(\mathcal{J}_{|\mathcal{J}|})$. *For* $1 \le i \le |\mathcal{J}| - \sigma$, *the completion time* $T(J_i)$ *of a job* J_i *is* $T(J_i) \le ((|\mathcal{J}| - i + 1)\, T_1(J_i) + \sum_{1 \le j < i} T_1(J_j)) = P + L$.

Proof. Since we consider the jobs according to their work, from Lemma 9, we have $J_i \in J_{RR}$ where $1 \le i \le |\mathcal{J}| - \sigma$. Such a job J_i completes its overall execution under the schedule of RR as job scheduler.

We first evaluate $T_1(\mathcal{J}(\overleftarrow{t}))$, which is the work done for $\mathcal{J}$ up to a time step t. Suppose that the job J_i terminates at the end of a quantum q where $T(J_i) = q(L + 1) - 1$. Let $t = qL - 1$ be the end of the quantum $q - 1$, which is L steps before the completion of J_i. The work done for J_i in interval $\overleftarrow{t}$ is $T_1(J_i(\overleftarrow{t})) = T_1(J_i) - L$. According to Lemma 8, no job completes more than $T_1(J_i(\overleftarrow{t})) + L$ amount of work in interval $\overleftarrow{t}$. Therefore, for any job J_j with $j > i$, we have

$$\begin{aligned} T_1(J_j(\overleftarrow{t})) &\le T_1(J_i(\overleftarrow{t})) + L \\ &= T_1(J_i) \ . \end{aligned} \tag{7}$$

For each job J_j where $j < i$, by definition, we always have

$$T_1(J_j(\overleftarrow{t})) \le T_1(J_j) \ . \tag{8}$$

Thus, at time step t, from Inequalities (7) and (8), the total work done for the job set $\mathcal{J}$ is

$$\begin{aligned} & T_1(\mathcal{J}(\overleftarrow{t})) \\ &= \sum_{1 \le j < i} T_1(J_j(\overleftarrow{t})) + T_1(J_i(\overleftarrow{t})) + \sum_{i < j \le |\mathcal{J}|} T_1(J_j(\overleftarrow{t})) \\ &\le (|\mathcal{J}| - i + 1) T_1(J_i) + \sum_{1 \le j < i} T_1(J_j) \ . \end{aligned} \tag{9}$$

Since RR always allots all processors to jobs, and all allotted processors are making useful work, RR executes P ready threads at any time step. Thus, the total work done for job set $\mathcal{J}$ increases by P at each time step. From Inequality (9), we have

$$\begin{aligned} t &= T_1(\mathcal{J}(\overleftarrow{t}))/P \\ &\le \left((|\mathcal{J}| - i + 1) T_1(J_i) + \sum_{1 \le j < i} T_1(J_j)\right)/P \ . \end{aligned}$$

Since $T(J_i) = t + L$, we complete the proof. □

Case 2 - Part (c)
The following lemma bounds the total response time of job sets scheduled by GRAD when $|\mathcal{J}| > P$ where swa ($\mathcal{J}$) denotes squashed work area, and $T_\infty(\mathcal{J})$ denotes the aggregate span.
Lemma 11 Suppose that a job set $\mathcal{J}$ is scheduled by GRAD on a machine with P processors where $|\mathcal{J}| > P$. The response time R($\mathcal{J}$) of $\mathcal{J}$ is bounded by

$$\mathrm{R}(\mathcal{J}) = \left(2 - \tfrac{2}{|\mathcal{J}|+1}\right)\left(\tfrac{\rho+1}{\delta}\,\mathrm{swa}\,(\mathcal{J}) + \tfrac{2}{1-\delta}T_\infty(\mathcal{J})\right) + |\mathcal{J}|\,L + O(LP\log_\rho P)\ . \quad (10)$$

Proof. The jobs in $\mathcal{J}$ can be divided into RR set $\mathcal{J}_{RR}$ and DEQ set $\mathcal{J}_{DEQ}$. Let $n = |\mathcal{J}|$ denote the number of jobs in $\mathcal{J}$. Recall that σ denotes the number of jobs in $\mathcal{J}_{DEQ}$, i.e. $\sigma \le P$. Consider the jobs in the ascending order of their completion time such that $T(J_1) \le T(J_2) \le \ldots \le T(J_n)$. From Lemma 9, we have $\mathcal{J}_{RR} = \{J_i \mid 1 \le i \le n - \sigma\}$ and $\mathcal{J}_{DEQ} = \{J_i \mid i > n - \sigma\}$. We will calculate the total response time of the jobs in $\mathcal{J}_{RR}$ and $\mathcal{J}_{DEQ}$ respectively.
Step 1: To calculate R($\mathcal{J}_{RR}$), we apply Lemma 10. For any job $J_i \in \mathcal{J}_{RR}$, its completion time is $T(J_i) \le (1/P)((n - i+1)T_1(J_i)+\sum_{1 \le j < i} T_1(J_j))+L$ according to Lemma 10. Thus, the total response time of the jobs in $\mathcal{J}_{RR}$ is

$$\mathrm{R}(\mathcal{J}_{RR}) \le \tfrac{1}{P}\sum_{1\le i\le n-\sigma}(2n-\sigma-2i+1)T_1(J_i) + Ln. \quad (11)$$

Step 2: We now calculate R($\mathcal{J}_{DEQ}$). The σ jobs in $\mathcal{J}_{DEQ}$ are scheduled by RR until the time step $t = T(J_{n-\sigma})$ at which the job $J_{n-\sigma}$ completes, and scheduled by DEQ afterwards. The total response time of $\mathcal{J}_{DEQ}$ is

$$\mathrm{R}\left(\mathcal{J}_{DEQ}\right) = \mathrm{R}\left(\mathcal{J}_{DEQ}\left(\overrightarrow{t+1}\right)\right) + \sigma \cdot t\ . \quad (12)$$

From Lemma 10, we know that the completion time of the job $J_{n-\sigma}$ is

$$t \le \left((\sigma+1)T_1(J_{n-\sigma}) + \sum_{1\le i<n-\sigma} T_1(J_i)\right)/P + L\ . \quad (13)$$

To get R($\mathcal{J}_{DEQ}$), we only need to calculate R($\mathcal{J}_{DEQ}\left(\overrightarrow{t+1}\right)$).
Since the job set $\mathcal{J}_{DEQ}$ is scheduled by DEQ as the job scheduler from time step t onwards, we can apply the total response time bound in Lemma 7 to calculate R($J_{DEQ}\left(\overrightarrow{t+1}\right)$). During the interval $\overleftarrow{t}$, job $J_{n-\sigma}$ completes $T_1(J_{n-\sigma})$ amount of work. From Lemma 8, we know that each job J_i with $i > n - \sigma$ has completed at least $T_1(J_{n-\sigma}) - L$ amount of work. Thus, such a job J_i has remaining work $T_1(J_i\left(\overrightarrow{t+1}\right) \le T_1(J_i) - T_1(J_{n-\sigma}) + L$. The squashed work of $\mathcal{J}_{DEQ}\left(\overrightarrow{t+1}\right)$ is

$$\begin{aligned}
&\mathrm{swa}\left(\mathcal{J}_{DEQ}\left(\overrightarrow{t+1}\right)\right)\\
&= \frac{1}{P}\,\text{sq-sum}\left(\left\langle T_1\left(J_i\left(\overrightarrow{t+1}\right)\right) \mid n-\sigma+1 \le i \le n\right\rangle\right)\\
&\le \frac{1}{P}\,\text{sq-sum}\left(\left\langle T_1(J_i) - T_1(J_{n-\sigma}) + L \mid n-\sigma+1 \le i \le n\right\rangle\right)\\
&= \tfrac{1}{P}\sum_{n-\sigma+1\le i\le n}(n-i+1)\left(T_1(J_i) - T_1(J_{n-\sigma}) + L\right)\\
&\le \tfrac{1}{P}\sum_{n-\sigma+1\le i\le n}(n-i+1)T_1(J_i) - \tfrac{(1+\sigma)\sigma}{2P}T_1(J_{n-\sigma}) + PL\ .
\end{aligned} \quad (14)$$

Let the constant $c = 2 - 2/(1 + P) < 2$. According to Lemma 7, we have

$$\mathrm{R}(\mathcal{J}_{DEQ}(\overrightarrow{t+1})) \le c \cdot \frac{\rho+1}{\delta} \mathrm{swa}\left(\mathcal{J}_{DEQ}(\overrightarrow{t+1})\right) + E_1 \ , \tag{15}$$

where $E_1 = c \cdot \frac{2}{1-\delta} T_\infty(\mathcal{J}) + cPL(\log_\rho P + 1)$.
We will now calculate the response time of $\mathcal{J}_{DEQ}$. Since we know $c = 2 - 2/(1 + P) > 1$, the responsiveness parameter $\rho > 1$, and the utilization parameter $\delta \le 1$, we have $c(\rho + 1)=\delta > 2$. Given Equation (12), and Inequalities (13), (14) and (15), the response time of $\mathcal{J}_{DEQ}$ is

$$\begin{aligned} & \mathrm{R}(\mathcal{J}_{DEQ}) \\ &= \mathrm{R}(\mathcal{J}_{DEQ}(\overrightarrow{t+1})) + \sigma \cdot t \\ &\le c \cdot \tfrac{\rho+1}{\delta} \mathrm{swa}\left(\mathcal{J}_{DEQ}(\overrightarrow{t+1})\right) + E_1 + \sigma \cdot t \\ &\le c \cdot \tfrac{\rho+1}{\delta P} \textstyle\sum_{n-\sigma+1 \le i \le n} (n-i+1) T_1(J_i) \\ &\quad + \tfrac{\sigma}{P} \textstyle\sum_{1 \le i < n-\sigma} T_1(J_i) + E_2 \ , \end{aligned} \tag{16}$$

where $E_2 = E_1 + (c \cdot \frac{\rho+1}{\delta} + 1)PL$.
Step 3: Given $\mathrm{R}(\mathcal{J}_{RR})$ in Inequality (11), $\mathrm{R}(\mathcal{J}_{DEQ})$ in Inequality (16), and $c(\rho+1)=\delta > 2$, the response time of $\mathcal{J}$ is the sum of them as follows:

$$\begin{aligned} & \mathrm{R}(\mathcal{J}) \\ &= \mathrm{R}(\mathcal{J}_{RR}) + \mathrm{R}(\mathcal{J}_{DEQ}) \\ &< \frac{1}{P} \sum_{1 \le i \le n-\sigma} (2n - \sigma - 2i + 1) T_1(J_i) + Ln + c \cdot \frac{\rho+1}{\delta P} \sum_{n-\sigma+1 \le i \le n} (n-i+1) T_1(J_i) \\ &\quad + \frac{\sigma}{P} \sum_{1 \le i < n-\sigma} T_1(J_i) + E_2 \\ &= \frac{1}{P} \sum_{1 \le i \le n-\sigma} (2n - 2i + 1) T_1(J_i) + E_2 + Ln + c \cdot \frac{\rho+1}{\delta P} \sum_{n-\sigma+1 \le i \le n} (n-i+1) T_1(J_i) \\ &\quad + \frac{\sigma}{P} \sum_{1 \le i < n-\sigma} T_1(J_i) - \frac{\sigma}{P} \sum_{1 \le i \le n-\sigma} T_1(J_i) \\ &\le c \cdot \frac{\rho+1}{\delta P} \sum_{J_i \in \mathcal{J}} (n-i+1) T_1(J_i) + c \cdot \frac{2}{1-\delta} T_\infty(\mathcal{J}) + Ln + E_2 \\ &= \left(2 - \frac{2}{n+1}\right)\left(\frac{\rho+1}{\delta} \mathrm{swa}(\mathcal{J}) + \frac{2}{1-\delta} T_\infty(\mathcal{J})\right) + Ln + O(PL \log_\rho P) \ . \end{aligned}$$

□

Lemmas 7 and 11 bound the total response time of a batched job set $\mathcal{J}$ when $|\mathcal{J}| \leq P$ and $|\mathcal{J}| > P$ respectively. Combining them, we have completed the proof of Theorem 4.

□

25

Quasi-Concave Functions and Greedy Algorithms

Yulia Kempner[1], Vadim E. Levit[2] and Ilya Muchnik[3]
[1] *Holon Institute of Technology,*
[2] *Ariel University Center of Samaria,*
[3] *Rutgers - the State University of New Jersey,*
[1,2]*Israel*
[3]*USA*

1. Introduction

Many combinatorial optimization problems can be formulated as: for a given *set system* over E (i.e., for a pair $(E, \mathcal{F})$ where $\mathcal{F} \subseteq 2^E$ is a family of feasible subsets of finite set E), and for a given function $F : \mathcal{F} \rightarrow \mathbf{R}$, find an element of $\mathcal{F}$ for which the value of the function F is minimum or maximum. In general, this optimization problem is **NP**-hard, but for some specific functions and set systems the problem may be solved in polynomial time. For instance, greedy algorithms may optimize linear objective functions over matroids [11] and Gaussian greedoids [5], [15], [32], while bottleneck objective functions can be maximized over general greedoids [16]. A generalization of greedoids in the context of dynamic programming is discussed in [1] and [2].
Another example is about set functions defined as minimum values of monotone linkage functions. These functions are known as quasi-concave set functions. Such a set function can be maximized by a greedy type algorithm over the family of all subsets of E [19],[24],[29],[30],[34], over antimatroids and convex geometries [17], [20], [25], join-semilattices [28] and meet-semilattices [21]. A relationship was also established between submodular and quasi-concave functions [28] that allowed to build series of branch and bound procedures for finding maximum of submodular functions.
Originally, quasi-concave set functions were considered [23] on the Boolean 2^E

$$\text{for each } X, Y \subset E,\ F(X \cup Y) \geq \min\{F(X), F(Y)\}. \tag{1}$$

In this work we extend this definition to various set systems. One of the natural extensions is a join-semilattice. Here, $\mathcal{S} \subseteq 2^E$ is a *join-semilattice* if it is closed under union, i.e., $A \cup B \in \mathcal{S}$ for each $A, B \in \mathcal{S}$.
Another direction of our research is to adapt the definition of the quasi-concave set functions to set systems that are not necessarily closed under union. Let E be a finite set, and a pair $(E, \mathcal{F})$ be a set system over E. A minimal feasible subset of E that includes a set X is called a *cover of* X. We will denote by $\mathcal{C}(X)$ the family of covers of X. Then the inequality (1) turns into the following.
Definition 1 *A function F defined on a set system* $(E, \mathcal{F})$ *is quasi-concave if for each* $X, Y \in \mathcal{F}$, *and* $Z \in \mathcal{C}(X \cup Y)$

$$F(Z) \geq \min\{F(X), F(Y)\}. \tag{2}$$

If a set system is closed under union, then the family of covers $\mathcal{C}(X \cup Y)$ contains the unique set $X \cup Y$, and the inequality (2) coincides with the original inequality (1).

This chapter is organized as follows. Section 1 contains an extended introduction. Section 2 gives basic information about monotone linkage functions. We show that for a number of combinatorial structures the class of functions defined as the minimum values of monotone linkage functions coincides with the class of quasi-concave set functions. Section 3 deals with the construction of efficient algorithms for maximizing quasi-concave functions which are associated with monotone linkage functions. It is shown that properties of combinatorial structures affect their corresponding optimization algorithms. Section 4 deals with applications to clustering in bioinformatics. In this section we use a particular class of quasi-concave set functions as natural criteria for cluster analysis. We describe how the Fibonacci heap structure can dramatically reduce the computational complexity. Section 5 contains conclusions and directions of future research.

2. Preliminaries

Here we will give definitions of some set properties that are discussed in the following sections. We will use $X \cup x$ for $X \cup \{x\}$, and $X - x$ for $X - \{x\}$.

A non-empty set system $(E, \mathcal{F})$ is called *accessible* if for each non-empty $X \in \mathcal{F}$, there exists an $x \in X$ such that $X - x \in \mathcal{F}$.

For each non-empty set system $(E, \mathcal{F})$ accessibility implies that $\varnothing \in \mathcal{F}$.

Definition 2 *A closure operator,* $\tau: 2^E \rightarrow 2^E$, *is a map satisfying the closure axioms:*

C1: $X \subseteq \tau(X)$

C2: $X \subseteq Y \Rightarrow \tau(X) \subseteq \tau(Y)$

C3: $\tau(\tau(X)) = \tau(X)$.

Definition 3 *The set system* $(E, \mathcal{F})$ *is a closure system if it satisfies the following properties*

(1) $\varnothing \in \mathcal{F}$, $E \in \mathcal{F}$

(2) $X, Y \in \mathcal{F}$ *implies* $X \cap Y \in \mathcal{F}$.

Let a set system $(E, \mathcal{F})$ be a closure system, then the operator

$$\tau(A) = \cap\{X : A \subseteq X \text{ and } X \in \mathcal{F}\} \tag{3}$$

is a closure operator.

A convex geometries was introduced by Edelman and Jamison [9] as a combinatorial abstraction of "convexity".

Definition 4 *[16] The closure system* $(E, \mathcal{F})$ is *a convex geometry if the family* $\mathcal{F}$ *satisfies the following property*

$$X \in \mathcal{F} - E \text{ implies } X \cup x \in \mathcal{F} \text{ for some } x \in E - X. \tag{4}$$

It is easy to see that property (4) is dual to accessibility. Then, we will call it *up-accessibility*. If in each non-empty accessible set system one can reach the empty set $\varnothing$ from any feasible set $X \in \mathcal{F}$ by moving down, so in each non-empty up-accessible set system $(E, \mathcal{F})$ the set E may be reached by moving up.

It is clear that a complement set system $(E, \overline{\mathcal{F}})$ (system of complements), where $\overline{\mathcal{F}} = \{X \subseteq E : E - X \subseteq \mathcal{F}\}$, is up-accessible if and only if the set system $(E, \mathcal{F})$ is accessible.

In fact, accessibility means that for all sets $X \in \mathcal{F}$ there exists a chain $\varnothing = X_0 \subset X_1 \subset \ldots \subset X_k = X$ such that $X_i = X_{i-1} \cup x_i$ and $X_i \in \mathcal{F}$ for $0 \le i \le k$, and up-accessibility implies the existence of the corresponding chain $X = X_0 \subset X_1 \subset \ldots \subset X_k = E$. Consider a set family for which this *chain property* holds for each pair of sets $X \subset Y$.

Definition 5 *A set system* $(E, \mathcal{F})$ *satisfies the chain property if for all* $X, Y \in \mathcal{F}$, *and* $X \subset Y$, *there exists an* $y \in Y - X$ *such that* $Y - y \in \mathcal{F}$. *We call the system a chain system*.

In other words, a set system $(E, \mathcal{F})$ satisfies the chain property if for all $X, Y \in \mathcal{F}$, and $X \subset Y$, there exists a chain $X = X_0 \subset X_1 \subset \ldots \subset X_k = Y$ such that $X_i = X_{i-1} \cup x_i$ and $X_i \in \mathcal{F}$ for $0 \le i \le k$.

It is easy to see that $(E, \overline{\mathcal{F}})$ is a chain system if and only if $(E, \mathcal{F})$ is a chain system as well.

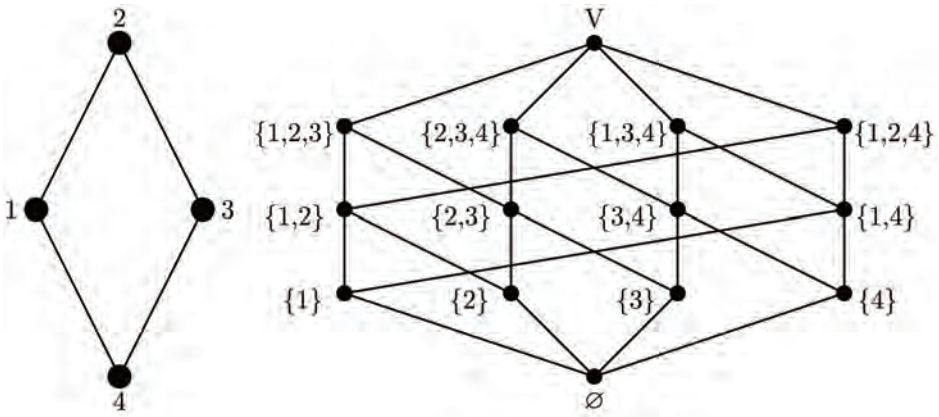
Consider a relation between accessibility and the chain property. If $\varnothing \in \mathcal{F}$, then accessibility follows from the chain property. In general case, there are accessible set systems that do not satisfy the chain property (for example, consider $E = \{1, 2, 3\}$ and $\mathcal{F} = \{\varnothing, \{1\}, \{2\}, \{2, 3\}, \{1, 2, 3\}\}$) and vice versa, it is possible to construct a set system, that satisfies the chain property and it is not accessible (for example, let now $\mathcal{F} = \{\{1\}, \{3\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$). In fact, if we have an accessible set system satisfying the chain property, then the same system but without the empty set (or without all subsets of cardinality less then some k) is not accessible, but satisfies the chain property. The analogy statements are correct for up-accessibility.

Examples of chain systems include convex geometries (see proposition 8) and complement systems called antimatroids, matroids and all independence systems (matchings, cliques, independent sets of a graph). Consider a less common example.

Example 6 *For a graph* $G = (V,E)$, *the set system* $(V, \mathcal{S})$ *given by*

$$\mathcal{S} = \{A \subseteq V : (A, E(A)) \text{ is a connected subgraph of } G\},$$

is a chain system. The example is illustrated in Figure 1.



(a) (b)

Fig. 1. $G = (V,E)$ (a) and a family of connected subgraphs (b).

To show that $(V, \mathcal{S})$ is a chain system consider some $A,B \in \mathcal{S}$ such that $A \subset B$. We are to prove that there exists an $b \in B - A$ such that $A \cup b \in \mathcal{S}$. Since B is a connected subgraph, there is an edge $e = (a, b)$, where $a \in A$ and $b \in B - A$. Hence, $A \cup b \in \mathcal{S}$.

For a set $X \in \mathcal{F}$, let $ex(X) = \{x \in X : X - x \in \mathcal{F}\}$ be the set of *extreme points* of X. Originally, this operator was defined for closure systems [9]. An element $e \in A$ was called an *extreme point* if $e \notin \tau (A - e)$. Our definition does not demand the existing of a closure operator, but when the set system $(E, \mathcal{F})$ is a convex geometry $ex(X)$ becomes the set of original extreme points of a convex set X.

Note, that accessibility means that for each non-empty $X \in \mathcal{F}$, $ex(X) \neq \varnothing$.

Definition 7 *The operator* $ex : \mathcal{F} \to 2^E$ *satisfies the heritage property if* $X \subseteq Y$ *implies* $ex(Y) \cap X \subseteq ex(X)$ *for all* $X, Y \in \mathcal{F}$.

We choose the name *heritage property* following B. Monjardet [26]. This condition is well-known in the theory of choice functions where one uses also alternative terms like *Chernoff condition* [7] or *property* α [31]. This property is also known in the form $X - ex(X) \subseteq Y - ex(Y)$.

The heritage property means that $Y - x \in \mathcal{F}$ implies $X - x \in \mathcal{F}$ for all $X, Y \in \mathcal{F}$ with $X \subseteq Y$ and for all $x \in X$.

The extreme point operator of a closure system satisfies the heritage property, but the opposite statement in not correct. Indeed, consider the following example illustrated in Figure 2 (a): let $E = \{1, 2, 3, 4\}$ and

$$\mathcal{F} = \{\varnothing, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,4\}, \{3,4\}, \{1,2,3\}, E\}.$$

It is easy to check that the extreme point operator ex satisfies the heritage property, but the set system $(E, \mathcal{F})$ is not a closure system ($\{2, 4\} \cap \{3, 4\} \notin \mathcal{F}$). It may be mentioned that this set system does not satisfy the chain property. Another example (Figure 2 (b)) shows that the chain property is also not enough for a set system to be a closure system. Here

$$\mathcal{F} = \{\varnothing, \{1\}, \{4\}, \{1,3\}, \{3,4\}, \{1,2,3\}, \{2,3,4\}, E\},$$

and the constructed set system satisfies the chain property, but is not a closure set ($\{1, 3\} \cap \{3, 4\} \notin F$).

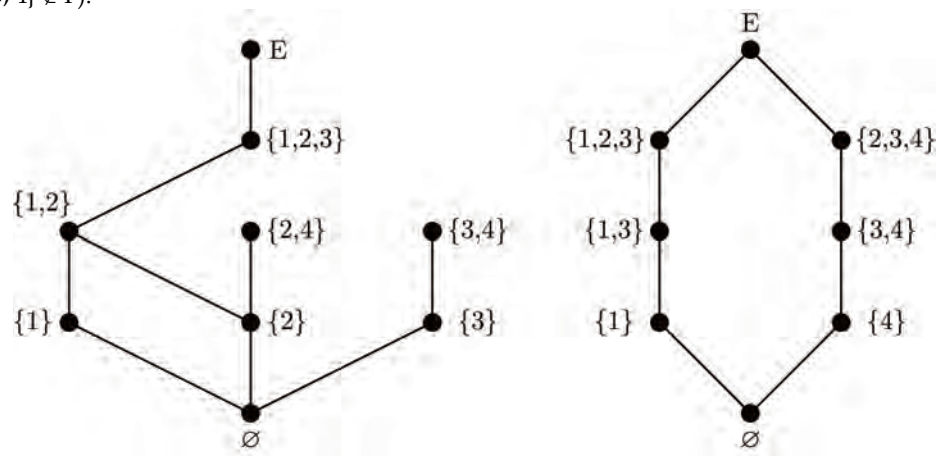


Fig. 2. Heritage property (a) and chain property (b).

Proposition 8 *A set system* $(E, \mathcal{F})$ *is a convex geometry if and only if*

(1) $\varnothing \in \mathcal{F}$, $E \in \mathcal{F}$

(2) *the set system* $(E, \mathcal{F})$ *satisfies the chain property*
(3) *the extreme point operator ex satisfies the heritage property.*

Proof. Let a set system $(E, \mathcal{F})$ be a convex geometry. Then the first condition automatically follows from the convex geometry definition. Prove the second condition. Consider $X, Y \in \mathcal{F}$, and $X \subset Y$. From (4) it follows that there is a chain $X = X_0 \subset X_1 \subset ... \subset X_k = E$ such that $X_i = X_{i-1} \cup x_i$ and $X_i \in$ F for $0 \le i \le k$. Let j be the least integer for which $X_j \supseteq Y$. Then $X_{j-1} \not\supseteq Y$, and $x_j \in Y$. Thus, $Y - x_j = Y \cap X_{j-1} \in \mathcal{F}$. Since $x_j \notin X$, the chain property is proved. To prove that $ex(Y) \cap X \subseteq ex(X)$, consider $p \in ex(Y) \cap X$, then $Y - p \in \mathcal{F}$ and $X \cap (Y - p) = X - p \in \mathcal{F}$, i.e., $p \in ex(X)$.
Conversely, let us prove that the set system $(E, \mathcal{F})$ is a convex geometry. We are to prove both up-accessibility and that $X, Y \in \mathcal{F}$ implies $X \cap Y \in \mathcal{F}$. Since $E \in \mathcal{F}$, up-accessibility follows from the chain property. Consider $X, Y \in \mathcal{F}$. Since $E \in \mathcal{F}$, the chain property implies that there is a chain $X = X_0 \subset X_1 \subset ... \subset X_k = E$ such that $X_i = X_{i-1} \cup x_i$ and $X_i \in \mathcal{F}$ for $0 \le i \le k$. If j is the least integer for which $X_j \supseteq Y$, then $X_{j-1} \not\supseteq Y$, and $x_j \in Y$. Since $x_j \in ex(X_j)$, we obtain $x_j \in ex(Y)$. Continuing the process of clearing Y from the elements that are absent in X, eventually we reach the set $X \cap Y \in \mathcal{F}$. ■

3. Monotone linkage functions

Monotone linkage functions were introduced by Joseph Mullat [29].
A function π: $E \times 2^E \to \mathbf{R}$ is called a *monotone linkage function* if

$$\text{for each } X, Y \subseteq E \text{ and } x \in E,\ X \subseteq Y \text{ implies } \pi(x, X) \le \pi(x, Y). \tag{5}$$

For each $X \subseteq E$ define function $F : (2^E - \varnothing) \to \mathbf{R}$ as follows

$$F(X) = \min_{x \in X} \pi(x, X). \tag{6}$$

Example 9 *Consider a graph* $G = (V,E)$, *where* V *is a set of vertices and* E *is a set of edges. Let* $\deg_H(x)$ *denote the degree of vertex* x *in the induced subgraph* $H \subseteq G$. *It is easy to see that function* $\pi(x,H) = \deg_H(x)$ *is monotone linkage function and function* $F(H)$ *returns the minimal degree of subgraph* H.

Example 10 *Consider a proximity graph* $G = (V,E,W)$, *where* w_{ij} *represents the degree of similarity of objects* i *and* j. *A higher value of* w_{ij} *reflects a higher similarity of objects* i *and* j. *Define a monotone linkage function* $\pi(i,H) = \sum_{j \in H} w_{ij}$, *that measures proximity between subset* $H \subseteq V$ *and their element* i. *Then the function* $F(H) = \min_{i \in H} \pi(i, H)$ *can be interpreted as a measure of density of the set* H.

It was shown [23], that for every monotone linkage function π, function F is quasi-concave on the Boolean 2^E. Moreover, each quasi-concave function may be defined by a monotone linkage function. In this section we investigate this relation on different families of sets.
For any function F defined on a set system $(E, \mathcal{F})$, we can construct the corresponding linkage function

$$\pi_F(x, X) = \begin{cases} \max\limits_{A \in [x,X]_{\mathcal{F}}} F(A), & x \in X \text{ and } [x, X]_{\mathcal{F}} \neq \varnothing \\ \min\limits_{A \in \mathcal{F}} F(A), & \text{otherwise} \end{cases} \tag{7}$$

where $[x,X]_{\mathcal{F}} = \{A \in \mathcal{F} : x \in A \text{ and } A \subseteq X\}$.
The function π_F is monotone. Indeed, if $x \in X$ and $[x,X]_{\mathcal{F}} \neq \emptyset$, then $X \subseteq Y$ implies $[x,X]_{\mathcal{F}} \neq \emptyset$ and

$$\pi_F(x,X) = \max_{A \in [x,X]_{\mathcal{F}}} F(A) \leq \max_{A \in [x,Y]_{\mathcal{F}}} F(A) = \pi_F(x,Y).$$

If $x \in X$ and $[x,X]_{\mathcal{F}} = \emptyset$, then $X \subseteq Y$ implies

$$\pi_F(x,X) = \min_{A \in \mathcal{F}} F(A) \leq \pi_F(x,Y).$$

It is easy to verify the remaining cases.
In the sequel we will consider various types of set systems. At first, we investigate the set systems closed under union, i.e., we study quasi-concave functions on join-semilattices.
Theorem 11 *A set function F defined on a join-semilattice $\mathcal{S}$ is a quasi-concave function if and only if there exists a monotone linkage function π such that $F(X) = \min_{x \in X} \pi(x,X)$ for each $X \in \mathcal{S} - \emptyset$.*

Proof. If a monotone linkage function π is given, then $F(X \cup Y) = \pi(x^*, X \cup Y)$, where $x^* \in \arg\min_{x \in X \cup Y} \pi(x, X \cup Y)$ [1]. Without loss of generality, assume that $x^* \in X$. Thus,

$$F(X \cup Y) = \pi(x^*, X \cup Y) \geq \pi(x^*, X) \geq F(X) \geq \min\{F(X), F(Y)\}.$$

Conversely, if we have a quasi-concave set function F, we can define the monotone linkage function $\pi_F(x,X)$ using the definition 7. Let us denote $G(X) = \min_{x \in X} \pi_F(x,X)$, and prove that $F = G$ on $\mathcal{S} - \emptyset$.
Now, for each $X \in \mathcal{S} - \emptyset$

$$G(X) = \min_{x \in X} \pi_F(x,X) = \pi_F(x^*,X) = \max_{A \in [x^*,X]_{\mathcal{S}}} F(A) \geq F(X),$$

where $x^* \in \arg\min_{x \in X} \pi_F(x,X)$.
On the other hand,

$$G(X) = \min_{x \in X} \pi_F(x,X) = \min_{x \in X} F(A^x),$$

where A^x is a set from $[x,X]_{\mathcal{S}}$ on which the value of the function F is maximal i.e.,

$$A^x = \arg\max_{A \in [x,X]_{\mathcal{S}}} F(A).$$

From quasi-concavity of F it follows that

$$\min_{x \in X} F(A^x) \leq F(\bigcup_{x \in X} A^x) = F(X).$$

Therefore, $G(X) \leq F(X)$, and, hence, $F(X) = \min_{x \in X} \pi_F(x,X)$. ■
Now, consider set systems that are not closed under union.

[1] argmin $f(x)$ denote the set of arguments that minimize the function f.

Let $(E, \mathcal{F})$ be an accessible set system. Denote $\mathcal{F}^+ = \mathcal{F} - \varnothing$. Then, having the monotone linkage function $\pi_{\mathcal{F}}$, we can construct for all $X \in \mathcal{F}^+$ the set function

$$G_F(X) = \min_{x \in ex(X)} \pi_F(x, X).$$

It is easy to see that

$$G_F(X) \geq F(X), \text{ for each } X \in \mathcal{F}^+. \tag{8}$$

Indeed, for each $X \in \mathcal{F}^+$

$$G_F(X) = \min_{x \in ex(X)} \pi_F(x, X) = \pi_F(x^*, X) = \max_{A \in [x, X]_{\mathcal{F}}} F(A) \geq F(X),$$

where $x^* \in \arg\min_{x \in ex(X)} \pi_F(x, X)$.

The following theorem finds conditions on the set system $(E, \mathcal{F})$ and on the function F ensuring that the function G_F coincides with F.

Theorem 12 *[18] Let $(E, \mathcal{F})$ be an accessible set system. Then for every quasi-concave set function $F : \mathcal{F}^+ \to \mathbf{R}$*

$$G_F = F \text{ on } \mathcal{F}^+$$

if and only if the set system $(E, \mathcal{F})$ satisfies the chain property.

Thus, for an accessible set system satisfying the chain property each quasi-concave function F determines a monotone linkage function π_F, and a set function defined as a minimum of this monotone linkage function π_F coincides with the original function F.

As examples of such set systems may be considered greedoids [16] that include matroids and antimatroids, and antigreedoids including convex geometries. By an antigreedoid we mean a set system $(E, \mathcal{F})$ such that its complement set system $(E, \overline{\mathcal{F}})$ is a greedoid.

Note, that if F is not quasi-concave, the function G_F does not necessarily equal F. For example, let $\mathcal{F} = \{\varnothing, \{1\}, \{2\}, \{1, 2\}\}$ and let

$$F(X) = \begin{cases} 0, & X = \{1, 2\} \\ 1, & \text{otherwise.} \end{cases}$$

The function F is not quasi-concave, since $F(\{1\} \cup \{2\}) < \min(F(\{1\}), F(\{2\}))$. It is easy to check that here $G_F \neq F$, because $\pi_F(1, \{1, 2\}) = \pi_F(2, \{1, 2\}) = 1$, and so $G_F(\{1, 2\}) = 1$. Moreover, the function G_F is quasi-concave. To understand this phenomenon, consider the opposite process.

Let $(E, \mathcal{F})$ be an accessible set system. If a monotone linkage function $\pi : E \times 2^E \to \mathbf{R}$ is given, we can construct the set function $F_\pi : \mathcal{F}^+ \to \mathbf{R}$:

$$F_\pi(X) = \min_{x \in ex(X)} \pi(x, X). \tag{9}$$

To extend this function to the whole set system $(E, \mathcal{F})$ define

$$F_\pi(\varnothing) = \min_{(x, X)} \pi(x, X).$$

Theorem 13 *[18] Let* $(E, \mathcal{F})$ *be an accessible set system. Then the following statements are equivalent*

(i) the extreme point operator $ex : \mathcal{F} \to 2^E$ *satisfies the heritage property.*

(ii) for every monotone linkage function π *the function* F_π *is quasi-concave.*

Thus, if a set system $(E, \mathcal{F})$ is accessible and the operator *ex* satisfies the heritage property, then for each set function F, defined on $(E, \mathcal{F})$, one can build the quasi-concave set function G_F that is an upper bound of the original function F. If, in addition, the set system has the chain property, the class of set functions defined as the minimum values of monotone linkage functions coincides with the class of quasi-concave set functions.

Corollary 14 *A set function F defined on a convex geometry* $(E, \mathcal{F})$ *is quasi-concave if and only if there exists a monotone linkage function* π *such that* $F(X) = \min_{x \in ex(X)} \pi(x, X)$ *for each* $X \in \mathcal{F} - \varnothing$.

Another approach to the result of Theorem 13 is in extending the function F to the Boolean 2^E by building a new linkage function π_{ex}.

Let $(E, \mathcal{F})$ be an accessible set system and π be a monotone linkage function. Define

$$\pi_{ex}(x, X) = \begin{cases} \pi(x, X), & x \in ex(X) \\ \pi(x, X) + c, & \text{otherwise} \end{cases} \tag{10}$$

where $c = \max_{(x,X) \in E \times 2^E} \pi(x, X) + 1$.

Theorem 15 *[25] Let* $(E, \mathcal{F})$ *be an accessible set system and the extreme point operator ex satisfies the heritage property. If function* π *is a monotone linkage function, then*

(i) function π_{ex} *is also monotone and*

(ii) its function $F_{\pi_{ex}}(X) = \min_{x \in X} \pi_{ex}(x, X)$ *coincides with the function* $F_\pi(X) = \min_{x \in ex(X)} \pi(x, X)$ *for each* $X \in \mathcal{F} - \varnothing$.

Now, Theorem 13 immediately follows from the properties of quasi-concave functions on the Boolean [23].

Remark 16 *[25] Any extreme point operator ex satisfying the heritage property may be represented by some monotone linkage function* π *in the following way*

$$ex(X) = \{x \in X : \pi(x, X) \leq u\} \tag{11}$$

and vice versa, if the linkage function π *is monotone, the operator ex defined by (11) satisfies the heritage property.*

4. Maximizers of quasi-concave functions

Consider the following optimization problem: given a monotone linkage function π, and an accessible set system $(E, \mathcal{F})$, find a feasible set $A \in \mathcal{F}^+$, such that $F_\pi(A) = \max\{F_\pi(B) : B \in \mathcal{F}^+\}$, where the function F_π is defined by (9). From quasi-concavity of the function F_π it follows that the set of optimal solutions is a join-semilattice with a unique maximal element. Our goal is to find this maximal element, which we call the $\cup -$ *maximizer*. For instance, for the functions defined in Example 9 $\cup -$ maximizer is the largest subgraph with the maximum minimum degree. In Example 10 we look for the largest subset with the highest density.

A greedy-type algorithm for finding the $\cup -$ maximizer on the Boolean was constructed by Mullat [29] and has been effectively applied in data mining [22], biology [33], and for computer vision [35].

Here we want to investigate the more general set systems.

4.1 Chain algorithm on convex geometries

A convex geometry is a closure system, and so closed under intersection. Hence, each set $X \subseteq E$ has an unique cover which is a closure of X, i.e., $\tau(X)$ and the family of feasible sets F of a convex geometry $(E, \mathcal{F})$ form a join-semilattice $L_{\mathcal{F}}$, with the lattice operation: $X \vee Y = \tau(X \cup Y)$. Hence, for convex geometries the inequality (2) reads as follows $F(X \vee Y) \geq \min\{F(X), F(Y)\}$ for each $X, Y \in L_{\mathcal{F}}$.

Consider the special structure that quasi-concave function F_π determines on a convex geometry. It has been already noted that the family of feasible sets maximizing function F_π is a join-semilattice with a unique maximal element. Denote this family by $\mathcal{T}^0$, and let a^0 be the value of function F_π on the sets from $\mathcal{T}^0$. We denote by $\mathcal{T}^1$ the family of sets, which maximize function F_π over $\mathcal{F}^+ - \mathcal{T}^0$, and by a^1 the value of function F_π on these sets. Continuing this process, we have $\mathcal{F}^+ = \cup_{i=0}^{t} \mathcal{T}^i$, where $t + 1$ is a number of different values of function F_π. It is easy to see that $\mathcal{L}_j = \cup_{i=0}^{j} \mathcal{T}^i$ is a subsemilattice of $L_{\mathcal{F}}$, where $\mathcal{L}_j = \{X \in \mathcal{F}^+ : F_\pi(X) \geq a^j\}$. We call these subsemilattices *upper level semilattices*. Denote by K^j the maximal element - **1** of the upper level semilattice $\mathcal{L}_j$. Since $\mathcal{L}_i \subseteq \mathcal{L}_{i+1}$, we obtain $K^0 \subseteq K^1 \subseteq ... \subseteq K^t$, where K^t is **1** of the join-semilattice $L_{\mathcal{F}}$, i.e., $K^t = E$.

Let $K^0 = H^0 \subset H^1 \subset ... \subset H^r = K^t$ be the subchain of all different **1-s** of the chain $K^0 \subseteq K^1 \subseteq ... \subseteq K^t$. Thus, to find a $\cup$ − maximizer, we have to find just H^0. In fact, we construct an algorithm that finds the complete chain $H^0 \subset H^1 \subset ... \subset H^r = E$ of different **1-s**. This chain of "local maximizers"[2] has a number of interesting applications [24].

For any real number u we define the *u-level set* of a family $\mathcal{F}$ as

$$\mathcal{F}_u = \{X \in \mathcal{F}^+ : F_\pi(X) > u\}.$$

It is clear that if F_π is quasi-concave, then the u-level set of a join-semilattice is a join-semilattice as well. The input of the following algorithm is a threshold u and a set $X \in \mathcal{F}$, while it returns **1** of non-empty $(\mathcal{F}^+ \cap [\varnothing, X])_u$. The algorithm is motivated by procedures from [28] and [29].

The Level-Set Algorithm (u, X)

1. Set $A = X$
3. While $A \neq \varnothing$ do
   3.1 Set $I_u(A) = \{x \in ex(A) : \pi(x, A) \leq u\}$
   3.2 If $I_u(A) = \varnothing$ then stop and return A
   3.3 Set $A = A - I_u(A)$
4. Return A.

Theorem 17 *Let* $(E, \mathcal{F})$ *be a convex geometry. Then, for every monotone linkage function* π *and the corresponding function* $F_\pi(X) = \min_{x \in ex(X)} \pi(x, X)$ *the Level-Set Algorithm* (u,X) *returns* **1** *of non-empty semilattice* $(\mathcal{F}^+ \setminus [\varnothing, X])_u$ *and returns* $\varnothing$ *when this u-level set is empty.*

Proof. At first, note that $A - I_u(A) = \cap_{x \in I_u(A)} (A - x)$. Since any convex geometry is closed under intersection, then all sets generated by the algorithm belong to the convex geometry.

---

[2] Indeed, for each $A \in \mathcal{F}^+$, and for each null H^l, if $A \nsubseteq H^l$ then $F(A) < F(H^l)$.

Consider the case when the algorithm returns $A \neq \varnothing$. Since $I_u(A) = \varnothing$, then $F_\pi(A) > u$, i.e. $A \in (\mathcal{F}^+ \cap [\varnothing, X])_u$. It remains to be proven that A is the null of the u-level set, i.e., that $B \in (\mathcal{F}^+ \cap [\varnothing, X])_u$ implies $A \supseteq B$. Suppose the opposite was true, and let $X = X_0 \supset X_1 \supset ... \supset X_k = A$ be a sequence of sets generated by the algorithm, where $X_{i+1} = X_i - I_u(X_i)$ for $0 \leq i < k$. Since $B \in (\mathcal{F}^+ \cap [\varnothing, X])_u$, then $X \supseteq B$. On the other hand, since $A \not\supseteq B$, there exists the least integer j for which $X_j \not\supseteq B$. Then $X_{j-1} \supseteq B$, and there is $x_j \in I_u(X_{j-1})$ that belongs to B. So, $X_{j-1} \supseteq B$, $x_j \in B$ and $x_j \in ex(X_{j-1})$, then from heritage property it follows that $x_j \in ex(B)$. Hence, monotonicity of function π implies $F(B) \leq \pi(x_j, B) \leq \pi(x_j, X_{j-}^{\ 1}) \leq u$, a contradiction.

If the algorithm returns $A = \varnothing$, then $(\mathcal{F}^+ \cap [\varnothing, X])_u = \varnothing$. Assuming the opposite, then there is a non-empty set $B \in (\mathcal{F}^+ \cap [\varnothing, X])_u$. By analogy, with the first part of the proof, we obtain that $F_\pi(B) \leq u$, a contradiction. ■

The following Chain Algorithm finds the chain of all local maximizers for a non-empty join-semilattice $L_{\mathcal{F}}$.

The Chain Algorithm (E, π, $\mathcal{F}$)

1. Set $\Gamma_0 = E$
2. $i = 0$
3. While $\Gamma_i \neq \varnothing$ do
   3.1 $u = F(\Gamma_i)$
   3.2 Γ_{i+1} = Level-Set(u, Γ_i)
   3.3 $i = i + 1$
4. Return the chain $\Gamma_0 \supset \Gamma_1 \supset ... \supset \Gamma_{i-1}$.

Theorem 18 *Let* (E, $\mathcal{F}$) *be a convex geometry. Then, for every monotone linkage function* π *and the corresponding function* $F_\pi(X) = \min_{x \in ex(X)} \pi(x, X)$, *the Chain Algorithm returns the chain* $\Gamma_0 \supset \Gamma_1 \supset ... \supset \Gamma_p$, *which coincides with* $H^0 \subset H^1 \subset ... \subset H^r$ *- the chain of all different* ***1-s*** *of the upper level semilattices*.

Proof. First, prove that for each $l = 0, 1, ..., p$, Γ_l is **1** of some upper level semilattice. It is clear, that if $F_\pi(\Gamma_l) = a_j$, then $\Gamma_l \in L_j$. To prove that Γ_l is **1** of L_j , we have to show that for each $A \in \mathcal{F}^+$, $A \not\subseteq \Gamma_l$ implies $F_\pi(A) < F_\pi(\Gamma_l)$. Suppose that the opposite is true, and let k be the least integer for which there exists $A \in \mathcal{F}^+$, such that $A \not\subseteq \Gamma_k$ and $F_\pi(A) \geq F_\pi(\Gamma_k)$. Note that $k > 0$, because $\Gamma_0 = E$ is **1** of join-semilattice $L_{\mathcal{F}}$, and so $A \not\subseteq \Gamma_0$ never holds. The structure of the Chain Algorithm implies $F_\pi(\Gamma_k) > F_\pi(\Gamma_{k-1})$. Hence $F_\pi(A) > F_\pi(\Gamma_{k-1})$ and, consequently, $A \subseteq \Gamma_{k-1}$. Thus $A \in (\mathcal{F}^+ \cap [\varnothing, \Gamma_{k-1}])_u$, where $u = F_\pi(\Gamma_{k-1})$. On the other hand, from Theorem 17 it follows that Γ_k is **1** of $(\mathcal{F}^+ \cap [\varnothing, \Gamma_{k-1}])_u$, i.e., $A \subseteq \Gamma_k$, a contradiction.

It remains to show that for each H^j there exists $l \in \{0, 1, ...p\}$ such that $\Gamma_l = H^j$. Assume the opposite, and let H^j be a maximal **1** for which the statement is not correct. Since $H^r = \Gamma_0$, then $j < r$, i.e., there exists $l \in \{0, 1, ...p\}$ such that $H^{j+1} = \Gamma_l$. From $F_\pi(H^j) > F_\pi(H^{j+1}) = F_\pi(\Gamma_l)$ and $H^j \subset H^{j+1} = \Gamma_l$, it follows that $H^j \in (\mathcal{F}^+ \cap [\varnothing, \Gamma_l])_u$, where $u = F_\pi(\Gamma_l)$. Thus $H^j \subseteq \Gamma_{l+1}$, where Γ_{l+1} is **1** of $(\mathcal{F}^+ \cap [\varnothing, \Gamma_l])_u$. On the other hand, since Γ_{l+1} is **1** of some upper level semilattice and H^j is the closest **1** to H^{j+1}, then $\Gamma_{l+1} \subseteq H^j \subseteq H^{j+1} = \Gamma_l$. Hence $H^j = \Gamma_{l+1}$, a contradiction. ■

Corollary 19 *Let* (E, $\mathcal{F}$) *be a convex geometry. Then, for every monotone linkage function* π, *the Chain Algorithm finds a* $\cup -$ *maximizer of the quasi-concave function* $F_\pi = \min_{x \in ex(X)} \pi(x, X)$.

Actually, a convex geometry is the unique structure on which the Chain Algorithm produces optimal solutions. To prove it we have to show that for each set system that is not a convex geometry there exists a monotone linkage function for which the Chain Algorithm does not find the $\cup$ − maximizer. It is obvious that if a set system is not up-accessible, then the Chain Algorithm may not reach the optimal solution.

Now, consider an up-accessible set system $(E, \mathcal{F})$ that does not satisfy the heritage property, i.e., there exists $A,B \in \mathcal{F}$ such that $A \subset B$, and there is $a \in A$ such that $B - a \in \mathcal{F}$ and $A - a \notin \mathcal{F}$. Up-accessibility of the set system $(E, \mathcal{F})$ implies that there exists a sequence of feasible sets

$$E = B_0 \supset B_1 \supset \ldots \supset B_p = B \supset B_{p+1} = B - a,$$

where $B_i = B_{i-1} - a_i$ for $1 \le i \le p$, and $a_{p+1} = a$. Define a linkage function π on pairs (x,X) where $X \subseteq E$, $X \neq \varnothing$ and $x \in X$:

$$\pi(x,X) = \begin{cases} 1, & (x = a_{i+1} \text{ and } 0 \le i \le p) \text{ or } (B - a \supseteq X,\ X \neq \varnothing \text{ and } x \in X) \\ 2, & \text{otherwise.} \end{cases}.$$

It is easy to verify that function π is monotone. Then the Chain Algorithm generates only one set E, on which the value of the function F_π is equal to 1, while $F_\pi(A) = 2$. Thus, the Chain Algorithm does not find a feasible set that maximizes the function F_π. So, we have the following theorem.

Theorem 20 *Let* $(E, \mathcal{F})$ *be an accessible and an up-accessible set system. Then the following statements are equivalent*

(1) *the set system* $(E, \mathcal{F})$ *is a convex geometry*

(2) *The Chain Algorithm finds a* $\cup$ − *maximizer of the quasi-concave function* $F_\pi = \min_{x \in ex(X)} \pi(x,X)$ *for every monotone linkage function* π

The Chain Algorithm is of greedy type, since it is based on the best choice principle: it chooses on each step the extreme elements (with respect to the linkage function) and, in such a way, approaches the optimal solution. The run-time of the algorithm depends largely on the efficiency of linkage function computation. For instance, in Example 10 the complexity of computing the initial linkage function values $\pi(x, V)$ for all the vertices in V is $O(|E|)$, where E is a set of edges. For straightforward implementation the time required for finding the minimum value is $O(|V|)$. After deleting the vertex with minimum value of π, the time required for updating the linkage function values for all the neighboring vertices of the deleted vertex is $O(|V|)$, since the update can be carried out in time $O(1)$ by subtracting the corresponding weight w_{ij}. So, the total time required for straightforward implementation of the Chain Algorithm in Example 10 is $O(|E| + |V|^2) = O(|V|^2)$.

In general case, the Chain Algorithm finds the $\cup$ − maximizer of a convex geometry (E,F) in $O(P|E| + U|E|^2)$ time, where P is the maximum complexity of computing the initial linkage function values $\pi(x,E)$ over all $x \in E$, and U is the maximum complexity of updating the linkage function values.

For some special linkage functions the running time can be improved by using more efficient data structure that will be discussed in the next section.

4.2 Chain algorithm on join-semilattices

Now we have a monotone linkage function π, and a join-semilattice $\mathcal{S} \subseteq 2^E$, and we are interested in finding a maximal maximizer of the function F_π defined as $F(X) = \min_{x \in X} \pi(x,X)$ according to (6).

Since a join-semilattice should not to be up-accessible, we have to find another way to reach each feasible set.
Consider the following operator:

$$\omega(X) = \begin{cases} \cup\{A \mid A \subseteq X, A \in \mathcal{S}\}, & [\varnothing, X]_{\mathcal{S}} \neq \varnothing \\ \varnothing, & otherwise \end{cases}. \quad (12)$$

If $\mathcal{S}$ is a join-semilattice, $\omega(X)$ is the largest set in $\mathcal{S}$ contained in X (if such a set exists). In other words, $\omega(X)$ is the **1** of the subsemilattice $[\varnothing, X]_{\mathcal{S}}$ if the subsemilattice is not empty, and $\varnothing$, otherwise.
Note, that a join-semilattice $\mathcal{S}$ should not have the minimum element, and we use the element $\varnothing$ only to complete the definition of the operator ω.
The operator ω is called *interior* (dual to closure) operator:

(i) $\omega(X) \subseteq X$,
(ii) $\omega(X) = \omega(\omega(X))$,
(iii) $X \subseteq Y \Rightarrow \omega(X) \subseteq \omega(Y)$.

$\omega(X)$ is *an interior* of X. The fixed points of ω ($X = \omega(X)$) are called the open sets of ω and forms the dual closure system [27]. A set system $(E, \mathcal{S})$ is a dual closure system if and only if the complement set system $(E, \mathcal{S})$ is a closure system. If $\mathcal{S}$ is a join-semilattice and the operator ω is defined by (12), then the family of open sets coincides with $\mathcal{S}$, excluding, possible, the empty set.

We assume that for each $X \subseteq E$ a procedure for finding interior $\omega(X)$ is available. Later we will consider some examples of procedures building interior efficiently.
From quasi-concavity of function F_π it follows that the set of maximizers is a join-semilattice with a unique maximal element. It is easy to see that the structure of upper level semilattices investigated for convex geometries holds for join-semilattice as well. To obtain the chain $H^0 \subset H^1 \subset \ldots \subset H^r = E$ of different ***1-s*** we use the Chain Algorithm with the following transformation: instead of assigning some set we replace it by its interior.

The Level-Set Algorithm-JS (u, X)

1. Set $A = \omega(X)$
3. While $A \neq \varnothing$ do
   3.1 Set $I_u(A) = \{x \in A : \pi(x, A) \leq u\}$
   3.2 If $I_u(A) = \varnothing$ then stop and return A
   3.3 Set $A = \omega(A - I_u(A))$
4. Return A.

The Chain Algorithm-JS (E, π, F)

1. Set $\Gamma_0 = \omega(E)$
2. $i = 0$
3. While $\Gamma_i \neq \varnothing$ do
   3.1 $u = F(\Gamma_i)$
   3.2 Γ_{i+1} = Level-Set(u, Γ_i)
   3.3 $i = i + 1$
4. Return the chain $\Gamma_0 \supset \Gamma_1 \supset \ldots \supset \Gamma_{i-1}$.

Similarly with the proof of Theorem 18 we obtain the following result.

Theorem 21 *Let $\mathcal{S} \subseteq 2^E$ be a non-empty join-semilattice. Then, for every monotone linkage function π and the corresponding function $F(X) = \min_{x \in X} \pi(x, X)$, the Chain Algorithm-JS returns the chain $\Gamma_0 \supset \Gamma_1 \supset ... \supset \Gamma_p$, which coincides with $H^0 \subset H^1 \subset ... \subset H^r$ - the chain of all different **1-s** of the upper level semilattices.*

Consider the complexity of the Chain Algorithm-JS. The run-time of the algorithm depends largely on the efficiency of interior construction. The Chain Algorithm-JS finds the $\cup$ − maximizer of a join-semilattice $(E,\mathcal{S})$ in $O(|E|(P + T + U|E|))$ time, where P is the maximum complexity of computing the initial linkage function values $\pi(x,E)$ over all $x \in E$, U is the maximum complexity of updating the linkage function values, and T is the maximum complexity of interior construction.

4.2.1 Algorithms for interior construction

The efficiency of the interior construction depends on the representation of a join-semilattice. Here we consider a join-semilattice specified by a quasi-concave function. In addition, we consider an antimatroid that is a specific case of a join-semilattice.

1. Quasi-Concave constraints. Assume that the family $\Omega \subseteq 2^E$ of feasible sets is determined by the following constraints: for each $H \in \Omega$, $\widehat{F}(H) > \alpha$, where $\widehat{F}$ is a quasi-concave function defined by a monotone linkage function $\widehat{\pi}$. It is easy to see that the set Ω is an α-level set of 2^E, i.e., $\Omega = \{X \subseteq E : \widehat{F}(X) > \alpha\}$. Since $\widehat{F}$ is a quasi-concave function, the set Ω is a join-semilattice. The problem is to find interior $\omega(X)$ over Ω for every set $X \subseteq E$, i.e., to find **1** of the non-empty join-semilattice $\Omega \cap [\varnothing, X]$. Note that the Level-Set Algorithm(α,X) enables us to find **1** of the non-empty join-semilattice $(2^E \cap [\varnothing, X])_\alpha$, i.e., $\omega(X)$ over Ω. The modified Level-Set Algorithm is as follows:

Quasi-Concave Interior Algortihm (α,X)

1. Set $A = X$
3. While $A \neq \varnothing$ do
   3.1 Set $I_\alpha(A) = \{x \in A : \widehat{\pi}(x,A) \leq \alpha\}$
   3.2 If $I_\alpha(A) = \varnothing$ then stop and return A
   3.3 Set $A = A - I_\alpha(A)$
4. Return A.

The Quasi-Concave Interior Algorithm finds the interior $\omega(X)$ in $O(P|X| + U|X|^2)$ time, where P is the maximum complexity of computing the initial linkage function values $\widehat{\pi}(x,X)$ over all $x \in X$, and U is the maximum complexity of updating the linkage function values.

2. Antimatroids. There are many equivalent axiomatizations of antimatroids, that may be separated into two categories: antimatroids defined as set systems and antimatroids defined as languages. An algorithmic characterization of antimatroids based on the language definition was introduced in [6]. Another algorithmic characterization of antimatroids that depicted them as set systems was developed in [17]. While classical examples of antimatroids connect them with posets, chordal graphs, convex geometries, etc., game theory gives a framework in which antimatroids are interpreted as permission structures for coalitions [4]. There are also rich connections between antimatroids and cluster analysis [20]. In mathematical psychology, antimatroids are used to describe feasible states of knowledge of a human learner [12].

Definition 22 *[16]A non-empty set system* $(E, \mathcal{S})$ *is an antimatroid if*

(A1) $(E, \mathcal{S})$ *is an accessible set system*

(A2) for all $X, Y \in \mathcal{S}$, *and* $X \not\subseteq Y$, *there exist an* $x \in X - Y$ *such that* $Y \cup x \in \mathcal{S}$.

It is easy to see that the chain property follows from (*A*2), but these properties are not equivalent.

Proposition 23 *[5][16]For an accessible set system* $(E, \mathcal{S})$ *the following statements are equivalent:*

(i) $(E, \mathcal{S})$ *is an antimatroid*

(ii) $\mathcal{S}$ *is closed under union* $(X, Y \in \mathcal{S}) \Rightarrow X \cup Y \in \mathcal{S}$.

Therefore an antimatroid is a join-semilattice that includes the empty set. The interior operator ω defined by (12) returns for each set $X \subseteq E$ the maximal feasible subset called the *basis* of X.

Since an antimatroid $(E, \mathcal{S})$ satisfies the chain property, to find $\omega(X)$, one can build the chain $\varnothing \subset X_0 \subset X_1 \subset \ldots \subset X_m = \omega(X)$ belonging to $\mathcal{S}$.

Antimatroid Interior Algortihm$(X, \mathcal{S})$

1. $A = \varnothing$
2. Find $x \in X - A$, such that $A \cup x \in S$
   if no such x exists, then stop and return A
3. Set $A = A \cup x$ and go to 2.

The Antimatroid Interior Algorithm returns the basis $\omega(X)$ for each set $X \subseteq E$ that immediately follows from the chain property.

Let an antimatroid $(E, \mathcal{S})$ be given by a membership oracle which for each set $A \subseteq E$ decides whether $A \in \mathcal{S}$ or not. Then the Antimatroid Interior Algorithm finds the interior of a set in at most $k(k + 1)/2$ oracle calls, where $k = |X|$. Thus the complexity of interior construction is $O(|X|^2\theta)$, where θ is the complexity of the membership oracle.

Consider another way to define antimatroids. Let $P = \{x_1 < x_2 < \ldots < x_n\}$ be a linear order on E. Define

$$D_P = \{X_i : \ X_i = \{x_1, x_2, \ldots, x_i\}, \ 1 \le i \le n\} \cup \{\varnothing\}.$$

It is easy to see that (E, D_P) is an antimatroid.

Let $(E, \mathcal{S}_1)$ and $(E, \mathcal{S}_2)$ be two antimatroids. Define

$$\mathcal{S}_1 \vee \mathcal{S}_2 = \{X \cup Y : X \in \mathcal{S}_1, Y \in \mathcal{S}_2\}.$$

Then $(E, \mathcal{S}_1 \vee \mathcal{S}_2)$ is also an antimatroid [16].

Every antimatroid can be represented as a join of a family of its maximal chains. Hence, each antimatroid may be defined by a set T of linear orders as

$$\mathcal{S} = \bigvee_{P \in T} D_P. \tag{13}$$

By analogy with convex realizers of convex geometries [10] the set T is called a realizer. Thus, if $\{P_1, P_2, \ldots, P_k\}$ is a realizer of $(E, \mathcal{S})$, then each element of $\mathcal{S}$ is a join of elements in $D_{P_1}, \ldots, D_{P_k}$. Note, that each $D_{P_i} \subseteq \mathcal{S}$.

Since each (E, D_{P_i}) is an antimatroid , there are k interior operators ω_{P_i}, where $\omega_P(X) = \{y \in E : y \le_P \min \overline{X}\}$, i.e., let $P = \{x_1 < x_2 < \ldots < x_n\}$ and a minimal element of $\overline{X}$ with respect to the order P be $x_i = \min \overline{X}$, then $\omega_P(X) = \{x_1, x_2, \ldots, x_{i-1}\}$.

Proposition 24 $\omega(X) = \bigcup_{P \in T} \omega_P(X)$.

Proof. Let $A = \cup_{P \in T} \omega_P(X)$. Since for each $P \in T$, $\omega_P(X) \subseteq X$ and $\omega_P(X) \in \mathcal{S}$, then $\omega_P(X) \subseteq \omega(X)$, which implies $A \subseteq \omega(X)$. Conversely, from (13) $\omega(X) = \cup_{P \in T} X_P$, where $X_P \in D_P$. Since $\omega(X) \subseteq X$ implies $X_P \subseteq X$ for all $P \in T$, then $X_P \subseteq \omega_P(X)$ and so $\omega(X) \subseteq A$. ■

Let an antimatroid $(E, \mathcal{S})$ be given by a realizer $T = \{P_1, P_2, ..., P_k\}$, then the following algorithm builds the interior set using Proposition 24.

Ordering Interior Algorithm($X, \mathcal{S}$)

1. For $i = 1$ to k do
   1.1 build $\omega_{P_i}(X)$
2. Return $\tau(X) = \cup_{i=1}^{k} \omega_{P_i}(X)$.

A straightforward implementation of the Ordering Interior Algorithm runs in $O(k|E|)$, where k is the cardinality of a realizer.

5. Ortholog clustering

This section deals with applications of quasi-concave functions to clustering in bioinformatics. We concentrate on the one of the problem of comparative genomics. Comparative genomics is a field of biological research in which the genome sequences of different species are compared. Although living creatures look and behave in many different ways, all of their genomes consist of DNA, the chemical chain that includes the genes that code for thousands of different kinds of proteins. Thus, by comparing the sequence of the human genome with genomes of other organisms, researchers can identify regions of similarity and difference. This information can help scientists better understand the structure and function of human genes and thereby develop new strategies to combat human disease. Comparative genomics also provides a powerful tool for studying evolutionary changes among organisms.

A fundamental problem in comparative genomics is the detection of genes from different organisms that are involved in similar biological functions. This requires identification of homologous genes that are similar to each other because they originated from a common ancestor. Such genes are called *orthologs* [13].

We describe an ortholog clustering method where we require that any sequence in an ortholog cluster has to be similar to other sequence from other genomes in that ortholog cluster.

5.1 Ortholog detection using multipartite graph clustering

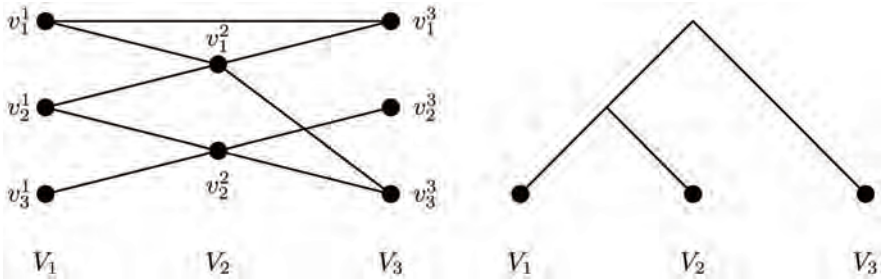
The input for the ortholog clustering problem is a set of genetic sequences along with information about the organisms they belong to. The goal is to find similar sequences from different organisms. The ortholog detection problem is complicated due to the presence of another type of very similar sequences in the same organism. These sequences, called *paralogs*, are result of duplication events when a gene in an organism is duplicated to occupy two different positions in the same genome. Although both types of genes are similar, only orthologs are likely to be involved in the same biological role. So, for detecting orthologs it is critical to focus on the similarities between genes from different organisms while ignoring the similarities between genes within an organism.

The requirement of selectively ignoring gene similarities for paralogous sequences can be conveniently represented in a multipartite graph. A graph is a multipartite if the set of

vertices in the graph may be divided into non-empty disjoint subsets, called parts, such that no two vertices in the same part have an edge connecting them. We use a multipartite graph, where different genomes correspond to different parts and the genes in a genome correspond to vertices in a part.

Another specific problem in finding ortholog clusters is that orthologous genes from closely related organisms will be much more similar than those from distantly related organisms. Fortunately, we often have estimates of evolutionary relationships between the organisms that define a hierarchical graph over the partite sets. Using this evolutionary graph, called *a phylogenetic tree*, we can correct the observed gene similarities by scaling up the similarities between the orthologs from distantly related organisms.

Consider the ortholog clustering problem with k different genomes, where the genome l, represented by V_l (l = 1, 2, ...k), contains n_l genes. Then, the similarity relationships between genes from different genomes can be represented by an undirected weighted multipartite graph $G = (V,E,W)$, where $V = \cup_{l=1}^{k} V_l$, every set V_l contains n_l vertices corresponding to n_l genes, and $E \subseteq \cup_{i \neq j} V_i \times V_j$ (i, j = 1, 2, ..., k) is a set of weighted edges representing similarities between genes. The example of a multipartite graph is illustrated in Figure 3 (a). The relationship between these genomes is given by the phylogenetic tree relating the species under study (see Figure 3 (b)).



(a) (b)

Fig. 3. Multipartite graph (a) and phylogenetic tree (b).

We consider an ortholog cluster as a largest subgraph with the highest density. For finding an ortholog cluster we assign a score $F(H)$ to any subset H of V. A score function F denotes a measure of proximity among genes in H. Then an ortholog cluster H^* is defined as the subset with the largest score value (a maximizer of F). To build a score function $F(H)$ we use Definition 6 that is based on using a linkage function $\pi(i,H)$ which measures the degree of similarity of the gene $i \in H$ to other genes in H.

Our linkage function considers the sequence similarity between genes within the ortholog cluster, their relationship to genes outside the cluster, and the phylogenetic distance between the corresponding genomes.

We require that H contains at least two genomes. So, let $H = \cup_{l=1}^{k} H_l$, where H_l is the subset of genes from V_l present in H. If $m_{ij} \geq 0$ is the similarity value between gene i from genome $g(i)$ and gene j from another genome $g(j)$, and $p(g(i), g(j))$ represents the distance between the two genomes, then the linkage function is defined as

$$\pi(i,H) = \sum_{l \neq g(i)}^{k} p(g(i),l)\Big(\sum_{j \in H_l} m_{ij} - \sum_{j \in V_l - H_l} m_{ij} \Big) \tag{14}$$

For each part $V_l \neq g(i)$ the term $\sum_{j \in H_l} m_{ij}$ aggregates the similarity values between the genes i and all other genes in the subset H_l, while the second term, $\sum_{j \in V_l - H_l} m_{ij}$, estimates the relationship between gene i and genes from genome l that are not included in H_l. A large positive difference between two terms ensures that the gene i is highly similar to genes in H_l and at the same time very dissimilar from the genes not included in H_l. From a clustering point of view, this ensures large values of intra-cluster homogeneity and inter-cluster separability for extracted cluster.

The scaling term $p(g(i),l)$ is used for correcting the observed sequence similarities by magnifying the sequence similarities corresponding to genomes which diverged in ancient times. Given the phylogenetic tree relating the species under study, the distance $p(g(i), g(j))$ between genomes $g(i)$ and $g(j)$ is defined as the height, $h_{g(i),g(j)}$, of the subtree rooted at the last common ancestor of these genomes. When the species are closely related, a function that depends on $h_{g(i),g(j)}$, but grows slower will better model the distance between the species. Choosing an appropriately growing function is critical because a faster growing function will have the undesirable effect of clustering together sequence from distance species but leaving out the sequence from closely related species. So, in this case the distance $p(g(i), g(j))$ may be defined as $(1 + \log_2 h_{g(i),g(j)})$.

It is easy to verify that function π defined in (14) is monotone. Firstly note that the distance $p(g(i), g(j)) \geq 0$ has no effect on the monotonicity. Consider the case when H is extended by some gene p. If $i \in g(p)$, then $\pi(i,H \cup p) = \pi(i,H)$, otherwise $\pi(i,H \cup p) - \pi(i,H) = 2p(g(i), g(p))m_{ip} \geq 0$

So, the function $F_\pi(X) = \min_{x \in X} \pi(x, X)$ is quasi-concave and we can use the Chain Algorithm to find the orthogonal cluster.

5.2 Analysis and implementation

The performance of the Chain Algorithm depends on the type of data structure one chooses to maintain the set of linkage function values. In Example 10 the total time required for straightforward implementation of the Chain Algorithm is $O(|V|^2)$. Here we build the efficient data structure that enables us to reduce the run-time of the algorithm. There are three operation that are performed at each iteration of the algorithm.

i. find-min - this operation performs in Step 3.1 of the Chain Algorithm where the value $F(\Gamma_i)$ is determined.
ii. delete-min - this operation performs in Step 3.2 of the Chain Algorithm when the Level-Set Algorithm finds set $I_u(A)$ of elements with the minimum value of function π and removes this set from the set A.
iii. decrease-key - this operation performs inside the Level-Set Algorithm. Deleting set $I_u(A)$ entails updating the linkage function values for all neighbors of elements from this set.

If $|V|$ elements are organized into a Fibonacci heap [14], we can perform a delete-min operation in $O(\log V)$ amortized time and a decrease-key operation in $O(1)$ amortized time, while a find-min operation can be done in constant time [8].

Proposition 25 *[33] With a Fibonacci heap, the Chain Algorithm finds an ortholog cluster in time* $O(|E| + |V| \log |V|)$.

Proof. The initialization of the algorithm includes computing $\pi(i, V)$ for each $i \in V$. The value $\pi(i, V)$ depends on the weights on edges incident to i and on the relationship of the

genome $g(i)$ with other genomes. We assume that the number of genomes is very small compared to the number of genes, i.e., $k << n$. Thus computing the initial linkage function values for all the vertices takes $O(|E|)$.
We use Fibonacci heap to store vertices according to their linkage function values. So, the value $F(\Gamma_i)$ can be found in $O(1)$ time, and since each deletemin operation takes $O(\log V)$ amortized time, the total time for all calls to delete-min is $O(V \log V)$.
Each deleting of an element with minimum value of linkage function π leads to updating the linkage function values for all neighbours of the element. Due to the additive property of the linkage function (14), the update can be carried out in time $O(1)$ by subtracting the corresponding value $2p(g(i), g(p))m_{ip}$ due to the deleted edge (i, p).
Decreasing the value of function π involves an implicit decrease-key operation, which can be implemented in $O(1)$ amortized time. As each edge is deleted once, all linkage function updates together require $O(|E|)$ time. Thus, the algorithm runs in $O(|E| + |V| \log |V|)$ time. ■
The proposed ortholog clustering method was applied to the protein sequences from complete genomes of seven eukaryotes present in the eukaryotic orthologous groups [33]. The analysis of these results shows that clusters obtained using proposed method show a high degree of correlation with the manually curated ortholog clusters.

6. Conclusions

In this article, we have investigated monotone linkage functions defined on convex geometries, antimatroids, and semilattices in general. It has been shown that the class of functions defined as minimum values of monotone linkage functions has close relationship with the class of quasi-concave set functions. Quasi-concave functions defined on semilattices, antimatroids and convex geometries determine special substructures of these set families. This structures allow building efficient algorithms that find minimal sets on which values of quasi-concave functions are maximum.
The mutual critical step of these algorithms is how to describe the set closure operator. If an efficient algorithm of the closure construction exists it causes the optimization algorithm to be efficient as well. On the other hand, we think that the closure construction problem is interesting enough to be investigated separately. Thus, we suppose that for an arbitrary semilattice the problem of closure construction has exponential complexity.
An interesting direction for future work is to develop our methods for relational databases, where a polynomial algorithm for closure construction is known [3].
We have considered some applications of quasi-concave functions to clustering a structured data set, where together with pair-wise similarities between objects, we are also given additional information about objects organization.
We focused on a simple structure - a partition model of data where the objects are a priori partitioned into groups. While clustering such data, we also considered an additional requirement of being able to differentiate between pairwise similarities across different partite sets. Existing clustering methods do not solve this problem, since they are limited to finding clusters in a collection of isolated objects.
The requirement of differentially treating pair-wise relationships across different groups was modeled by a multipartite graph along with a hierarchical relationship between these groups. The problem was reduced to finding the cluster (subgraph) of the highest density in the multipartite graph.

This problem is usually formulated as finding the maximum weight multipartite clique. However, no efficient procedure exists for solving this problem. Due to this, clusters are often modeled as quasi-cliques or dense graphs.
Traditionally, quasi-cliques are defined, using a threshold, as a relaxation of a complete subgraph - the relaxation can be on the degree of a vertex or on the total number of edges in the quasi-clique. In contrast to traditional quasi-clique definition, our definition does not use any threshold parameters.
The proposed multipartite graph clustering method was successfully applied to the ortholog clustering problem. It may be also adapted to other clustering problem both in comparative genomics and in computer vision [35].

7. References

[1] Bagotskaya, N.V.; Levit, V.E. & Losev, I.S. (1988). On one generalization of matroids insuring applicability of dynamic programming method, In: *Information Transmission and Processing Systems*, Vol. 2, IPIT USSR Academy of Sciences, Moscow, (33–36). (in Russian)

[2] Bagotskaya, N.V.; Levit, V.E. & Losev, I.S. (1990). A combinatorial structure insuring applicability of dynamic programming method, *Automation and Remote Control* 50, (1414-1420).

[3] Beeri,C. & Bernstein, P.A. (1979). Computational problems related to the design of normal form relational schemes, *ACM Transactions on Database Systems* 4, No.1, (30-59).

[4] Bilbao, J.M. (2003). Cooperative games under augmenting systems, *SIAM Journal of Discrete Mathematics* 17, (122-133).

[5] Björner, A. & Ziegler, G.M. (1992). Introduction to greedoids, In: *Matroid applications*, ed. N. White, Cambridge Univ.Press, Cambridge, UK.

[6] Boyd, E.A. & Faigle, U. (1990). An algorithmic characterization of antimatroids, *Discrete Applied Mathematics* 28, (197-205).

[7] Chernoff, H. (1954). Rational selection of decision functions, *Economica* 22, (422-443).

[8] Cormen, T.H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. (2001). *Introduction to Algorithms*, second ed. MIT Press and McGraw-Hill, (476-497).

[9] Edelman, P.H. & Jamison, R.E. (1985). The theory of convex geometries, *Geom. Dedicata* 19, (247-270).

[10] Edelman, P.H. & Saks, M.E. (1988). Combinatorial representation and convex dimension of convex geometries, *Order* 5, No.1, (23-32).

[11] Edmonds, J. (1971). Matroid and the greedy algorithm, *Mathematical Programming* 1, (127-136).

[12] Eppstein, D. (2008). Upright-Quad Drawing of st-planar learning spaces, *Journal of Graph Algorithms and Applications* 12, No. 1, (51-72).

[13] Fitch, W.M. (1970). Distinguishing homologous from analogous proteins, *Systematic Zoology* 19, (99-113).

[14] Fredman, M.L. & Tarjan, R.E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms, *Journal of the ACM* 34, No.3, (596-615).

[15] Goecke, O. (1988). A greedy algorithm for hereditary set systems and a generalization of the Rado-Edmonds characterization of matroids, *Discrete Applied mathematics* 20, (39-49).

[16] Korte, B.; Lovász, L. & Schrader, R. (1991). *Greedoids*, Springer-Verlag, New York/Berlin.

[17] Kempner, Y. & Levit, V.E. (2003). Correspondence between two antimatroid algorithmic characterizations, *The Electronic Journal of Combinatorics* 10, R44.

[18] Kempner, Y. & Levit, V.E. (2008). Duality between quasi-concave functions and monotone linkage functions, arXiv:0808.3244 [math.CO].

[19] Kempner, Y.; Mirkin, B. & Muchnik, I. (1997). Monotone linkage clustering and quasi-concave functions, *Appl.Math.Lett.* 10, No.4 (19-24).

[20] Kempner, Y. & Muchnik, I. (2003). Clustering on antimatroids and convex geometries, *WSEAS Transactions on Mathematics* 2, Issue 1, (54-59).

[21] Kempner, Y. & Muchnik, I. (2008). Quasi-concave functions on meet-semilattices, *Discrete Applied Mathematics* 156, No. 4, (492-499).

[22] Kuusik, R. & Lind, G. (2004). Generator of Hypotheses - An approach of data mining based on monotone system theory, *International Journal of Computational Intellegence* 1, No. 1, (43-47).

[23] Malishevski, A. (1998). Properties of ordinal set functions, In: *A.Malishevski, Qualitative Models in the Theory of Complex Systems*, Nauka, Moscow, (169-173) (in Russian).

[24] Mirkin, B. & Muchnik, I. (2002). Layered clusters of tightness set functions, *Appl. Math. Lett.* 15, (147-151).

[25] Mirkin, B. & Muchnik, I. (2002). Induced layered clusters, Hereditary Mappings, and Convex Geometry, *Appl. Math. Lett.* 15, (293-298).

[26] Monjardet, B. & Raderanirina, V. (2001). The duality between the antiexchange closure operators and the path independent choice operators on a finite set, *Mathematical Social Sciences* 41, (131-150).

[27] Monjardet, B. (2003). The presence of lattice theory in discrete problems of mathematical social sciences. Why. *Mathematical Social Sciences* 46, (103-144).

[28] Muchnik, I. & Shvartser, L.V. (1989). Kernels of Monotonic Systems on a Semi-lattice of Sets, *Automation and Remote Control* 50, No. 8, part 2, (1095-1102).

[29] Mullat, J. (1976). Extremal subsystems of monotone systems: I, II, *Automation and Remote Control* 37, (758-766); (1286-1294).

[30] Mullat, J. (1995). A fast algorithm for finding matching responses in survey data table, *Mathematical Social Sciences* 30, (195-205).

[31] Sen, A.K. (1971). Choice functions and revealed preference, *Review of Economic Studies* 38, (307-317).

[32] Serganova, V.V.; Bagotskaya, N.V.; Levit, V.E. & Losev, I.S. (1988). Greedoids and the greedy algorithm, In: *Information Transmission and Processing Systems*, Vol. 2, IPIT USSR Academy of Sciences, Moscow, (49-52). (in Russian)

[33] Vashist, A.; Kulikowski, C.A. & Muchnik, I. (2007). Ortholog clustering on a multipartite graph, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4, No. 1 (17-27).

[34] Zaks (Kempner), Y. & Muchnik, I. (1989). Incomplete classifications of a finite set of objects using monotone systems, *Automation and Remote Control* 50, (553-560).

[35] Zhang, R.; Vashist, A.; Muchnik, I.; Kulikowski, C. A. & Metaxas, D. N. (2005). A new combinatorial approach to supervised learning : Application to gait recognition, *LNCS* 3723 (55-69).

26

Semantic Matchmaking Algorithms

Umesh Bellur[1] and Harin Vadodaria
[1]*Department of Computer Science and Engineering, Indian Institute of Technology, Bombay*
[2]*Sybase Software, Pune*
India

1. Introduction

The advantages of loose coupling offered by service oriented architectures (*SOA*) have made it a popular choice for today's enterprise systems. The popularity has driven standardization efforts in the areas of service advertisement and invocation and services specified using these standards are termed as Web services. A Web service is self containing, self describing application that can be deployed, published and invoked over the Internet. The *publish-find-bind* approach is the fundamental idea behind Service Oriented Architectures that web services aim to implement. The ultimate vision of *SOA* is to enable a client to automatically select an appropriate service from a pool of dynamically discovered services and invoke it without having any apriori knowledge about the service provider and the specifics of the service itself. This vision has thrown up various challenges such as - *service discovery based on an abstract query, selection of service from the discovered pool, service composition, dynamic service binding and invocation, quality of service, negotiation of service contracts and trust*.
Enhancing what has traditionally been sytanctic descriptions of services with semantics is necessary to resolve most of these issues. Once semantic descriptions are available, one needs to deal with matchmaking of these descriptions to a query. In the rest of this chapter we present concepts involved in semantic matchmaking as it applied to web services and a set of algorithms that solve the semantic matchmaking problem.

1.1 Background concepts

In this section, we present necessary background concepts essential to understand the rest of the chapter. The chapter centers around Web services although there exist several other implementation of SOA concepts such as Jini for Java.

1.1.1 Service discovery

Service discovery is the process of evaluating a query for a service and returning a set of compatible services. *WSDL* and *UDDI* are two standards used in service discovery. The Simple Object Access Protocol (SOAP) is a messaging protocol used to invoke web services and get back results asynchronously.

- The Web Services Description Language (*WSDL*) [3] is a language for description of service and contains operations supported by the service. Each operation is described

by it's input and outputs. *WSDL* description of a service defines XML message format for communication with the service. A compiler at the client generates stubs based on the WSDL description for: 1) Marshaling and unmarshaling objects into SOAP messages. 2) Sending SOAP messages over communication protocol. The application is then bound with these stubs to invoke the service.

- The Universal Description and Discovery Interface [2] is a registry that contains information about different services offered by various service providers. This information is usually output as a WSDL document. UDDI provides mechanisms for: 1) Publishing a service to the registry 2) Searching a required service from the registry. The state of the art today limits storage in UDDI to Strings and searches are syntactic in nature as well.

1.1.2 Ontology

Ontology represents knowledge about a particular domain. This knowledge includes entities in the domain, their property and relationship with each other. Entities in the ontology are termed *Concepts*. A well defined syntax is required to unambiguously represent concepts of a domain. RDF[16] framework is suitable for describing an ontology. Web Ontology Language (OWL)[17] is developed on the top of RDF and is used for ontology description. Given below is a part of Entertainment ontology.

```
<owl:Class rdf:about="#Concert">
<rdfs:subClassOf rdf:resource="#Theatre"/>
</owl:Class>
<owl:Class rdf:about="#Drama">
<rdfs:subClassOf rdf:resource="#Theatre"/>
</owl:Class>
<rdf:Description rdf:resource="#Concert">
<owl:DisjointWith rdf:resource="#Drama">
</rdf:Description>
```

1.1.3 OWL-S: semantic markup for web services

Semantic web is not merely a collection of marked up content but includes (software applications packaged as) services as well. It is essential for a software agent to discover, compose, invoke and monitor web resources in order to take advantage of a service. OWL-S [1] (formerly, DAML-S) is a language for describing services which makes this possible. It uses RDF as basic framework. OWL-S is required to perform following tasks automatically.

- **Web Service Discovery:** Extract the information from the page in order to find a required service.
- **Web Service Invocation:** OWL-S along with the domain ontology specifies the invocation methods of a Web Service (e.g. necessary inputs, expected outputs).
- **Web Services Composition and Interoperation:** OWL-S provides declarative way to specify prerequisite and consequences of a service which helps software agents in composing different web services.

OWL-S provides Service Profile, Service Model and Service Grounding to represent Description, Functionality and Access Mechanism respectively.

- **Service Profile:** Service profile facilitates Service Provider to describe its service. It is up to the Service Provider how much details are given in the Service Profile. E.g. a Book selling service may also provide browsing facility but it is not necessary that it is included in Service Profile. We can categorize the information provided by Service Profile as:
  - *Provider's Information* - This may include name of the provider and contact details.
  - *Functional Description* - specifies inputs required, output generated and conditions to be set at the beginning and change in the real world after service completes its function. In short, inputs, outputs, preconditions and effects are described here.
  - *Profile Attributes* - Some parameters that service wants to specify e.g. quality guarantees, service categorization etc. They are represented by Service Parameter and Service Category.
- **Service Model:** It describes service as a process, either atomic or composite: receives and sends a single message or retains/changes state through a sequence of messages. A service can give some output and set some condition thus changing real world.
  - Inputs and output parameters are expressed as a subclass of the *parameter* class in OWL-S.
  - Preconditions and effects are modeled as logical formulas or expressions which are treated as either string literals or XML literals depending on the language used. The expression class in OWL-S specifies two separate subclasses *condition* and *effect* for precondition and effect respectively.

Often outputs and effects of the service are coupled together with a condition bounding them. E.g. service for selling software modules may have different results and effects depending on a failed or succeeded transaction.

Composite processes are more difficult to handle. They have a set of sub processes associated with a control structure. The control structure will specify the order in which different sub processes are executed. In case of composite process, client needs to send a series of messages to get the final result. Different types of control structures are: Sequence, Split, Split+Join, Any-Order, Choice, If-Then-Else, Iterate, Repeat While and Repeat Until. Data flow and parameter binding is very critical issue in case of composite process. OWL-S has adopted Consumer-Pull convention i.e. if p2 requires input which comes from p1, p2 is responsible for explicitly describing this fact.

Service Grounding: Grounding deals with the realization of services. It provides concrete details necessary to invoke the service such as message format, transfer protocol etc. OWL-S uses WSDL standard for Service grounding. WSDL provides a wrapper and can carry OWL-S message on standard network protocols. WSDL can not capture the semantic of a message while OWL-S in its own is not capable to deal with the standard transfer protocol. Both languages overlap at description of message at abstract level. Mapping from OWL-S to WSDL is done in 3 steps:

- An OWL-S atomic process corresponds to a WSDL operation.
- Inputs and outputs of OWL-S process correspond to input part and output part of WSDL messages respectively.
- Inputs and output of OWL-S process correspond to WSDL's abstract type.

An example of OWL-S profile is as follows:

```
    <!--- Reference to ontology used in OWL-S --->
<owl:Ontology rdf:about="">
<owl:imports rdf:resource=
"http://www.someurl.org/ontology/hotel.owl" />
</owl:Ontology>

<!--- Description of input parameter in profile --->
<profile:hasInput  rdf:resource="#_CITY"/>

<!--- Connecting the parameter with a concept from ontology --->
<process:Input rdf:ID="_CITY">
<process:parameterType rdf:datatype=
"http://www.w3.org/2001/XMLSchema#anyURI">
http://www.someurl.org/ontology/hotel.owl#City</process:parameterType>
</process:Input>
```

2. What is semantic matchmaking?

The publish-find-bind architecture targets dynamic service invocation - i.e., the client of the service invocation has no prior knowledge of the service description and hence cannot link in pre-compiled stubs. Specification standards such as WSDL and registry standards such as UDDI facilitate the discovery process that is needed for dynamic invocation. Together UDDI and WSDL can serve the goal of automatic discovery of web services. However, the matching mechanism provided by UDDI and WSDL is no better than a simple string matching in XML. In reality automated service discovery can not be accomplished by mere string matching. For example, a simple service that takes two integers as input and produces a float as output could actually perform one of a variety of operations like interest calculation on a principal and period, average points per game given the total points and games etc. A simple syntax based matching can produce many false positives since nature of service is not captured in the service description.

In order to overcome this limitation, concept of semantics has been introduced with OWL-S. In this approach, functionality of a service is described in terms of inputs, outputs, preconditions and effects. Input and output terms of the service are expressed as concepts belonging to a set of ontologies. Use of ontology allows referring to a single concept from two or more syntactically different terms. Thus, it eliminates the limitations caused by syntactic difference between terms since matching is now possible on the basis of concepts of ontologies used to describe input and output terms.

For semantic matchmaking, if we assume that both, advertisement and query are defined in OWL-S format then an advertisement *Advt* and query *Query* match if

- For every input parameter in *Advt*, there is one input parameter in *Query*. Let $Query_{in}$ and $Advt_{in}$ represent the list of input concepts of query and the advertisement respectively. The service can correctly perform the task if all the input concepts defined in the advertisement are satisfied by the requester. Hence, matching of inputs exist if

  $$\forall c \in Advt_{in}, \exists d \in Query_{in},$$
  $$s.t.\ match(c, d) \neq Fail$$

- For every output parameter in *Query*, there is one output parameter in *Advt*. Let $Query_{out}$ and $Advt_{out}$ represent the list of output concepts of query and the advertisement

respectively. The service can be used by the requester if all the output concepts defined in the query are satisfied by the advertisement. Hence, matching of outputs exist if

$$\forall c \in Query_{out}, \exists d \in Advt_{out},$$
$$s.t.\ match(c, d) \neq Fail$$

- For every precondition in *Advt*, there is one precondition in *Query*. Let $Query_{precondition}$ and $Advt_{precondition}$ represent the list of preconditions of query and the advertisement respectively. The service can correctly perform the task if all the preconditions defined in the advertisement are satisfied by the requester. Hence, matching of preconditions exist if

$$\forall c \in Advt_{precondition}, \exists d \in Query_{precondition},$$
$$s.t.\ match(c, d) \neq Fail$$

- For every effect in *Query*, there is one effect in *Advt*. Let $Query_{effect}$ and $Advt_{effect}$ represent the list of effects of query and the advertisement respectively. The service can be used by the requester if all the effects defined in the query are satisfied by the advertisement. Hence, matching of effects exist if

$$\forall c \in Query_{effect}, \exists d \in Advt_{effect},$$
$$s.t.\ match(c, d) \neq Fail$$

2.1 An example

Let us now look at an example of how a request is matched with service advertisements. The service that is advertised is a car selling service which, when given a *Price* as input, return which car can be bought at that price. A strip-down version of advertisement is shown in Figure 2. As, is clear the input to the service are instances of the concept *Price* and the output is instances of the concept *Car*.

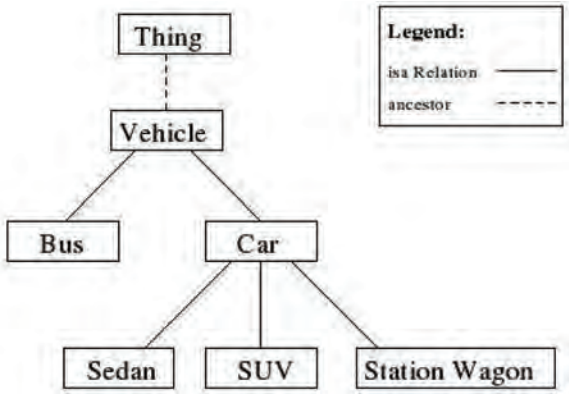


Fig. 1. A fragment of Vehicle Ontology [12]

Shown below is an example request in the same format. The request shows that the service sought should take as input instances of *Price* and should generate output as instances of *Sedan*.

Now, in the given example, for service to match with the request we need to match inputs as well as outputs. In this case, inputs match directly as they both contain the same concept *Price*. The outputs also match as *Car* provided by the service in the given ontology is a super class of the *Sedan* which is expected in the request. Hence, this will be considered as a suitable match although the score(or rank) of this match will vary accordingly with different semantic matchmaking algorithms.

```
<profile:Profile rdf:ID="CarSellingService">
  <profile:serviceName>CarSellingService</profile:serviceName>
  <profile:providedBy> ... </profile:providedBy>
  <input>
    <profile:ParameterDescription rdf:ID="Price_Input">
      <profile:parameterName>Price</profile:parameterName>
      <profile:restrictedTo rdf:resource="Concets.daml#Price"\>
    </profile:ParameterDescription>
  </input>
  <output>
    <profile:ParameterDescription rdf:ID="Car_Output">
      <profile:parameterName>Car</profile:parameterName>
      <profile:restrictedTo rdf:resource="Vehicle.daml#Car"\>
    </profile:ParameterDescription>
  </output>
</profile:Profile>
```

Fig. 2. Advertisement of a car selling service [12]

```
<profile:Profile rdf:ID="RequestSedanSellingService">
  <input>
    <profile:ParameterDescription rdf:ID="Price_Input">
      <profile:parameterName>Price</profile:parameterName>
      <profile:restrictedTo rdf:resource="Concets.daml#Price"/>
    </profile:ParameterDescription>
  </input>
  <output>
    <profile:ParameterDescription rdf:ID="Sedan_Output">
      <profile:parameterName>Sedan</profile:parameterName>
      <profile:restrictedTo rdf:resource="file:data/Vehcle.daml#Sedan"/>
    </profile:ParameterDescription>
  </output>
```

Fig. 3. Request for a car selling service service [12]

If we had an advertisement with *Sedan* as a concept, it must be ranked higher than the above advertisement as it is closer to the given request. From this example, it is clear, the match performed is a semantic match. The reason is because the fact that *Car* is a superclass of *Sedan* has been used while matching. In a syntactic matching scenario, this would result in no match as *Car* and *Sedan* are syntactically very different.

Note that in given example service semantics are described by input and output parameters only. In addition to these parameters, preconditions and effects can also be added to define restriction over parameter values.

3. Taxonomy of semantic matchmaking algorithms

In this section, we present the qualitative and quantitative aspects on which a semantic matchmaking algorithm can be evaluated. We then use this to compare and contrast different efforts in the area.

3.1 Qualitative aspects

Semanticmatching as compared to syntacticmatching As the term *semantic matchmaking* suggests, a semantic matchmaking algorithm should consider the meaning of concepts while performing comparisons between services and requests. It should take into account the various relations which exist between the concepts in the ontology in the process of matchmaking.

For example, in the ontology given by 1, when a request contains *Sedan*, a service with concept *Car* should be given more weightage than concept *Vehicle* as *Sedan* is closer to *Car* in the ontology. Similarly, a service with *Sedan* should be given an appropriate score(less than *Car*) when a request for *Car* is made by taking into account the fact that *Sedan* could only partially satisfy the request. Its important to note that in pure syntactic matching, this kind of reasoning is not possible as the meaning of the concepts are not considered.

False positives and negatives False positives are returned when a semantic matchmaking algorithm matches an advertisement to a given request even if it was not relevant. Analogically, a false negative is the case where a semantic matchmaking algorithm fails to match a relevant advertisement to the given request. There is a trade off between the number of false positives and false negatives returned by a matchmaking algorithm. As the algorithm becomes more flexible, the number of false positives increase and number of false negatives decrease. Therefore, its necessary to regulate the flexibility of the semantic matchmaking algorithm so as to have a balanced number of false positives and negatives. The requesting service should have some control over the flexibility of the algorithm.

Notion of Flexible matching The semantic matchmaking algorithm should promote the advertisers to be more precise in their description. It can be done by providing a degree of match for the matched advertisements. The degree of match should be higher for advertisements which are closer to the request and hence imposing penalty on advertisements which are very general. If this is not done, then all the advertisers will make advertisements as general as possible to increase their chances of match rather being specific about what they actually have.

Consider that AdOp is one of the concepts of the outputs of an advertisement Qop is one of the concepts of the outputs of a query. Four degrees of matching are:

- **Exact:** If AdOp is *an equivalent concept* to QOp, then they are labeled as *Exact match.*
- **Plug in:** If QOp is *superclass of/subsumes* AdOp, then AdOp can be plugged in place of QOp. Thus, it is marked as *Plug in match.*
- **Subsumes:** If AdOp is *superclass of/subsumes* QOp, then service may fulfill the requirements of the request since advertisements provides output in some of the subclasses of the concept defined by QOp. Thus, it is a *subsume match.*

- **Fail:** If no subsumption relation is found between QOp and AdOp, then it is declared as failure.

Soft constraints Soft constraints are constraints which should be preferably but not necessarily be satisfied. For example, if we are ordering a DVD from a web-based DVD store, we could specify that we would prefer to pay by a credit card. A semantic matchmaking algorithm should be able to take soft constraints into account while performing matches. Hence, an advertisement which satisfies a soft constraint should be given better ranking than an advertisement which does not (assuming they satisfy other constraints with same degree of match).

Preference of concepts The user should be able to specify which concepts are preferred. A semantic matchmaking algorithm should to take into account the preference of concepts as specified by the user. For example, if a user needs to book a hotel for his journey, then most important concepts for him would be the *city* and *date*. Other concepts(for e.g. prices) even if matched would be useless unless it is in the same city as mentioned by the user. Hence its important that the algorithm gives higher ranking to advertisements which match concepts of higher preferences.

User defined matching The user should have some control over the matching process. The algorithm should give the user ability to regulate various aspects such as the flexibility of the algorithm, the *quality* or *rating* which is expected for the service etc. User-defined matching helps make the matching process more suited to one's needs.

Heterogeneous ontologies The semantic matchmaking example, we discussed in previous section assumes that both the service and the request use the same shared ontology for description. However, in a truly distributed environment where services are autonomous this assumption may not hold true. Hence, an algorithm must be able to perform semantic matching across descriptions with heterogeneous ontologies.

Quality-of-Service(QoS) enabled discovery All the aspects we have discussed so far deal with how closely the advertisement matches functionally with the request. However, non-functional and QoS properties such as price, performance, throughput, reliability, availability, trust etc. are equally important while deciding whether a service is satisfactory for a given request. Hence, a semantic matchmaking algorithm must take into account the presence of such parameters while matching. User feedback must be taken into account in this framework to define QoS properties for various services.

3.2 Quantitative measures

Efficiency : As we know that the search has to be made over all possible advertisements for services. Given the current size of Web, the number of services existing on web and hence the number of advertisements will be huge. Hence, for the semantic matchmaking process to scale up to the size of web, the computational complexity of the algorithm should not be high.

Precision : Precision is defined as the number of "relevant and retrieved" advertisements over the number of "retrieved" advertisements. As the algorithm becomes more flexible in matching, the number of false positives increase and hence precision decreases. Therefore,

AdvtInput	Date,City
AdvtOutput	Theatre,Music

Table 1. Advertisement

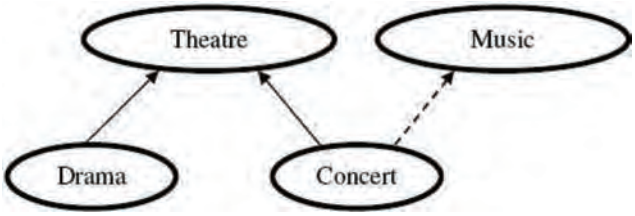


Fig. 4. A part of Entertainment Ontology

QueryInput	Date,City
QueryOutput	Concert,Drama

Table 2. Query

to obtain higher precision the semantic matchmaking algorithm has to be more rigid in matchmaking.

Recall Recall is defined as the number of "relevant and retrieved" advertisements over the number of "relevant" advertisements. As the algorithm becomes more flexible in matching, the number of false negatives decrease and hence recall increases. Therefore, to obtain higher recall the semantic matchmaking algorithm has to be more flexible in matchmaking.

F1 and break-even : As discussed earlier, we need to have regulated amount of flexibility in the algorithm to balance precision and recall. Since there is a trade-off between precision and recall, we can use unified measures which will give weightage to both precision and recall. For example, F1 is defined as the harmonic mean of precision and recall. Hence, when maximized, it would result in both precision and recall set to acceptable values. Similarly, break-even is the point where the precision and recall curves meet each other.

Precision and recall are very coarse-grained measures as they categorise a document into two categories :- *relevant* or *irrelevant*. Especially, in the context of semantic matchmaking where the degree of match between query and advertisement comprises of many levels, such coarse-grained measures are not the best indicators of performance of matchmaking.

We need fine-grained evaluative measures which can distinguish between documents matching with various degrees of match. [16] proposes a method based on fuzzy logic which provides fuzzy equivalents of precision and recall as measures to quantify performance of matchmaking.

These equivalents are computed in terms of two membership functions, one defined by the semantic matchmaking engine and one by the domain experts. The two membership functions are $fe : Q \times S \rightarrow [0, 1]$, and $fr : Q \times S \rightarrow [0, 1]$. fe is delivered by the algorithm and fr is calculated by the feedback of domain experts. These functions are computed using fuzzification of the degree of match performed between the advertisement and the request. The fuzzy logic equivalents of Recall(R_G) and Precision(P_G) are defined in Equations 1 and 2.

$$R_G = \frac{\sum_{S_i \epsilon S} min\{fr(R, S_i), fe(R, S_i)\}}{\sum_{S_i \epsilon S} fr(R, S_i)} \tag{1}$$

$$P_G = \frac{\sum_{S_i \epsilon S} min\{fr(R, S_i), fe(R, S_i)\}}{\sum_{S_i \epsilon S} fe(R, S_i)} \tag{2}$$

Since, theses measures are fuzzy, they take into account the values for all the advertisements and not only those documents which are relevant or returned.

4. A survey of matchmaking algorithms

In this section we take a in-depth look into existing semantic matchmaking algorithms.

4.1 Greedy approach

This algorithm was proposed by [12]. It is based on semantic matchmaking based on input and output terms.

Algorithm presented in [12] is a greedy approach for matchmaking. Algorithm tries to match every output concept of Query with one of the concepts of Advertisement. It starts from all output concepts (call it candidate list) of Query and removes a concept from candidate list as soon as it is matched with a concept from Advertisement with degree of matching > Fail.

[12] uses following scheme for degrees of matching.

- **Exact:** If AdOp is an equivalent concept to QOp, then they are labeled as Exact match. If QOp is a subclass of AdOp, then match is considered Exact under the assumption that provider agrees to provide output in every possible subclasses of AdOp.
- **Plug in:** If AdOp subsumes QOp, then AdOp can be plugged in place of QOp. Here also, it is assumed that provider agrees to provide output in some of the subclasses of AdOp.
- **Subsumes:** If QOp subsumes AdOp, then service may fulfill the requirements of the request since advertisements provides output in some of the subclasses of the concept defined by QOp.
- **Fail:** If no subsumption relation is found between QOp and AdOp, then it is declared as failure.

4.1.1 Discussion

Scheme for Degrees of matching assumes that service provider agrees to provide output in every possible subclass of the output concept. Also, Algorithm is dependent on the order in which concepts are defined in the Query. Consider following example.

As depicted in figure 4, Drama and Concert are subclass of the concept Theatre. Concert is also a sub-concept of Music via inferred relationship.

Output concept list for *Query* and *Advt* are:

- $Advt_{output}$ = Theatre, Music
- $Query_{output}$ = Concert, Drama

At first, algorithm will try to match Concert with all concepts of candidate list of $Advt_{output}$. We have,

- Theatre *is superclass of* Concert ⇒ Exact match
- Music *subsumes* Concert ⇒Plugin match

Since the algorithm uses greedy approach, it will match Concert with Theatre and removes both from respective lists. Now there is only one concept in $Advt_{output}$ and,

Match(Drama, Music) = Fail

Thus, the algorithm will return *Fail* match for *Query* and *Advt*. In reality, we can have following matching.

- Theatre *is super class of* Drama→ Exact match
- ″ Music *subsumes* Concert → *Plugin* match

Overall degree of matching for *Query* and *Advt* is *Plugin*. If we have changed order of concepts in *Query* $_{outputlist}$, we could have achieved this matching. Thus, algorithm in [12] is dependent on the order in which concepts are defined. Thus, algorithm may produce false negative results.

Consider the scenario when the output concept is not removed from the candidate list. Suppose, the advertisement is for {*Theatre*, *Cost*} and the request is for {*Drama*, *Concert*}. Here, the above algorithm would return an exact match as both *Drama* and *Concert* are immediate subclasses of *Theatre*. Hence, the requester would receive only one reservation for a *Theatre* whereas he expected two reservations for a *Drama* and a *Concert*. This would result in a false positive.

4.1.2 Quantitative analysis

For each advertisement, we have to compare each output concept of query with all the advertisement concepts and each input concept of advertisement to all the input concepts of query. Hence, the number of operations are given by

$$C = \{(Qo \times Ao) + (Qi \times Ai)\} \tag{3}$$

where Qo and Ao are the number of output concepts and Qi and Ai are the number of input concepts in query and advertisement respectively. Since, the algorithm iterates over N advertisements, the total complexity is given by

$$C = N \times \{(Qo \times Ao) + (Qi \times Ai)\} \tag{4}$$

in general, Qo,Ao,Qi and Ai are bounded by small integers. Hence, the complexity is linear in N (the number of advertisements) with small constants.

$$Complexity \approx O(N) \tag{5}$$

4.2 Bipartite graph based matching

To solve the problems mentioned in previous section, we introduce in this section another approach [5] towards semantic matchmaking which makes use of bipartite graph matching to produce a match.

The algorithm also introduces a different set of rules of match between concepts in which **PlugIn** and **Subsume** levels are interchanged in their degree of match. The assumption that if an advertiser advertises a concept, it would provide all the immediate subtypes of that concept is dropped. Hence, if the query concept is subsumed by the advertisement concept a **Subsume** is returned while if the query concept subsumes the advertisement concept **PlugIn** is returned. **PlugIn** is still ranked higher than a **Subsume** match. You can see that this scheme of matching is opposite to the one discussed in previous section.

Bipartite graph A bipartite graph is a graph in which the vertex set can be divided into two disjoint sets such that no edge of the graph lies between the two vertices in the same set.

Matching A matching of a bipartite graph G = (V,E) is a subgraph G′ = (V,E′) such that no two edges e1,e2 in E′ share the same vertex.

Let the set of output concepts for query and advertisement be Q and A. We will construct a graph G = (Q+A,E) which has one vertex corresponding to each concept in query and advertisement . If there exists a degree of match between (≠ Fail) between a concept $v1$ belonging to Q and a concept $v2$ belonging to A, then we define an edge ($v1$, $v2$) with weight as the degree of match. We need a matching in which all the output concepts of Q are matched with some concept of A. If such a matching exists, we would say that the advertisement and the query match.If there exist multiple such matchings, we will choose the one which is optimal(the criterion is defined below). However, if such a matching doesn't exist the query and the advertisement doesn't match.

optimality criterion We need to select the matching which is best from the perspective of semantic match. For, this we would assign different numerical weights to edges with different degrees of match. Let us suppose, we assign minimum weight to **exact**, then **Plugin** and then **subsumes**.Let $max(wi)$ be the maximum weighted edge in the matching. An optimal matching in this case would be a complete matching with minimum $max(wi)$.

Algorithm for optimal matching Hungarian algorithm [10] computes a complete matching for a weighted bipartite graph such that sum of weights of all the edges in the matching is minimised. To adapt Hungarian algorithm to above case, where a matching with minimum value of $max(wi)$ is needed, we would assign weights according to the scheme shown as below.

Degree of Match : match(a,b)	**Weight**
Exact	w1 = 1
Plugin	$w2 = (w1 * \mid V0 \mid) + 1$
Subsume	$w3 = (w2 * \mid V0 \mid) + 1$

It can be proved that with the above weighting scheme, a matching in which Σwi is minimized is equivalent to matching in which $max(wi)$ is minimised [5].

Matchmaking Algorithm The search procedure accepts a query as input and tries to match its output concepts and input concepts with each advertisement. If there exists a match in both input and output concepts, it appends the advertisement to the result set. To match inputs as well as it outputs, it invokes Hungarian algorithm on a graph created with weights as given in above table to compute an optimal matching of the graph. The degree of match is defined by the weight of the maximum-weight edge in the matching. In the end, a list of advertisements sorted on the basis of input and output concepts is returned.

4.2.1 Discussion

The above algorithm eliminates the correctness issues with the algorithm described in the previous section. It also regulates false positives and false negatives as discussed in the example above. However, it does not allow for priority of concepts and soft constraints to be input by the user. Like the previous algorithm, the algorithm does not provide a ranking of the results. The algorithm assumes a shared ontology between the advertisements and the request. In the following sections, we would look at some algorithms which allow some of these features.

4.2.2 Quantitative analysis

Using the same notation as in Section 4.1.2, we get

The weights *wo, w*1 and *w*2 are computed in *O*(1) time. The weights of edges in the graph can be determined in *Qo*×*Ao* operations, by comparing all pair of concepts.
The time complexity of Hungarian algorithm is bounded by Qo^3. Hence, the total complexity of the search is bounded by:

$$C = N \times \{(Qo \times Ao) + Qo^3 + (Qi \times Ai) + Ai^3\} \quad (6)$$

Hence, we get If we assume that number of input and output concepts in the query and advertisement are small, we can approximate:

$$Complexity \approx O(N) \quad (7)$$

The complexity of above algorithm is asymptotically similar to the previous algorithm. However, the constants will be different.

4.2.3 Addition of precondition and effect matching

Original algorithm proposed by [5] was based on matching of input and output terms only. However, precondition and effect matching can also be added using same bipartite graph based technique as discussed in [6]. As discussed earlier, in OWL-S description, preconditions and effects are represented as boolean expression. Algorithm for condition matching works in two phases.

- **Parameter Compatibility:** Whether parameters used in both expressions are equivalent or not. From input-output terms matching, we obtain the mapping between terms used in query and advertisement. If every parameter used in the query's condition has an equivalent parameter (obtained from the mapping constructed during input-output term matching phase) in the advertisement's condition such that, degree of matching between two parameters > Fail Match, we have parameter compatibility between these two conditions.
- **Condition Equivalence:** This refers to structural similarity between two conditions. For our purpose, we do not need strict equivalence. If condition specified in the query contains all parameters specified in advertisement's condition AND the relation between various parameters in advertisement's condition are retained in query condition, we can flag it as condition equivalence. In other words, if condition in the query is denoted by QCondition and condition in advertisement is denoted by ACondition then what we need is,
  $QCondition \Rightarrow Acondition$
  which essentially says that, variable space in which QCondition is true is a subset of the variable space in which ACondition is true.
  This is true when we match for preconditions. The relation will be reversed when we match for effects. i.e.
  $ACondition \Rightarrow Qcondition$
  This problem is constraint satisfiability problem which NP-Complete by its nature. Some heuristics like DPLL algorithms are used to solve this problem in exponential time.

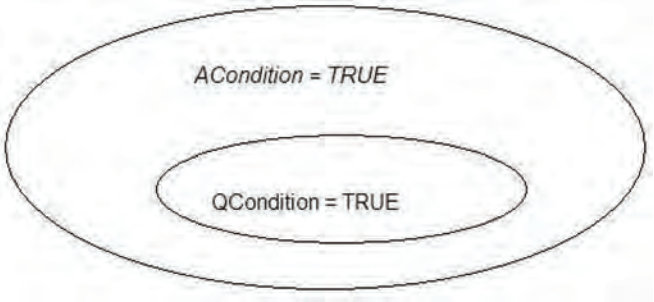


Fig. 5. Solution space for *ACondition* and *Qcondition*

4.3 Semantic matchmaking across heterogeneous ontologies

In this section, we discuss a framework for semantic matchmaking which relaxes the requirement of a single ontology and allows advertisements and requests to be expresses in different ontologies. The approach to compare concepts across ontologies uses different ways to assess the similarity of various concepts used in description of services and requests [14].

SynonymSets Synonymsets are semantically equivalent or very similar words. Hence, synonyms can be considered as the same entity. Wordnets are used to derive the synonym set of the name of the parameters. In a cross-ontology evaluation scenario these words(like person and human) are likely to refer to the same entity.

Distinguishing features of concepts Some concepts could have quite different names, while still being semantically similar. To incorporate semantics into the similarity measure in such cases we can also use some distinguishing features of concepts. We choose the properties of classes such as object properties and data type properties to perform semantic similarity tests. The assumption is that semantically similar parameters with different names are likely to have some common features or properties. The matching is performed between the properties of the two concepts.

Semantic neighbourhoods and relations The semantic relations which exist between various classes could be used to perform semantic matchmaking. The idea is that the target concepts (i.e.which are subject of comparison) which are related to the same set of classes through similar relations, may be semantically similar. For example, semantic relations like Subclass, Disjoint With,Equivalent Classes etc. can help determine semantic similarity amongst various concepts.

To integrate the information obtained by above methods, a weighted sum of the similarity of each function component is used to compute the overall similarity.

Another similarity measure defined in terms of set theory is based on the normalisation of Tversky's model and the set-theory functions of intersection($A \cap B$) and difference (A/B). It is shown below.

$$S(a,b) = \frac{A \cap B}{A \cap B + \alpha(a,b)A/B + (1-\alpha(a,b))B/A}, for 0 \leq \alpha \leq 1 \tag{8}$$

where a and b are parameter classes

A and B corresponds to the description sets of a and b

(i.e. synonym sets,feature sets and semantic neighbourhood)

and α is a function which defines relative importance of non-common characteristics

The above mentioned similarity measures are used to compute the edge weights of edges in the bipartite graph discussed in the previous section. The function elements(concepts) are extracted from the advertisement as well as the requested profile. A bipartite graph is formed using these concepts as nodes. The edge weights are then computed using the similarity measures described above. Bipartite graph matching algorithms are then applied to produce matches and their scores which are used to generate the sorted list of relevant advertisements.

4.3.1 Discussion

The algorithm provides a way to make semantic matchmaking possible over descriptions with heterogeneous ontologies. The algorithm uses a similarity function of concepts for matching which is based on a weighted sum of synonym sets,semantic neighbourhood and distinguishing features. The algorithm however does not allow the user to input preferences of concepts. Preference of concepts would give the requester more expressive power to express their needs. In further sections, we would look into some algorithms which support preferences amongst the concepts.

4.4 Semantic matchmaking based on description-logics

We now discuss an algorithm which performs semantic matchmaking on advertisements and requests which are defined in Description Logics. Description Logics(DL) are a family of logic formalisms used for knowledge representation. They can be used to represent the knowledge of a service or an application domain in a structured and formal way which can be understood by a computer system. As we will see the algorithm discussed below provides a ranking of the matched advertisements which was not the case with the previous algorithms.

In this section, we'll discuss an algorithm for the DL of the Knowledge Representation System CLASSIC of AT&T Bell Labs [15]. The basic syntax is based on predicate logic and comprises of three kinds of descriptions.

- **concept names** concept names stand for sets of objects, such as book, room etc.
- **number restrictions** these correspond to restrictions which quantify the amount of a concept. for example, ($> 3 author$) denotes that there should be more than three authors.
- **universal quantifications** these can be used to specify some restriction on all the objects belonging to a concept. For example, $\forall$ *supplier.japanese* implies that all the suppliers (i.e.objects belonging to the concept supplier) must be Japanese.

An advertisement (as well as a request) can be described as a conjunction of these concepts. For example, one might represent an advertisement for an apartment as

$$A = apartment \cap \forall hasRooms.roomswithTV \cap (\geq 3\ hasRooms)$$

Requests can be represented in the similar way. The matchmaking algorithm then matches the request with the candidate advertisements one by one and provides a ranking for the match.The algorithm, recursively calls itself for the parts which are universally quantified and keeps a global score which denotes the degree of match. If there happens to be a case, in which there exists a universal quantification statement for a particular concept in only one of the advertisement or request, the recursive call is made with a T (universal truth) as predicate. Hence, if a description does not mention $\forall hasRooms$, we would assume that $\forall hasRooms.T$ is present in the description. The algorithm is as follows.

4.4.1 Algorithm

The algorithm for ranking follows a recursive procedure as mentioned above. It starts with a global rank of zero for every advertisement and then increases it for every concept which differs in the advertisement and the request. Therefore, lower the rank, higher is the degree of match. The algorithm uses four rules to increase the rank which are given below :-

- Add 1 to rank for each concept name which are present in query but not in advertisement
- Add 1 to rank for each number restriction in request which can not be satisfied by the advertisement
- If for a concept $\forall R.E$, there does not exist a $\forall R.F$ in the advertisement, add to rank the answer to the recursive call with T, and E.
- If for a concept $\forall R.E$, there does exist a $\forall R.F$ in the advertisement, add to rank the answer to the recursive call with F, and E.

Total match exists when the algorithm returns 0. The above algorithm can be modified easily to provide for preference of concepts. By adding different weights for different concepts, we can penalize the match selectively according to our preferences. Thus, an important concept would cause a larger number to be added to n, hence decreasing the degree of match for advertisements which can not satisfy them. In the next section, we will see how preference of concepts can increase your expressive power in defining the query. The taxonomy can be also taken into account, while defining weights for various concepts. Hence, in the taxonomy of figure 2.1, we could say that n will be increased a larger amount if we have vehicle and SUV as compared to if we have vehicle and cars. The weights can also be learnt by the system, by providing a set of advertisements and their ranks according to human users. Hence, the system would be able to learn to distinguish between concepts which are more important by learning weights to fit given training examples of ranked advertisements.

4.4.2 Discussion

In this section we saw an algorithm which performs semantic matchmaking on advertisements and requests described using Description Logics. As we discussed, the algorithm performs an approximate matching of advertisements and requests and provides a ranking of candidate advertisements with varying degrees of match. The algorithm can also be used to take into account preference of concepts as provided by the user.

4.5 Semantic matchmaking based on ranked instance retrieval

In this section we present another method for semantic matchmaking which takes into account the preference of concepts a provided by the user [4].This algorithm uses the concept of a ranking tree to match and compare various advertisements w.r.t. a particular query.

We will take an example to describe how the preference of concepts gives you more expressive power in making the request. Suppose, the user wants to find out a service which offers DVD's for movies. Hence she could make a query like, Q1 := $OffersDVD$. However, this would provide tons of hits. To narrow down our search, she would want to provide more search criterion. Suppose she specifies that she prefers 24 hours shipping over three-day shipping and a service with shipping time more than three days is not acceptable.

In this case, writing a query like Q2 := $OffersDVD \sqcap (24HoursShipping \sqcap 3DaysShipping)$, would get unacceptable results as the proper requirement has not been expressed. To

express the correct requirement, there should be a way to annotate the concepts with preferences, thus providing a way to determine which concept is preferred. Hence, if we provide a query like Q3 :=$OffersDVD^1 \sqcap (24HoursShipping^2 \sqcap 3DaysShipping^1)$, the service with 24HoursShipping would be rated more than 3DaysShipping and hence would generate acceptable results.

The above method of annotating preferences, could also be used to specify soft constraints as discussed earlier. Suppose, we have a top concept T, such that every concept is an instance of type T. Now, suppose if we need to specify that we would prefer to have a service with CreditCardPayment, however its not a necessity, we could do that by writing the query as

Q4 :=$OffersDV\ D^1 \sqcap (24HoursShipping^2 \sqcap 3DaysShipping^1) \sqcap (CreditCardPayment^1 \sqcup\ T^0)$.

Similarly, we could use a bottom concept $\perp$ to denote the fact that OffersDVD is a necessary concept but should not affect the ranking.Suppose we write our query as,

Q5 :=$OffersDV\ D^1 \sqcup (24HoursShipping^2 \sqcup 3DaysShipping^1) \sqcup (CreditCardPayment^1\ \sqcup\ \perp^0)$.

In the above query, the second part of disjunction is not satisfiable and hence every hit must satisfy OffersDVD. Hence, ranking is only affected by the other concepts which could be taken into account by the matchmaking algorithm. Therefore, as we discussed allowing annotations of concepts with their preferences could give us a lot more expressive power in describing our request. It will also allow the user to specify soft constraints and constraints like in Q5.

Due to existence of disjunctive knowledge in description logics (in query Q5), a single numerical value is not sufficient for expressing rankings. Suppose, we have Q := $A^1 \sqcup (B^1 \sqcup C^2)^0$. Since, B ⊔ C has preference 0, it should not contribute to the top level rank. However, for two equal top level ranks, we should use $(B^1 \sqcup C^2)^0$ to refine the ranking. A ranking tree is appropriate for such kind of reasoning.In the following part of this section, we will discuss a ranking tree and how it can be used for matchmaking of advertisements and requests with such annotations.

4.5.1 Ranking tree

We define a ranking tree as follows:

1. for r ∈ [0 ; 1] (r) is a ranking tree.
2. let r ∈ [0 ; 1] and t1,t2 .. tn be ranking trees with n ≥ 1, then (r,t1,t2,...,tn) is a ranking tree.

for example t1:= (0,(1), (0,(1),(0))) is ranking tree.

Ordering on ranking tree Let a = (r_a, $a1$, $a2$, $a3$....an) and b = (r_b, $b1$, $b2$, $b3$.....bn) where a1,a2...an and b1,b2 ... bn are ranking trees.

Let a < b⇔$r_a < r_b$ now, a ◁ b iff

1. a < b or
2. $r_a = r_b$ and $\exists i : a_i < b_i$ and $\forall 1 \leq j \leq n : b_j \not< a_j$ or
3. $r_a = r_b$ and $\forall 1 \leq i \leq n : a_i \trianglelefteq b_i$

4.5.2 Matchmaking algorithm

Given an annotated query and an advertisement, we must evaluate the ranking tree of the advertisement. The query is represented as either a conjunction or disjunction of subqueries.

The ranking tree is evaluated by calling the routine recursively for each subquery and using the resulting ranking trees to form the top-level ranking tree. The rank of top-level rank tree is an of average user preferences of all the subqueries which are satisfied. For a negated query, the rank of top-level tree is replaced by the average of user preferences of all the concepts which are not satisfied by the query. An atomic concept belonging to the query is satisfied by advertisement if its contained in it.

Discussion The above algorithm supports most of features discussed in chapter 2.The algorithm supports preference of concepts and allows for soft constraints to be specified. In the next section, we would look at Quality-of-Service(QoS) aspects of semantic matchmaking.

4.6 QoS enabled matchmaking

A web service is a web-based interface which providing electronic description of a concrete service. The service could be of varied types from functionality of a software component (such as data backup) to a real-life business activity(such as shipping). Hence, the QoS properties of a service vary over a wide range depending upon the type of service. For example, for a network service, it could be response-time,availability etc. On the other hand, for a pizza delivery service it could the quality of food. QoS is a very important factor in deciding whether the service will be able to fulfil the demands of requester. Hence, it is very important for a semantic matchmaking algorithm to take into account QoS parameters along with the functional properties to perform a more meaningful and successful match.

To support QoS information while discovering services through matchmaking process, we need to evaluate how well a service can fulfil user's non-functional (quality) requirements based on the service's past performance. Hence, there must be an interface where the users can submit their feedback on the perceived QoS of consumed services. While discovering services, we can take into account data from the following sources [11]:-

- QoS values promised by the providers in their advertisements
- feedback on the perceived QoS submitted by the users on the interface
- reports produced by trustworthy QoS monitoring agents such as rating agencies
- QoS information provided by similar discovery components which exist over the network in a distributed setting

A complementary ontology for providing detailed QoS information have been proposed in (Semantics in service discovery and management.) Algorithms similar to what we have studied in this chapter so far, can be applied to perform matching on QoS parameters. Hence, a service would be matched on its functional properties as well as non-functional properties and QoS parameters and the information provided by both the aspects would be combined to provide a common ranking of advertisements which could be used by a requester.

5. Comparing the algorithms

In this section, we provide, a comparison table comparing the algorithms on various aspects we have presented so far.

Feature	Greedy Algo.	Bipartite Matching	Heterogeneous Ontologies	DL Algo	Ranked Instance Retrieval
IO based Matching	Yes	Yes	Yes	Yes	Yes
PE based Matching	No	Yes	No	No	No
Correctness issues	Yes	No	No	No	No
Ranking of hits	No	Yes	Yes	Yes	Yes
Soft constraints	No	No	No	No	Yes
User preferences	No	No	No	Yes(in extended version)	Yes
Heterogeneous ontologies	No	No	Yes	No	No
Computational complexity	O(N)	Within polynomial limits	High	O(N)	High

Table 3. Comparison table for different matchmaking algorithms

6. Applications of semantic matchmaking

Semantic matchmaking has been applied in a variety of contexts. It is a very importan field of research and forms a basis for service discovery and composition. Reliable and efficient algorithms for semantic matchmaking are extremely important in the new vision of web(semantic web).

The algorithm we discussed here have been used to make a combined matchmaker for DAML-S/OWL-S and UDDI. UDDI allows only keyword search based on the names, which is not enough as no inferencing or flexible matching can be performed. A matching engine which augments UDDI registries with an additional semantic layer allows for a capability based matching. It uses the ontologies published on the web. The matchmaking process allows services to be discovered on the basis of their capabilities and hence result in their interoperability and enhanced problem solving abilities with minimizing human intervention.

Preference SQL [9],a powerful extension to SQL aims for providing support for preference queries in relational databases. Its objective is similar to that of matchmaking. It aims to find approximate matches to user's queries which are based on preferences defined by him. It also performs matchmaking to find the best possible match between a request and existing data.

OWLS-MX is a semantic web services matchmaker which retrieves services for a given query. It uses both logic based semantic matching and token-based syntactic similarity measure to perform the match between a query and the services [13].

[9] discusses the need for semantic matchmaking in geo web services which are in growing demand these days. Geo Web services are services which provide location based information on the web. For example, an user who wants to know about hazardous objects

near her proximity would need to first find out her own location by a geo coder service. Then she could use another service to locate the all the chemical factories etc near it. With a semantic matchmaker, such services could be discovered and interoperate with each other to form more complex services.
[7] use semantic matchmaking approach for skill management of various business entities. Semantic matching is performed between buyers and sellers of skills. It is very important for knowledge intensive companies because it can be used to search for professionals who have expertise in a given area within and across companies.
These are but a few applications of matchmaking engines and algorithms that we have presented in this chapter.

7. Open issues

We currently do not have well recognized evaluation metrics for the efficiency of algorithms which would define their scalability in real world scenarios. We also need testcases and testbeds where various algorithms can be plugged in and tested against each other for various features described in previous sections.
Currently fuctional information used in semantic matchmaking is limited to IOPE. This can be extended in the following ways.

- Semantic matchmaking can involve use of contexts. Currently there is no framework defined for context identification and evaluation. In fact, context providers can be thought of web services Web service providing output in terms of contextual information. [6] has proposed an architecture for such a context aware discovery mechanism but still, inclusion of contexts needs major change in current mechanism for semantic matchmaking.
- Preconditions and effects are represented as boolean conditions and matching based on them is limited to structural similarity of expressions. However, if we can treat static and dynamic nature of parameters [6], evaluation of expression can be partially done at discovery time.

McIlraith suggested an approach for matching based on non-functional requirements of web service. We still need an improved framework to specify non-functional requirements more clearly. Also, weights of the results obtained from matching based on functional and non-functional requirements has to be set in such a way that candidate services are ranked as close as uesr's preference.
We presented an algorithm that tries to perform semantic matchmaking across heterogeneous ontologies, still the discrepancies amongst the ontologies could lead to many failed matches. We need a sophisticated mediation layer in the system which would help in translation of natural language requests to ontology based requests. We also need to find better algorithms and framework for dealing with non-functional properties of services such as trust and reliability. Such properties are very important for building semantic matchmaking systems which can be used reliably by users.

8. Concluding remarks

We have seen a variety of algorithms dealing with different aspects of semantic matchmaking. They involve matchmaking based on functional and non-functional requirements of web service. The process of semantic matchmaking assumes that semantic

information is attached to services. But, the question we need to ask is: Is this expressiveness worth the complexity of semantic matchmaking? In other words, is it possible for a semantic matchmaking system to deliver performance comparable to syntactic matching systems(like keyword search)?

9. Acknowledgments

The authors would like to acknowledge the contribution made by Amit Gupta of the Department of CSE, IIT Bombay.

10. References

[1] *OWL-S: Semantic Markup for Web Services*. http://www.w3.org/Submission/OWL-S/. Visited on 10th June,2008.

[2] *Universal Description Discovery and Integration (UDDI)*. Visited on 10th June, 2008.

[3] *Web Services Description Language (WSDL)*. http://www.w3.org/TR/wsdl. Visited on 10th June,2008.

[4] Matthias Beck and Burkhard Freitag. Semantic matchmaking using ranked instance retrieval. *SMR '06: Proceedings of the 1st International Workshop on Semantic Matchmaking and Resource Retrieval, Co-located with VLDB*, 178 of CEUR Workshop Proceedings, 2006.

[5] Umesh Bellur and Roshan Kulkarni. Improved matchmaking algorithm for semantic web services based on bipartite graph matching. *ICWS 2007. IEEE International Conference on Web Services, 2007*, 2007.

[6] Umesh Bellur and Harin Vadodaria. On extending semantic matchmaking to include precondition and effect matching. Accepted for publication in the Proceedings of the International Conferences on Web Services, 2008, Beijing, China, 2008.

[7] Simona Colucci et al. A formal approach to ontology-based semantic match of skills descriptions. *Journal of Universal Computer Science, Volume 9, Issue 12*, pages 1437–1454, 2003.

[8] Amit Gupta. Semantic matchmaking algorithms. Technical report, Department of Computer Science and Engineering, IIT-Bombay, 2008. Seminar Report, Third Year BTech Seminar guided by Prof. Umesh Bellur.

[9] Werner Kiebling and Gerhard Kostler. Preference sql: design, implementation, experiences. *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 990–1001, 2002.

[10] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, pages 2:83–97, 1955.

[11] Fabio Porto Le-Hung Vu, Manfred Hauswirth and Karl Aberer. A search engine for qos-enabled discovery of semantic web services. *International Journal of Business Process Integration and Management 2006 - Vol. 1, No.4*, pages 244– 255, 2006.

[12] T. Payne M. Paolucci, T. Kawmura and K. Sycara. Semantic matching of web service capabilities. *Springer Verlag, LNCS, Proceedings of the International Semantic Web Conference*, 2002.

[13] Benedikt Fries Matthias Klusch and Katia Sycara. Automated semantic web service discovery with owls-mx. *AAMAS '06: Proceedings of the Fifth international joint conference on Autonomous agents and multiagent systems*, pages 915–922, 2006.

[14] Jiajin Le ruiqiang Guo and Dehua Chen. Matching semantic web services across heterogenous ontologies. *CIT 05: Proceedings of the Fifth international conference on computer and information technology*, 2005.

[15] Francesco M. Donini Tommaso Di Noia, Eugenio Di Sciascio and Marina Mongiello. Semantic matchmaking in a p2p electronic marketplace. *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 582–586, 2003.

[16] Christos Anagnostopoulos Vassileios Tsetsos and Stathes Hadjiefthymiades. On the evaluation of semantic web service matchmaking systems. *ECOWS '06: Proceedings of the European Conference on Web Services, IEEE Computer Society*, pages 255–264, 2006.

27

Solving Inter-AS Bandwidth Guaranteed Provisioning Problems with Greedy Heuristics

Kin-Hon Ho[1], Ning Wang[2] and George Pavlou[3]
[1]*Department of Computer Science, City University of Hong Kong,*
[2]*Centre for Communication Systems Research, University of Surrey,*
[3]*Department of Electronic and Electrical Engineering, University College London,*
[1]*Hong Kong*
[2,3]*United Kingdom*

1. Introduction

The current Internet consists of more than 26,000 Autonomous Systems (ASes) or domains, each being a network or group of networks managed by a single authority commonly known as Internet Network Provider (INP). With wide deployment of real-time multimedia applications in recent years, the emerging future-generation Internet is expected to provide end-to-end Quality of Service (QoS) guarantees across multiple ASes. In situations where stringent end-to-end QoS is required, ensuring that an adequate bandwidth is guaranteed by each AS along as the entire route in the Internet backbone is essential to achieve relevant performance targets (Zhang et al., 2004). Yet in practice, an AS is only capable of provisioning bandwidth guarantees within its own network. Hence, extending bandwidth guarantees beyond its boundary requires the AS to agree the supply of sufficient bandwidth from other ASes. This bandwidth supply is likely to be associated with a financial cost and therefore there is an economic incentive for an AS to carefully select its downstream provider ASes so as to minimize the cost of using that bandwidth.

Having purchased access to sufficient bandwidth from downstream ASes, the AS needs to utilize both this purchased bandwidth and its own network capacity in the most effective way in order to provide bandwidth guarantees for customer traffic. INPs thus need to optimize the utilization of these resources. Traffic Engineering (TE) is an effective technique to optimize IP operational network performance and subsequently improve network QoS capabilities (Awduche et al., 2002). INPs can thus use TE as an effective means for bandwidth guarantee provisioning while optimizing network resource utilization.

Concatenation of bandwidth guarantees between ASes makes it possible to provide an end-to-end guarantee between a source-destination pair in the Internet. These guarantees across ASes owned by different INPs require some level of agreement between themselves, usually summarized in a negotiated Service Level Agreement (SLA) at the AS level. An SLA is an agreement contracted between a customer AS and a provider AS that describes the characteristics of a service, specifying in particular the supported QoS and the associated cost. However, given that the Internet consists of thousands of ASes, SLA negotiation between ASes has to be carefully managed in an effective and scalable manner. In this chapter we adopt a cascaded negotiation model which allows ASes to build up end-to-end SLAs that provide end-to-end bandwidth guarantees. In this model, apart from route

reachability information, each AS receives from adjacent downstream ASes a set of what we call bandwidth offers to designated remote AS destinations. If an AS decides to accept a bandwidth offer, an SLA is established between the two ASes. The AS can then in turn make bandwidth offers to its upstream (customer) ASes; these offers reflect both the AS' own resources and the SLAs established with the downstream ASes. The full set of SLAs enables all the ASes to support traffic with end-to-end bandwidth guarantees. However, the AS' tasks of making appropriate decisions on which bandwidth offers to accept, how much bandwidth to purchase and how to allocate the bandwidth among traffic aggregates are non-trivial. Inappropriate bandwidth offer selection or traffic assignment could result in respectively high cost or poor resource utilization. In order to obtain the best solutions, we propose a network dimensioning system that incorporates optimization modules that solve the two following problems:

- how to determine an appropriate amount of bandwidth to be purchased from each bandwidth offer so that the total cost of the bandwidth is minimized;
- given the knowledge of the available intra-AS bandwidth and the bandwidth purchased from downstream ASes, how to assign routes to the predicted traffic aggregates so that bandwidth demand is met while optimizing resource utilization.

We call these two problems the *Inter-AS Bandwidth Provisioning* and *Traffic Assignment* problems respectively. Our proposed network dimensioning system enables ASes to move from trial-and-error to a systematic approach for provisioning their end-to-end bandwidth guarantees. More specifically, we propose two efficient greedy heuristics to solve these optimization problems. It has been a long history that greedy heuristics are used for solving network optimization problems, such as traffic engineering (Sridharan et al., 2005), multicast routing (Shi & Turner, 2002) etc. Nevertheless, the optimization problems of end-to-end bandwidth guarantees provisioning across multiple ASes has not been addressed until recently, and in this chapter we will illustrate how greedy heuristics can gracefully solve these novel problems. The main contributions of this chapter can be summarized as follows:

- We propose a systematic network dimensioning system that can be used by ASes to achieve effective provisioning of end-to-end bandwidth guarantees. The network dimensioning system formulates two problems that respectively provide economic and engineering optimization, namely the inter-AS bandwidth provisioning and traffic assignment problems.
- We show that a heuristic approach can be used to solve the inter-AS bandwidth provisioning problem. To illustrate this, we use an efficeint genetic algorithm embedded with two problem-specific greedy heuristics. Our proposed algorithm optimizes the bandwidth provisioning with 5%-30% and 75%-90% less cost than a conventional heuristic and a random-based algorithm respectively.
- We use a greedy-penalty heuristic algorithm to solve the traffic assignment problem. The proposed greedy-penalty heuristic results in 10% less total bandwidth consumption than a random-based algorithm.

2. Cascaded inter-AS negotiation model

The provision of end-to-end bandwidth guarantees requires each intermediate AS on the path from the source AS to the destination AS to guarantee the agreed bandwidth. However, this cannot be realized without first negotiating and agreeing SLAs among the ASes. Since the Internet is a collection of a large number of ASes, attention needs to be paid to how to manage such negotiation and SLA establishment in an effective and scalable manner. In this chapter,

we adopt a cascaded model, as proposed by the MESCAL project for negotiating QoS guarantees (e.g. bandwidth and delay) among ASes (Howarth et al., 2005).
The model is based on two concepts: (1) negotiation of bandwidth offers between ASes; (2) establishment of unidirectional SLAs between ASes for the agreed bandwidth. The key idea of the cascaded model is as follows. An AS offers bandwidth guarantees to its upstream ASes; each bandwidth offer specifies the reachable remote destination(s), the available bandwidth (e.g., maximum offered bandwidth) and a cost, for example, per unit of bandwidth. These destinations are either in customer ASes or reachable through downstream ASes. An upstream AS in general receives multiple bandwidth offers for any given destination, and has to decide which one to accept. Each accepted bandwidth offer is then established as a unidirectional SLA. The AS can then in turn make bandwidth offers to its upstream ASes, by combining its local bandwidth capabilities with the SLA. This process continues in a cascaded manner for further upstream ASes, and an end-to-end SLA chain can be built, with each SLA relying on the SLAs between downstream ASes.

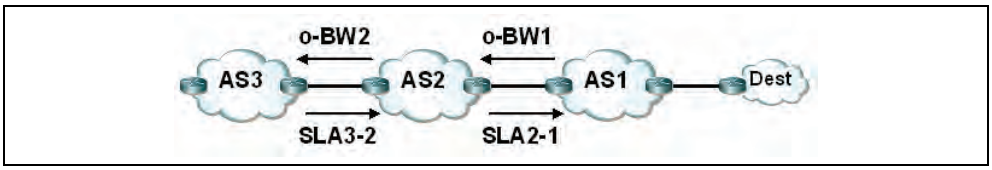


Fig. 1. Illustration of the cascaded inter-AS negotiation model

Fig. 1 illustrates an example. Let *o-BW1* be the bandwidth guarantee offered by AS1 towards destination 'dest'. AS2 receives this offer *o-BW1*. We assume that AS2 decides to accept the bandwidth offer: AS2 then establishes an SLA with AS1 (*SLA2-1*) for this bandwidth. Now AS2 has a bandwidth guarantee provided by AS1 for access to 'dest'. AS2 can in turn extend this bandwidth guarantee by concatenating its local bandwidth capability with *SLA2-1*, and then offering a bandwidth (*o-BW2*) to AS3. *o-BW2* is the minimum of (a) the local bandwidth capability that AS2 is prepared to guarantee across its network and (b) *SLA2-1*. Now *o-BW2* indicates the bandwidth guarantee from AS2 to destination 'dest'. AS3 receives *o-BW2* from AS2 and it in turn repeats the decision process, possibly purchasing the offered bandwidth and establishing *SLA3-2*. In summary, once offers from other adjacent downstream ASes have been agreed as SLAs, an INP may build new extended services upon cascaded existing ones. Further details of the cascaded model can be found in (Howarth et al., 2005).
The cascaded model has several advantages: (1) it makes possible to build scalable end-to-end QoS guarantees between any two ASes while only maintaining SLAs with adjacent ASes; (2) it has backward compatibility with BGP, making inter-AS QoS deployment possible through extensions to BGP; (3) it retains privacy for all ASes regarding the details of their interactions.
The decision on which bandwidth offers to accept, and how to effectively utilize the established SLAs and the AS' intra-AS resources is non-trivial. In the next section, we propose a network dimensioning system, incorporating TE mechanisms, to solve this problem and make the best decisions.

3. Decomposition of the network dimensioning system

We consider two optimization problems, an economic and an engineering one, that need to be solved for provisioning end-to-end bandwidth guarantees. First, an AS needs to

determine the appropriate amount of bandwidth to be purchased from each adjacent downstream AS so that the total bandwidth cost is minimized. Second, given these available bandwidth resources defined in the SLAs and the local network's bandwidth, the AS has to determine how to assign routes to the supported traffic in order to satisfy their bandwidth requirements while at the same time optimizing network resource utilization. We illustrate on Fig. 2 a decomposition of a network dimensioning system which consists of several components. We envisage this system as being offline and running infrequently as part of a resource provisioning cycle, e.g. in the order of weeks.

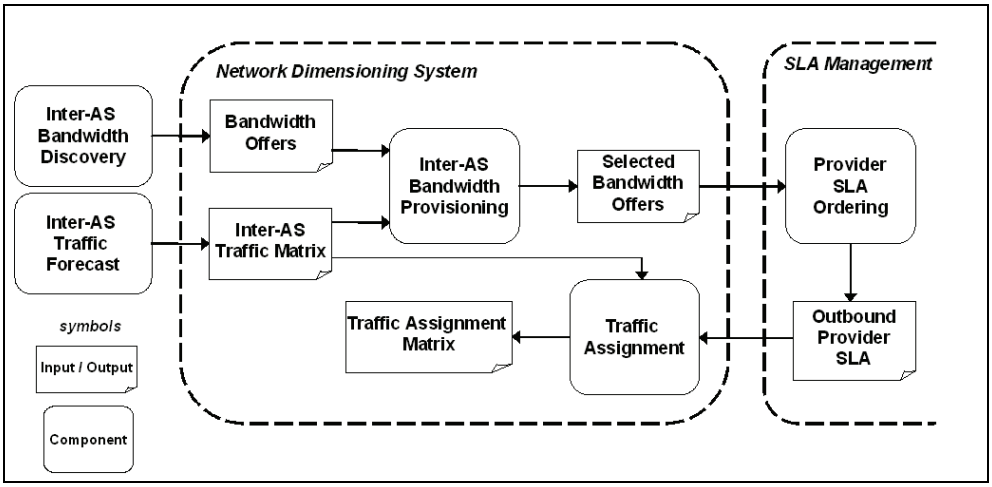


Fig. 2. Architecture of the network dimensioning system

3.1 Components of the network dimensioning system

The proposed network dimensioning system consists of the following components:

1. ***Inter-AS Traffic Forecast*** predicts inter-AS traffic in the network for a period of time and records this information in an inter-AS Traffic Matrix (TM). Each element in the inter-AS TM is the aggregate traffic load that enters the network at an ingress point† and is destined for a remote destination prefix. The TM entry is represented by the tuple

   < *ingress point, remote destination prefix, long-term average traffic demand* >

   Some known methods can be used to compute the traffic aggregate, such as the effective bandwidth approach (Guerin et al., 1991) if the mean and peak rates of the traffic are known.

   The inter-AS TM is an important element for network and traffic engineering. Whilst an accurate inter-AS TM could be obtained through fine-grained flow-level traffic measurement this is not suitable for long term predictions (Awduche et al., 2002). Nevertheless, these problems have recently been addressed with a methodology that allows an inter-AS TM to be predicted through measurement (Teixeira et al., 2005) and estimation for web traffic (Feldmann et al., 2004). Alternatively, an inter-AS TM can be extrapolated from customer SLAs.

2. ***Inter-AS Bandwidth Discovery*** discovers bandwidth offers from adjacent downstream ASes through offline techniques, e.g. advertisement. A bandwidth offer is uniquely identified by a connection point at which the offer is provided. Bandwidth offers are provided by adjacent ASes, and so the connection point, or inter-AS link on which it is offered, uniquely identifies the adjacent AS. Each bandwidth offer specifies a maximum bandwidth towards a remote destination prefix and is associated with a cost, for example per unit of bandwidth. Each bandwidth offer is represented by the tuple

   < *egress router, adjacent AS border router address , remote destination prefix,*

   maximum offered bandwidth, cost >

3. ***Inter-AS Bandwidth Provisioning*** (IBP) addresses the economic problem described in the beginning of this section. For the sake of service resilience and load balancing, an increasing number of ASes have multiple connections to adjacent downstream ASes. As a result, an AS may receive multiple offers to each destination prefix from different adjacent downstream ASes. The goal of IBP is to take as input the inter-AS TM and a set of bandwidth offers, and to produce as output a decision on which bandwidth offers to accept and the amount of bandwidth to be purchased from each of the accepted offers. Based on the IBP outcome, the AS will then establish SLAs (in this chapter called outbound provider SLAs) with the adjacent downstream ASes to contract the bandwidth guarantees. We assume that the establishment of outbound provider SLAs is performed by the component "provider SLA ordering", a process whose details are outside the scope.
4. ***Traffic Assignment*** (TA) deals with the engineering problem described in the beginning of this section. The goal of TA is to take as input an inter-AS TM, a set of outbound provider SLAs that are established after the IBP phase, and the available bandwidth resources of the AS, i.e. intra- and inter-AS link capacities, and then to assign appropriate routes for the supported traffic so that the bandwidth requirements are met while optimizing network resource utilization. An assignment of the route includes selection of an outbound provider SLA, an inter-AS link and an intra-AS route for the supported traffic. The key output of the TA is a Traffic Assignment matrix that records the outbound provider SLAs, inter-AS links and intra-AS routes that have been selected for the supported traffic. Based on this matrix, an INP can implement the TA solution by configuring the network accordingly.

3.2 Inter-AS bandwidth overprovisioning

We can employ overprovisioning in the IBP phase. This implies that some network resources are left unused so as to protect the core backbone from failures and to accommodate some degree of traffic demand fluctuation (Nucci et al., 2005). Overprovisioning is also the current solution adopted by some INPs for QoS provisioning within their networks. For these reasons, we consider a certain amount of inter-AS bandwidth overprovisioning in this chapter. During the IBP phase, the AS should not merely purchase bandwidth that marginally accommodates the forecasted traffic demand, because the bandwidth guarantee may not be maintained if even a small traffic upsurge occurs. A solution to this is to purchase more bandwidth than the forecasted traffic demand in order to insure against such traffic fluctuations. This also provides a buffer against inter-AS link failures, which may cause traffic to be shifted from one outbound provider SLA to another.

The task of IBP is thus to decide an appropriate amount of bandwidth to be purchased from the adjacent downstream ASes by taking into account overprovisioning. To do so, we introduce an overprovisioning factor $f_{over} \geq 1.0$ to specify the degree of inter-AS bandwidth overprovisioning. In principle, this factor is determined by considering the network's traffic characteristics and the target link utilization. However, since optimization of f_{over} is not the subject to be concerned, we assume that a single value is used to represent the optimal overprovisioning that has already been determined by the ASes. The concept of overprovisioning factor has also been used by other researchers, e.g. (Nucci et al., 2005).
In this work, inter-AS bandwidth overprovisioning is implemented as follows. If $t(i,k)$ denotes the average demand of an inter-AS traffic flow aggregate, we define an inflated traffic flow, $t'(i,k) = t(i,k) \cdot f_{over}$.

4. Optimal inter-AS bandwidth provisioning

In this section and the next, we present the problem statement, formulation and algorithms of both the IBP and the TA problems.

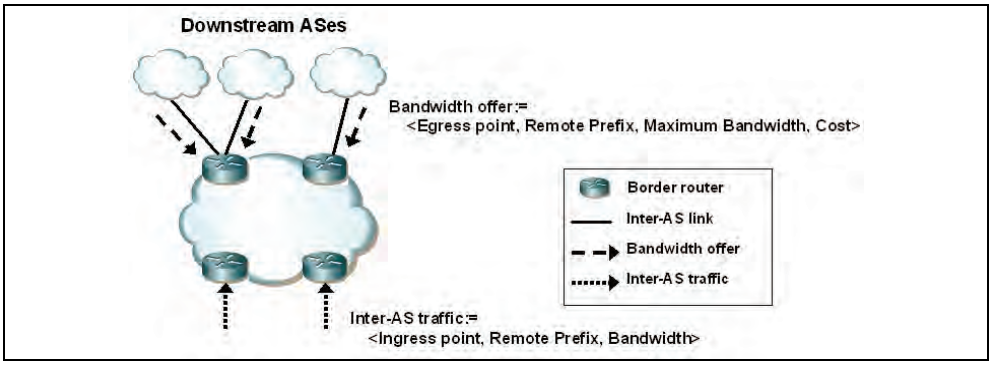


Fig. 3. Elements of the inter-AS bandwidth provisioning

Fig. 3 illustrates an AS topology with the key elements of the IBP problem. A set of border routers is connected to adjacent ASes. An ingress (or egress) router is the border router that receives (or sends) traffic from (or to) an adjacent AS. Each border router is associated with one or more inter-AS links. Each bandwidth offer is associated with a single inter-AS link. Each border router may receive multiple bandwidth offers for a remote destination prefix from different adjacent downstream ASes through different attached inter-AS links, for example, the top left border router in Fig. 3. Each inter-AS traffic flow enters the AS through a designated ingress router. We define the total inter-AS bandwidth provisioning cost to be the total charge an AS pays for purchasing bandwidth from its adjacent downstream ASes. The inter-AS bandwidth provisioning problem can be summarized as follows:
Given a set of bandwidth offers from adjacent downstream ASes, an inter-AS traffic matrix and a physical network topology, determine an appropriate amount of bandwidth to be purchased from each bandwidth offer so that the total inter-AS bandwidth provisioning cost is minimized while respecting the capacity constraints of the inter-AS links.
In solving the IBP problem we assume that the inter-AS traffic is non-splittable. This method not only can determine the appropriate amount of bandwidth to be purchased but also ensures that each traffic flow will be accommodated by at least one SLA during TA without causing the traffic to be split.

Note that some types of ASes, such as tier 2 and 3, may have both peering and customer-provider connections with adjacent ASes. A peering connection between two ASes refers to the case where each AS carries a similar amount of customer traffic from the other AS for free. On the other hand, a customer-provider connection refers to the case where the provider charges the customer for carrying traffic across its network. The IBP description in Section 3 assumed that an AS has only customer-provider connections with its adjacent downstream ASes and that a cost is associated with each bandwidth offer. In fact, peering connections can also be considered by IBP. In this case, the cost of bandwidth is typically zero and the maximum bandwidth represents the agreed amount of traffic to be exchanged.

4.1 Inter-AS bandwidth provisioning problem formulation

We formulate IBP as an integer programming problem. Table 1 shows the notation used throughout this chapter. The objective of the IBP problem is to minimize the total IBP cost:

$$Minimize \sum_{i\in I}\sum_{k\in K}\sum_{oBw(k,j,n)\in Out(k)} x_{i,k}^{j,n}\cdot chg_{k}^{j,n}\cdot t'(i,k) \tag{1}$$

subject to:

$$\sum_{i\in I}\sum_{k\in K} x_{i,k}^{j,n}\cdot t'(i,k)\le c_{inter}^{j,n} \quad \forall(j,n) \text{ where } j\in J, n\in NEXT_j \tag{2}$$

$$\sum_{i\in I} x_{i,k}^{j,n}\cdot t'(i,k)\le MaxBw_{k}^{j,n} \quad \forall(k,j,n) \text{ where } k\in K, j\in J, n\in NEXT_j \tag{3}$$

$$x_{i,k}^{j,n}, y_{k}^{j,n}\in\{0,1\} \tag{4}$$

$$x_{i,k}^{j,n}\le y_{k}^{j,n} \quad \forall(i,k,j,n) \text{ where } i\in I, k\in K, j\in J, n\in NEXT_j \tag{5}$$

$$\sum_{oBw(k,j,n)\in Out(k)} x_{i,k}^{j,n}=1 \quad \forall(i,k) \text{ where } i\in I, k\in K \tag{6}$$

$$\sum_{n\in NEXT_j} y_{k}^{j,n}\le 1 \quad \forall(k,j) \text{ where } k\in K, j\in J \tag{7}$$

Constraint (2) ensures that no inter-AS link carries traffic exceeding its capacity. Constraint (3) ensures that no bandwidth offer carries traffic exceeding its maximum capacity. Constraint (4) ensures that the discrete variables assume binary values. Constraint (5) ensures that, whenever traffic flow *t'(i,k)* is assigned to bandwidth offer $oBw_{k}^{j,n}$, then this bandwidth offer must have been selected. Constraint (6) ensures that only one bandwidth offer is selected for each inter-AS traffic flow. Hence, traffic splitting over multiple bandwidth offers is not considered. Constraint (7) ensures that only one of the bandwidth offers, which are advertised at a border router through different inter-AS links, is selected for each remote destination prefix. This constraint ensures the BGP rule that only one route toward a remote destination prefix is selected as the best route. This makes the IBP implementation easier through BGP configuration.

Notation	Description
	- General notation -
K	A set of destination prefixes
I	A set of ingress routers
J	A set of egress routers
f_{over}	Overprovisioning factor
$t(i,k)$	Bandwidth demand of an inter-AS traffic flow entering the AS at ingress router $i \in I$ towards destination prefix $k \in K$. It is considered by the TA problem
$t'(i,k)$	Inflated traffic flow $t(i,k)$. It is considered by the IBP problem
$Out(k)$	A set of bandwidth offers that has reachability to destination prefix k
$NEXT_j$	A set of next hop addresses (addresses of the border routers in adjacent downstream ASes) that is associated with egress router $j \in J$
$C_{inter}^{j,n}$	Capacity of the inter-AS link that connects egress router j to next-hop address $n \in NEXT_j$
$bw_{inter}^{j,n}$	Residual bandwidth of $C_{inter}^{j,n}$
C_{intra}^{l}	Capacity of intra-AS link l
bw_{intra}^{l}	Residual bandwidth of C_{intra}^{l}
	- Notation used in the IBP problem -
$oBw_k^{j,n}$	Bandwidth offer that is associated with destination prefix k and is advertised through the inter-AS link that connects egress router j to next-hop address n
$MaxBw_k^{j,n}$	Maximum bandwidth of the offer $oBw_k^{j,n}$
$Chg_k^{j,n}$	A charge per unit bandwidth for $oBw_k^{j,n}$
$x_{i,k}^{j,n}$	Variable indicating whether traffic flow $t'(i,k)$ is assigned to bandwidth offer $oBw_k^{j,n}$
$y_k^{j,n}$	Variable indicating whether the bandwidth offer $oBw_k^{j,n}$ is selected
	- Notation used in the TA problem -
$pSLA_k^{j,n}$	Outbound provider SLA of the bandwidth offer $oBw_k^{j,n}$
$pSLAC_k^{j,n}$	Contracted bandwidth specified in outbound provider SLA $pSLA_k^{j,n}$
$pSLABw_k^{j,n}$	Residual bandwidth of $pSLAC_k^{j,n}$
$dist_{i,j}^{k}$	Number of hops on the intra-AS route between ingress router i and egress router j towards destination prefix k
$P_{i,j}$	A set of feasible intra-AS routes between ingress router i and the egress router j to which the selected outbound provider SLA is associated.
$w_{i,k}^{p}$	Variable indicating whether path $p \in P_{i,j}$ is chosen to realize traffic flow $t(i,k)$
$z_{i,k}^{j,n}$	Variable indicating whether traffic flow $t(i,k)$ is assigned to outbound provider SLA $pSLA_k^{j,n}$
$\Upsilon_{i,k}^{l}$	Variable indicating whether traffic flow $t(i,k)$ is assigned to intra-AS link l

4.2 Modified inter-AS bandwidth provisioning problem

We assume that when multiple bandwidth offers towards the same remote destination prefix k are present at a given border router j (i.e. $\exists n \ \ MaxBw_k^{j,n} > 0$), the AS has already determined the best one as a candidate bandwidth offer. Thus, each border router will consider at most one bandwidth offer towards each remote destination. The decision of selecting the best bandwidth offer might be based on business factors such as the relationships between ASes and the reputations of adjacent downstream ASes. As a result of this assumption, the variable $y_k^{j,n}$ of which bandwidth offer has been considered for each remote destination prefix k at each border router j is pre-determined and this satisfies constraint (7) since at most one bandwidth offer will be considered (i.e. $\sum_{n \in NEXT_j} y_k^{j,n} \le 1$). Therefore, constraint (7) is automatically enforced.

4.3 A Lower bound of the inter-AS bandwidth provisioning problem

We derive an approximated optimal solution of the IBP problem that can be obtained efficiently by relaxing some constraints. This approximated optimal solution is thus a lower bound of the IBP problem. A lower bound typically has better result than the optimal solution because some problem constraints are relaxed. However, due to the relaxation, it is not a valid solution to the problem. Nevertheless, the lower bound is a good approximation of an optimal solution for heuristic algorithms to compare their performance. We show the derivation of a lower bound for the IBP problem as follows.

We derive a lower bound by relaxing some IBP problem constraints. First of all, constraint (7) is automatically enforced by our assumption that each border router has only considered the best candidate bandwidth offer towards each remote destination prefix. Second, we relax the non-bifurcation integer constraint (4). In many practical situations, integer programming problems, which require all variables to be integers, are NP-hard. Instead, a linear programming problem that has only non-integer variables can be generally solved efficiently in the worst case. Therefore, we relax constraint (4) to

$$0 \le \chi_{i,k}^{j,n} \le 1 \text{, non-integer} \tag{8}$$

Finally, we find that a lower bound can be readily calculated by the following method if inter-AS link capacity constraint (2) is relaxed. Relaxation of a capacity constraint means that the constraint is simply ignored based on the assumption that capacity is large enough to accommodate the traffic.

Given $Pr_{low} = \underset{\forall k,j,n}{Min} \ \ Chg_k^{j,n}$ and $Pr_{high} = \underset{\forall k,j,n}{Max} \ \ Chg_k^{j,n}$, we can define

$$b_k^{\psi} = \sum_{j \in J} \sum_{n \in Next_j} MaxBw_k^{j,n} \mid Chg_k^{j,n} = \psi \quad \forall \ Pr_{low} \le \psi \le Pr_{high} \ and \ k \in K \tag{9}$$

where $b_k^{\psi} \ge 0$ is the sum of maximum capacity of all the bandwidth offers to remote destination prefix k with a charge equal to ψ, and

$$d_k = \sum_{i \in I} t'(i,k) \quad \forall \ k \in K \tag{10}$$

where $d_k \geq 0$ is the sum of bandwidth demands of all the traffic flows to destination prefix k. For each traffic demand d_k towards remote destination prefix k, we first attempt to assign it to the lowest cost bandwidth offer. If the lowest cost bandwidth offer cannot entirely accommodate the traffic demand due to capacity limitation, then the residual demand will be assigned to the next lowest cost bandwidth offer. This traffic demand assignment iterates until the bandwidth offer with a particular cost can entirely accommodate the traffic demand. A lower bound is calculated based on the traffic assigned to each bandwidth offer and its associated cost. A lower bound, using the abovementioned method, can be calculated by

$$\sum_{k \in K} \sum_{\psi = Pr_{low}}^{Pr_{high}} \left\{ Min\left[Max\left(d_k - \sum_{\alpha = Pr_{low}}^{\psi - 1} b_k^{\alpha}, 0 \right), b_k^{\psi} \right] \cdot \psi \right\} \tag{11}$$

For a particular cost ψ, the *max* function determines the residual traffic demand that has not been allocated to the bandwidth offers that have lower cost than the one being considered. The *min* function attempts to assign this residual traffic demand to the bandwidth offer with the cost currently being considered. The inner summation symbol considers all bandwidth offers toward a remote destination prefix with different costs. The outer summation symbol considers all the remote destination prefixes.

4.4 A genetic algorithm embedded with greedy heuristics

We propose an efficient Genetic Algorithm (GA) to obtain a near-optimal solution of the IBP problem. Genetic Algorithm is an algorithm that operates by the natural selection of 'survival of the fittest (Holland 1975). It has been successful in solving many large-scale optimization problems. In order to making the proposed GA in solving the IBP problem more efficiently, we propose two problem-specific greedy heurtsics embedded into the GA.

To solve the IBP problem, we modify and extend the GA (Chu & Beasley, 1997) proposed for solving the Generalized Assignment Problem (Martello & Toth, 1990). The steps of our GA are as follows:

Step 1. Create a feasibility mapping table which maps all the feasible bandwidth offers to each inter-AS traffic flow. A bandwidth offer $oBw_k^{j,n}$ is feasible for an inter-AS traffic flow $t'(i,k)$ if the following constraints are satisfied:

$$oBw_k^{j,n} \in Out(k) \tag{12}$$

$$t'(i,k) \leq C_{inter}^{j,n} \tag{13}$$

$$t'(i,k) \leq MaxBw_k^{j,n} \tag{14}$$

Constraint (12) ensures that the remote destination prefix in the bandwidth offer matches the requested remote destination prefix of the traffic flow. Constraints (13) and (14) ensure respectively that the bandwidth demand of the traffic flow does not exceed the capacity of either the inter-AS link to which the bandwidth offer is associated or the maximum capacity of the bandwidth offer. These constraints, however, do not guarantee that constraints (2) or (3) are met for the entire chromosome.

Step 2. Generate an initial population of C randomly constructed chromosomes. Fig. 4 shows a representation of an individual chromosome which consists of T genes where T is the number of inter-AS traffic flows and each gene represents an assignment between a traffic flow and a bandwidth offer. The identifier given to each traffic flow represents each inter-AS traffic flow $t'(i,k)$. Let $S_{t'(i,k),c} = <k,j,n>$ represent the bandwidth offer $oBw_k^{j,n}$ that has been assigned to traffic flow $t'(i,k)$ in chromosome $c \in C$. Each gene of the initial chromosomes is generated by randomly assigning a feasible bandwidth offer to each traffic flow according to the feasibility mapping table created in step 1. Note that an initial chromosome may not be a feasible solution as capacity constraint (2) or (3) could be violated.

Traffic flow	1	2	⋯	T-1	T
Bandwidth offer	o-BW1	o-BW2	⋯	o-BWm	o-BWn

Fig. 4. Representation of an individual's chromosome

Step 3. Decode each chromosome to obtain its fitness value. The fitness of chromosome c is equal to the total inter-AS bandwidth provisioning cost, given by

$$-\sum_{i\in I}\sum_{k\in K} Chg(s_{t'(i,k),c})\cdot t'(i,k) \tag{15}$$

The negative sign reflects the fact that a solution with lower cost has higher fitness. We define $Chg(s_{t'(i,k),c})\cdot t'(i,k)$ to be the IBP cost for the traffic flow $t'(i,k)$. If the chromosome contains an infeasible solution, a common approach is to penalize its fitness for the infeasibility. Instead of this, we adopt the approach in (Chu & Beasley, 1997) and associate an unfitness value for each chromosome. The unfitness value of chromosome c is the degree of infeasibility of the chromosome, which equals the amount of violated capacity summed over all the inter-AS links and all the bandwidth offers,

$$\begin{aligned}&\sum_{j\in J}\sum_{n\in Next_j} Max\left\{0\ ,\ \sum_{i\in I,k\in K:S_{t'(i,k),c}=<k,j,n>} t'(i,k) - C_{inter}^{j,n}\right\} + \\ &\sum_{k\in K}\sum_{j\in J}\sum_{n\in Next_j} Max\left\{0\ ,\ \sum_{i\in I:S_{t'(i,k),c}=<k,j,n>} t'(i,k) - MaxBw_k^{j,n}\right\}\end{aligned} \tag{16}$$

With the separation of fitness and unfitness values, chromosomes can be evaluated in a two-dimensional plane, so the selection and replacement can direct the search towards feasible solutions by replacing highly unfit chromosomes with lightly unfit or entirely fit ones.

Step 4. Select two parent chromosomes for reproduction. We use the pairwise tournament selection method. In pairwise tournament selection, two individual chromosomes are chosen randomly from the population and the one that is fitter (higher fitness value) is selected for a reproductive trial. Two pairwise tournament selections are held, each of which produces one parent chromosome, in order to produce a child chromosome.

Step 5. Generate two child chromosomes by applying a simple one-point crossover operator on the two selected parents. The crossover point p_{co} is randomly selected. The first child chromosome consists of the first p_{co} genes from the first parent and the remaining $(n - p_{co})$ genes from the second parent. The second child chromosome takes the parent genes that have not been considered by the first child chromosome.
Step 6. Perform a probabilistic mutation on each child chromosome. The mutation simply exchanges elements in two selected genes (i.e. exchange the assigned bandwidth offers between two randomly selected traffic flows) without violating constraints (12) – (14).
Step 7. The fitness and unfitness values of child chromosomes can be improved by applying the following two problem-specific heuristic operators:

- *Heuristic-A*: For each inter-AS traffic flow that has been assigned to an infeasible bandwidth offer such that either capacity constraint (2) or (3) is violated, find a feasible bandwidth offer that incurs the lowest IBP cost for the traffic flow. Denote $\Delta f(i,k)$ the difference between the original IBP cost induced by the traffic flow and the new IBP cost after the traffic flow has been reassigned to a feasible bandwidth offer. Among those inter-AS traffic flows, select the one with the lowest $\Delta f(i,k)$ and assign it to the corresponding selected feasible bandwidth offer. This heuristic operator iterates at most H times where H is a parameter that optimizes the algorithm's performance or stops when no inter-AS traffic flows have been assigned to infeasible bandwidth offers.
- *Heuristic-B*: For each inter-AS traffic flow, find a feasible bandwidth offer that produces the lowest IBP cost. If such a feasible bandwidth offer has been found, reassign the traffic flow to it.

Heuristic-A aims to reduce the unfitness value of the child chromosome by reassigning traffic flows from infeasible to feasible bandwidth offers while keeping the total IBP cost as low as possible. *Heuristic-B* attempts to improve the fitness of the child chromosome by reassigning traffic flows to feasible bandwidth offers with lower costs.
Step 8. Replace two chromosomes in the population by the improved child chromosomes. In our replacement scheme, chromosomes with the highest unfitness are always replaced by the fitter child chromosomes. If no unfit solution exists, the lowest fitness ones are replaced.
Step 9. Repeat step 4 - 8 until N_{cd} child chromosomes have been produced and placed in the population.
Step 10. Check if the GA termination criterion is met. The termination criterion is that either both the average and the best fitness over all the chromosomes in the two consecutive generations are identical or once the selected number of iterations, N_{it}, has been reached in order to avoid excess algorithm execution time. Steps 4 - 9 iterate until the termination criterion is met.

5. Optimal traffic assignment

Let us assume that the bandwidth offers selected by the IBP (Section 4) have now been accepted and configured as a set of outbound provider SLAs. Given this set and the available bandwidth capacity within the AS, we now consider how to assign routes to the traffic so as to meet the traffic's bandwidth requirements. Fig. 5 shows that from the viewpoint of AS-1, a route to the destination can be decomposed into three parts: (1) the intra-AS route, (2) the inter-AS link and (3) the inter-AS route from the downstream AS (AS-2) to the destination AS (AS-3). Sufficient bandwidth must be provisioned in all parts of this route in order to satisfy the bandwidth demand. Once the outbound provider SLA is

known, the available bandwidth resource on any part of the route is known to the AS: the intra- and inter-AS links are owned by the AS and the available bandwidth from the downstream AS to the destination AS is guaranteed by the outbound provider SLA. As a result, the TA problem can be defined as follows:

Given a set of outbound provider SLAs, an inter-AS TM and a physical network topology, assign end-to-end routes to the supported traffic so that the bandwidth requirement is satisfied while optimizing network resource utilization. A route assignment includes the selection of an outbound provider SLA, an inter-AS link and an explicit intra-AS route from the ingress router to the egress router where the selected outbound provider SLA is associated.

We assume that explicit intra-AS routes are implemented by MPLS. In addition, there are many optimization criteria for network resource utilization, such as minimizing resource consumption or load balancing. For simplicity, the network resource utilization used in this chapter is a general metric, the total bandwidth consumed in carrying traffic across the network.

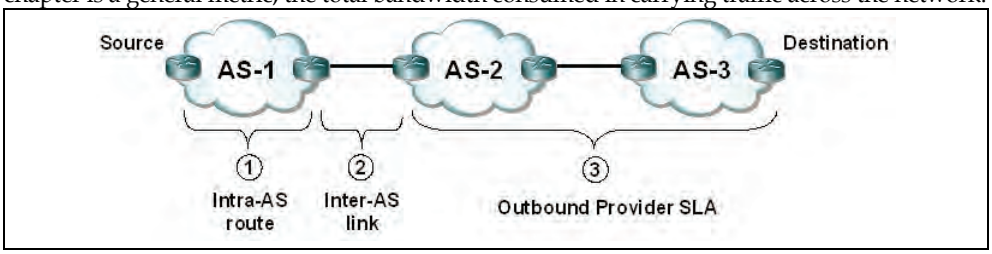


Fig. 5. Essential components for end-to-end bandwidth guarantee

5.1 Traffic assignment problem formulation

As with the IBP problem of Section 4, we formulate the TA problem as an integer-programming problem. The fundamental objective is to provide bandwidth guarantees to inter-AS traffic by satisfying their bandwidth demands. We define the bandwidth demand of an inter-AS traffic flow *t(i,k)* to be met if the following three constraints are satisfied:

There exists at least one feasible path fpath∈Pi,j from ingress router i to egress router j to which the selected outbound provider SLA is associated , i.e.

$$\underset{\forall l \in f_{path}}{Min}\ bw_{\text{intra}}^{l} \geq t(i,k) \tag{17}$$

$$bw_{inter}^{j,n} \geq t(i,k) \tag{18}$$

$$pSLABw(k,\ j,n) \geq t(i,k) \tag{19}$$

Constraint (17) ensures that there exists at least one feasible path between the ingress point and the selected egress point, and the bottleneck bandwidth of the path is not less than the bandwidth demand of the traffic flow. Constraints (18) and (19) ensure that the inter-AS link and the outbound provider SLA respectively have sufficient bandwidth to accommodate the traffic flow.

The objective of minimizing the total bandwidth consumption within the network can be translated to the problem of minimizing the total number of hops that a traffic flow must traverse in the network, i.e.

$$Minimize \sum_{i\in I}\sum_{k\in K}\sum_{oBw(k,j,n)\in Out(k)} \boldsymbol{z}_{i,k}^{j,n}\cdot dist_{i,j}^{k}\cdot t(i,k) \tag{20}$$

subject to:

$$\sum_{i\in I}\sum_{k\in K}\boldsymbol{z}_{i,k}^{j,n}\cdot t(i,k)\le \boldsymbol{C}_{inter}^{j,n} \quad \forall(j,n) \text{ where } j\in J, n\in NEXT_j \tag{21}$$

$$\sum_{i\in I}\sum_{k\in K}\Upsilon_{i,k}^{l}\cdot t(i,k)\le \boldsymbol{C}_{intra}^{l} \quad \forall l\in E \tag{22}$$

$$\sum_{i\in I}\boldsymbol{z}_{i,k}^{j,n}\cdot t(i,k)\le pSLAC_{k}^{j,n} \quad \forall(k,j,n) \text{ where } k\in K, j\in J, n\in NEXT_j \tag{23}$$

$$\boldsymbol{z}_{i,k}^{j,n}, \Upsilon_{i,k}^{l}, \boldsymbol{w}_{i,k}^{p}\in\{0,1\} \tag{24}$$

$$\sum_{oBw(k,j,n)\in Out(k)}\boldsymbol{z}_{i,k}^{j,n}=1 \quad \forall(i,k) \text{ where } i\in I, k\in K \tag{25}$$

$$\sum_{p\in P_{i,j}}\boldsymbol{w}_{i,k}^{p}=1 \quad \forall(i,k) \text{ where } i\in I, k\in K \tag{26}$$

$$\Upsilon_{i,k}^{l}\le \boldsymbol{w}_{i,k}^{p} \quad \forall(l,p,i,k) \text{ where } l\in p, p\in P_{i,j}, i\in I, k\in K \tag{27}$$

Constraints (21), (22) and (23) ensure that the total traffic assigned to the inter-AS link, the intra-AS link and the outbound provider SLA do not exceed their respective capacities. Constraint (24) ensures the discrete variables assume binary values. Constraint (25) ensures that only one outbound provider SLA is selected for each traffic flow. Constraint (26) ensures that each traffic flow *t(i,k)* is routed along a single intra-AS route in order to preserve scalability and minimize network management complexity. Constraint (27) ensures that, whenever traffic flow *t(i,k)* is assigned to intra-AS link *l*, then the path to which *l* is associated must have been selected. Moreover, given the lossless property of the links, an additional constraint that has not been presented is the flow conservation constraint which ensures that the traffic flowing into a node must equal the traffic flowing out of the node for any intermediate node.

5.2 A greedy heuristic algorithm for the traffic assignment problem

In comparing the two problems in the network dimensioning system, the complexity of the TA Problem is higher than the IBP problem, in terms of number of decision variables and constraints. In addition, the TA is performed more frequently than the IBP: network capacity expansion is usually less frequent than traffic engineering. Based on these reasons, the algorithm for solving the TA problem should be more *efficient* than the IBP algorithm. In general, a GA can produce a better performance but with higher time complexity than simple greedy-based heuristics. Due to the higher complexity of the TA problem, we do not consider using GA to solve the TA problem as we did for the IBP problem. Instead, we present a simple and efficient greedy heuristic algorithm to solve the TA problem, namely greedy-penalty heuristic.

Greedy-penalty heuristic: It is possible that the order in which traffic flows are assigned to outbound provider SLAs may produce different selection results. For example, if we take a traffic flow $t(i,k) = 2$, we might assign it greedily to some outbound provider SLA $pSLA_k^{j,n}$ with intra-AS distance $dist_{i,j}^k = 3$. In this case, the total bandwidth consumed equals 6. If on the other hand we allocate it later in the process, the outbound provider SLA may not have sufficient bandwidth because its bandwidth has been allocated to other traffic flows and the considered traffic flow might have to be assigned to another outbound provider SLA $pSLA_k^{j',n'}$, for example, with $dist_{i,j'}^k = 6$. As a result, the total bandwidth consumed equals 12. In this case, we have a penalty on the consumption of additional bandwidth (i.e. 12 − 6 = 6) and we use *penalty* to refer to this value. A penalty-based algorithm aims to minimize the number of hops a flow must traverse by placing customer traffic flows in certain order according to *penalty*. We propose a greedy-penalty heuristic algorithm that takes into consideration the penalty value. Such an algorithm has also been used to solve the GAP (Martello & Toth, 1990).

Step 1 For each unassigned traffic flow, we measure the desirability of assigning it to each feasible outbound provider SLA that satisfies constraint (19). The desirability is the total bandwidth consumed by the traffic flow along the intra-AS route between the ingress and the egress router with which the outbound provider SLA is associated (i.e. the number of intra-AS hops times the bandwidth demand). Intra-AS route computation is done by Constrained Shortest Path First (CSPF) (Osborne & Simha, 2002), which finds a route that is shortest in terms of hop while satisfying the bandwidth requirement. The smaller the desirability, the smaller amount of bandwidth to be consumed, and thus the better the selection.

Step 2 Compute *penalty* for each unassigned traffic flow, being the difference between the desirability of the traffic flow's best and second best selection (i.e. the two outbound provider SLAs which yield the smallest desirability). If there is only one feasible outbound provider SLA with sufficient spare capacity to accommodate the traffic flow, we need to set *penalty* to infinity and immediately assign the traffic flow to it. Otherwise, this outbound provider SLA may subsequently become unavailable, resulting in an invalid solution.

Step 3 Among all unassigned traffic flows, the one yielding the largest *penalty* is placed with its best selection. In other words, this traffic flow is assigned to the feasible outbound provider SLA that achieves the smallest desirability. If multiple traffic flows which have the same largest penalty exist, the one with the largest bandwidth demand is placed. If there are several such traffic flows, one is chosen randomly.

Step 4 Once the outbound provider SLA is selected, the requested bandwidth is allocated on the corresponding selected intra-AS route and the outbound provider SLA to establish an end-to-end bandwidth guaranteed route. We iterate step 1 to step 4 until all the traffic flows have been considered.

6. Performance evaluation

We evaluate the proposed GA and the greedy-penalty heuristic algorithms by simulation. The simulation software was written in Java. The computation was carried out on a laptop with an Intel Pentium Centrino 1.5GHz Processor with 512MB RAM. All the results presented in this chapter are an average of 50 different simulation trials.

6.1 Network model

We use a network topology generated by BRITE (Brite) with 100 nodes and average node degree of 4. These numbers were chosen to represent a medium to large INP topology. All intra-AS links are unidirectional and each has capacity of 500 units. Note that, since no realistic data is publicly available, we assume that the values of link capacity, bandwidth offers, and traffic demand are unitless. Therefore, these values that we use in this chapter may represent any specific value depending on the definition of the corresponding unit.

Among the 100 nodes, 30 nodes are randomly selected as border routers and the remaining nodes are core routers. In practice, each border router may connect with several inter-AS links to adjacent ASes. However, for simplicity, and without loss of generality, we abstract these inter-AS links into one. Thus, each border router is associated with one virtual inter-AS link which can logically represent one or multiple physical inter-AS links. Therefore, 30 virtual inter-AS links are considered and each has capacity of 500 units.

6.2 Bandwidth offer model

It is well known that whilst a typical default-free routing table may contain routes for more than 100,000 prefixes, only a small fraction of prefixes are responsible for a large fraction of the traffic (Feamster et al., 2003). Based on this finding, we consider 100 remote destination prefixes to be included in the bandwidth offers. In fact, each of them may not merely represent an individual prefix but also a group of distinct address prefixes that have the same end-to-end path properties, e.g. geographical location, offering AS and maximum available bandwidth. Hence, the hundred prefixes we considered could reflect an even larger number of prefixes.

In a network, each border router can be an ingress or egress point. Without loss of generality, we consider the network scenario where if a border router receives a bandwidth offer towards destination prefix k from adjacent AS Y, then AS Y cannot inject traffic for k into it. This corresponds to multi-hop traffic (Feldmann et al., 2001) in which the traffic traverses the network instead of being directed to another egress link of the same border router. We adopt this model in order to evaluate the TA objective of total bandwidth consumption in the network. As a result, we cannot assign all the destination prefixes on each border router as bandwidth offers. Instead, at each border router we randomly select half of these hundred destination prefixes as bandwidth offers and the other half as inter-AS traffic. In other words, we set the average number of distinct bandwidth offers advertised at each border router to be half of the number of prefixes. Furthermore, each border router can generate the number of traffic flows towards half of these prefixes that have not been selected for bandwidth offers. We note that this destination prefix generation process is just a best effort attempt to model prefix distribution, as no synthetic model for the actual behavior of prefix distribution in real networks was found in the literature. The remote destination prefixes associated with the bandwidth offers are randomly selected. The maximum capacity of each bandwidth offer is uniformly generated between 100 and 200 units. The charge associated with each bandwidth offer varies according to the simulation scenarios.

6.3 Traffic model

Ingress points and remote destination prefixes of the inter-AS traffic matrix are randomly generated. Previous work has shown that inter-AS traffic is not uniformly distributed (Fang

& Peterson, 1999). According to (Broido et al., 2004)), the AS traffic volumes are top-heavy and can be approximated by a Weibull distribution with shape parameter 0.2-0.3. We therefore generate the inter-AS TM with traffic demand following this distribution with the shape parameter 0.3. As previously mentioned, we do not allow traffic-prefix looping, so that if the AS receives a bandwidth offer towards remote destination prefix k from an adjacent AS, then this adjacent AS cannot inject traffic into the AS for k. The number of inter-AS traffic flows to be considered ranges from 500 to a maximum 1500.

As mentioned in Section 3.1, each inter-AS traffic flow is an aggregate of individual traffic flows that have identical ingress points and remote destination prefixes. Hence, the number of inter-AS traffic flows we considered does not reflect the exact total number of individual traffic flows. Instead, the number could represent more individual traffic flows. We assume that moderate overprovisioning is considered by the IBP and unless specified, f_{over} = 1.25 (i.e. 25% inter-AS bandwidth overprovisioning). Table 2 shows the number of traffic flows, their corresponding traffic volume and overall inter-AS link utilization. Note that the total traffic volume presented in the table has already taken into account the overprovisioning factor.

Number of traffic flows	Total Traffic volume	Overall inter-as egress link utilization (%)
500	4465	30%
625	5578	37%
750	6719	45%
875	7813	52%
1000	8915	60%
1125	10046	67%
1250	11142	74%
1375	12259	82%
1500	13402	90%

Table 2. Inter-AS traffic

6.4 Algorithm parameters

For the IBP's GA parameters, we adopt the suggested values from previous GA research to achieve satisfactory effectiveness and convergence rate of the algorithm (Lin et al., 2003). The population size is 200, the value of H of the heuristic operator (a) is 200 since the IBP problem is highly constrained by two capacity constraints, N_{cd} is set to 50, the probability of mutation is 0.01 and N_{it} is set to 100.

6.5 Evaluation of the IBP algorithms

We compare the performance of our proposed GA described in Section 4.4 with the following alternatives:

Greedy-cost heuristic: The Greedy-cost heuristic sorts all the inter-AS traffic flows in descending order of bandwidth demand and selects one at a time in that order. From the bandwidth offers that have sufficient bandwidth to accommodate the given traffic flow, we select the one which incurs the least IBP cost. The flow is then allocated to this bandwidth offer and its corresponding inter-AS route. This step is repeated for the next traffic flow until all flows have been considered. One can imagine this heuristic might be a conventional algorithm used by INPs to solve the IBP problem.

Greedy-random heuristic: A greedy-random heuristic algorithm is included as a baseline comparison. The random heuristic algorithm is similar to the Greedy-cost heuristic except that the bandwidth offer selection of traffic flows is done at random. It may be viewed as the solution obtained by a trial-and-error or an ad hoc IBP approach.

6.5.1 Evaluation of the Total IBP Cost

The aim of the proposed GA is to achieve better and near-optimal IBP cost in comparison with the alternative algorithms. Hence, the main objective of the evaluation in this section is to quantify the effectiveness of the proposed GA over the alternative algorithms.

Fig. 6 shows the total IBP cost achieved by the Greedy-cost and the GA as a function of inter-AS traffic flows. The performance of the Greedy-random heuristic is not presented in this figure since it has a significant performance gap from the other heuristics. Nevertheless, it is compared to the alternative algorithms in Table 3. The legend in the figure shows the names of the algorithms followed by the percentage of established peering connections as mentioned at the beginning of Section 4.

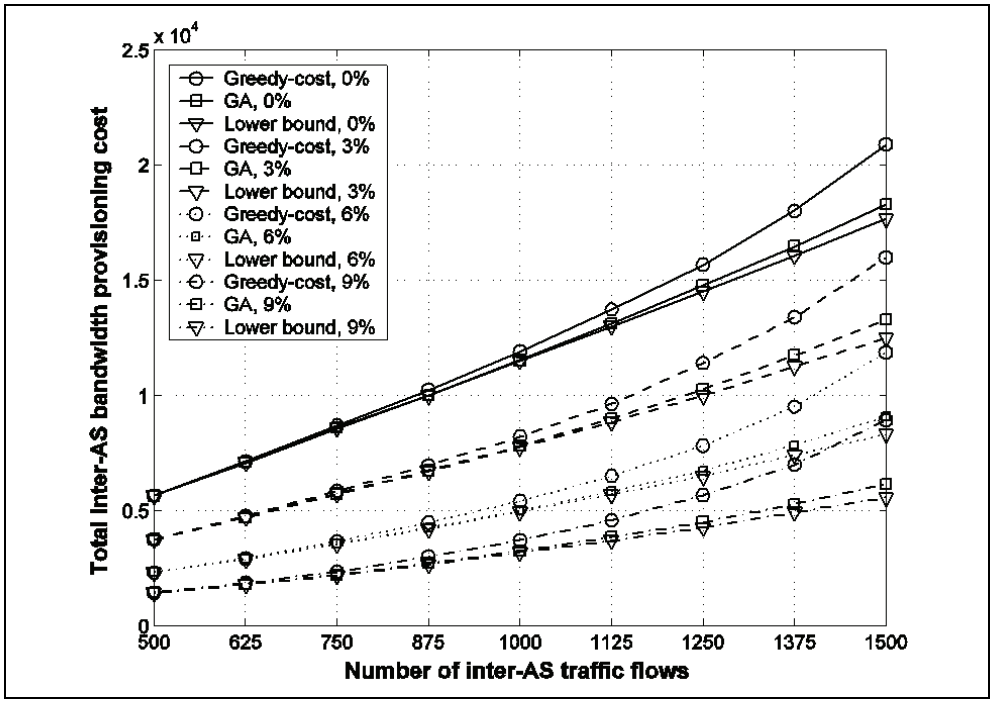


Fig. 6. Evaluation of the total inter-AS bandwidth provisioning cost

The figure presents the results of two practical scenarios, and we evaluate whether the proposed GA performs consistently well under these scenarios. The first scenario consists of all customer-provider connections. In other words, no peering connection (i.e. 0%) is established and the charge of each bandwidth offer is non-zero. We generate an integer uniformly between 1 and 10 to represent each cost. The figure shows that the GA has a lower total IBP cost at all numbers of inter-AS traffic flows. We conjecture that when the

number of inter-AS traffic flows is small, the inter-AS links and the bandwidth offers have relatively plenty of bandwidth to cover all the traffic, and so the GA and the Greedy-cost algorithm would give equivalent IBP results and costs. In contrast, as the number of inter-AS traffic flows increases, both the overall inter-AS link and bandwidth offer utilizations increase and some inter-AS links or bandwidth offers have even reached their capacity limits. In this case, some traffic flows may be assigned to other bandwidth offers which have higher costs. This evaluation shows that a careful selection of bandwidth offers is important in order to minimize the total IBP cost. This can be achieved by the GA.

In addition, the total IBP costs of the GA at all volumes of traffic flows are closer to the lower bound than the Greedy-cost heuristic. This shows that the GA is not only able to achieve a better cost than the Greedy-cost, but also able to achieve a near-optimal cost.

In the second scenario not only are customer-provider connections considered but also peering connections. We evaluate three levels of established peering connections: 3%, 6% and 9% of the total number of bandwidth offers. Simulation data presented in this scenario is as for the previous one except that a designated number of bandwidth offers is randomly selected as peering connections. In current Internet peering practice, most ASes will only accept on a peer link traffic from the peers' customers. Since our purpose is to merely evaluate the performance of the algorithms, we follow the assumption in (Feigenbaum et al., 2002) that general policy routing and peering/transit restrictions are ignored.

Fig. 6 shows that the GA performs better than the Greedy-cost at all degrees of peering connection and all number of inter-AS traffic flows. This is similar to the results of the 0% peering scenario. The GA has better total IBP costs than the Greedy-cost heuristic as the degree of peering connection increases. This is because more and more peering connections do not incur any charges, so that the GA can more effectively utilize the cost-free bandwidth in order to further minimize the total IBP cost. In general, this performance improvement not only applies to the second scenario where some peering connections exist but also applies to the 0% peering scenario where some exceptional low cost bandwidth offers exist.

Number of Inter-AS traffic flows	1000	1125	1250	1375	1500
Over Greedy-cost with 0% peering	3.33	5.0	5.92	8.67	12.75
Over Random with 0% peering	76.16	75.97	75.68	75.6	75
Over Greedy-cost with 3% peering	4.98	6.91	10.13	12.61	17.16
Over Random with 3% peering	83.66	83.08	83.06	81.95	81.38
Over Greedy-cost with 6% peering	7.71	10.6	14.3	18.01	24.0
Over Random with 6% peering	89.22	88.7	88.47	87.67	87
Over Greedy-cost with 9% peering	12.59	16.45	20.96	24.87	31.76
Over Random with 9% peering	92.7	92.41	91.98	91.47	90.85

Table 3. Performance improvement of the GA over the alternative algorithms (in %)

Table 3 shows the relative improvement of the GA over the Greedy-cost and the Greedy-random heuristic algorithms at all numbers of inter-AS traffic flows with different degrees of peering connection. By summarizing the table and considering a reasonably high traffic volume, the proposed GA has approximately 5%-30% and 75%-90% performance improvement over the Greedy-cost and the Greedy-random heuristics respectively under different scenarios. In comparison with the Greedy-random heuristic, the performance of

the GA is remarkable. This shows the importance and value of using systematic approaches, such as the proposed GA, over the trial-and-error and ad hoc approaches.

6.5.2 Evaluation of the proposed GA average running time

In Table 4 we provide the average running time of the GA. The average running time increases as the number of traffic flows increases. We can see that even for quite high numbers of traffic flows the running times are acceptable. These times are perfectly acceptable taking into account the timescale of the provisioning system operation.

Number of traffic flows	Average running time (secs)
500	36.6
1000	78.6
1500	150.4

Table 4. Average running time of the GA

6.5.3 Discussion of the IBP algorithms

The simulation study in this section has evaluated the performance of three IBP algorithms. Simulation results have firstly shown that the proposed GA is efficient and is able to achieve better total IBP cost than the random-based and the conventional heuristic algorithms. The relative total IBP cost improvement achieved by the GA over the Greedy-cost heuristic and the random-based algorithms are great, with 5%-30% and 75%-90% cost savings respectively. We conclude that the IBP solutions obtained by the proposed GA are good overall. This has an implication for INPs that a systematic approach could be developed to optimize the total IBP cost significantly.

6.6 Evaluation of the TA algorithms

The previous section evaluated the performance of the proposed IBP algorithms. Once the IBP phase is completed, an AS performs TA to optimize network resource utilization in order to provide end-to-end bandwidth guarantees for the supported traffic. In this section, we evaluate the performance of our proposed TA algorithms.

We assume that outbound provider SLAs are successfully established in line with the first scenario in the evaluation of IBP algorithms, i.e. the GA IBP outcomes with a linear cost function and all customer-provider connections (0% peering). These outbound provider SLAs are then the input to the TA problem. We consider the following three approaches for the TA problem, namely Cost-only, Cost-Performance and Performance-only approaches. The words "Cost" and "Performance" used in the names of these approaches mean that the ordered priorities of the algorithm optimization targets are on the total IBP cost and the total bandwidth consumption respectively.

Cost-only: Given an IBP solution produced by the GA, there are multiple solutions for assigning traffic to satisfy all the TA problem constraints. Any of these solutions can be selected as the solution of the Cost-only approach since it does not optimize the total bandwidth consumption in the network. We use the Random-TA heuristic algorithm, as shown in Fig. 7, to find a solution for the Cost-only approach.

Random-TA Heuristic Algorithm

Sort inter-AS traffic flows in decreasing order of bandwidth demand
For each traffic flow in that order
- Assign an egress point randomly to the traffic flow
- Establish a bandwidth constrained path between the ingress and egress point
- Update utilized resources

End For

Fig. 7. The random-TA heuristic

Cost-Performance: Given an IBP solution produced by the GA, the Cost-Performance approach takes the proposed greedy-penalty heuristic algorithm as the TA algorithm to optimize the total bandwidth consumption in the network.
Performance-only: The Performance-only approach does not use the IBP solution. Instead, it takes all the bandwidth offers (rather than the outbound provider SLAs) as input and uses the Greedy-penalty heuristic algorithm to solve the TA problem. The total IBP cost is then equal to the sum of the cost of each accepted bandwidth offer. Since the total IBP cost is calculated by taking overprovisioning into consideration, we approximate the total IBP cost of the Performance-only approach by multiplying its solution cost by f_{over} in order to compare it with the total IBP costs achieved by the other two approaches.

6.6.1 Cost vs. performance

We evaluate the proposed three TA approaches. We test the hypothesis that the Greedy-penalty heuristic algorithm can improve the total network bandwidth consumption.

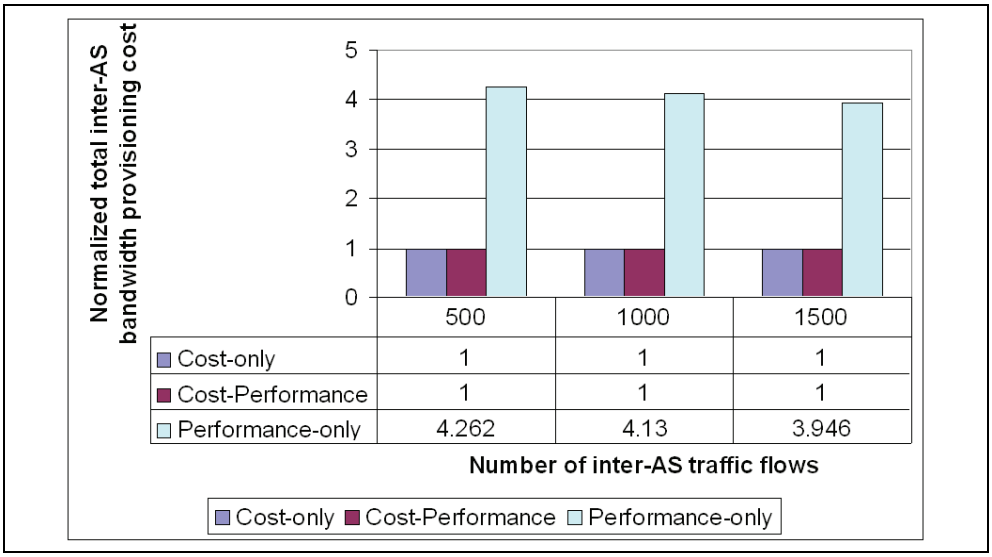


Fig. 8. Normalized total inter-AS bandwidth provisioning cost

Fig. 8 shows the total IBP costs of all the TA approaches at three different volumes of inter-AS traffic flows: 500, 1000 and 1500. The total IBP costs are normalized by the cost of

the solution produced by the GA. The total IBP costs of the Cost-only and the Cost-Performance approaches are identical because they both use the IBP solution produced by the GA. In contrast, the total IBP cost of the Performance-only approach is on average 4 times higher than the others. This significantly higher cost results from neglecting the IBP optimization so that some expensive bandwidth offers are selected, although, as we can see below, using them can significantly improve the total bandwidth consumption in the network.

Indeed, although the Performance-only approach has a very high total IBP cost, Fig. 9 shows that its total bandwidth consumption is approximately half of the other two approaches. Nevertheless, because of its high total IBP cost, the Performance-only approach can be assumed impractical. This implies that there can be conflict between the IBP cost and bandwidth consumption. Therefore, we need a compromising solution that would balance the interests of these two metrics. The Cost-Performance approach attempt to achieve such solution as it has low IBP cost and low total bandwidth consumption compared to the Cost-only approach with the amount closer to the Performance-only approach. This reduced total bandwidth consumption reveals that the proposed Greedy-penalty heuristic algorithm has on average a 10% improvement over the Random-TA heuristic algorithm.

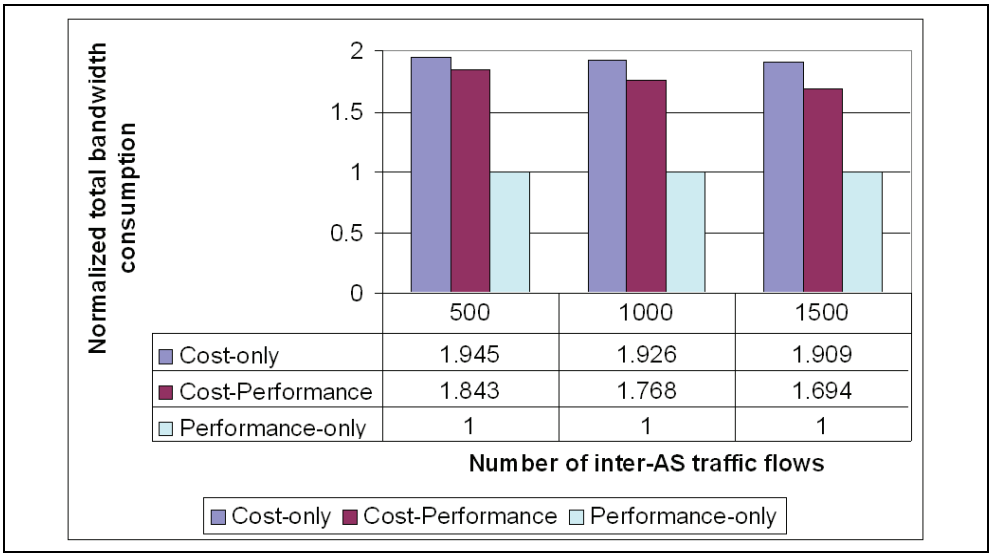


Fig. 9. Normalized total bandwidth consumption in the network

6.6.2 Evaluation of the greedy-penalty heuristic algorithm average running time

Table 5 provides the average running time of the proposed greedy-penalty heuristic algorithm. The average running time increases as the number of traffic flows increases. These running times are perfectly acceptable taking into account the timescale of the provisioning system operation. The computation time could have been much longer if GA was used due to its evolutionary process.

Number of traffic flows	Average running time (secs)
500	6.2
1000	22.1
1500	64.76

Table 5. Average running time of the greedy-penalty heuristic

6.6.3 Discussion of the TA approaches

The simulation described in this section has evaluated the performance of three TA approaches. Simulation results have shown that the proposed Greedy-penalty heuristic algorithm used by the Cost-Performance approach is efficient and is able to achieve on average 10% less total bandwidth consumption than the random-based algorithm used in the Cost-only approach. The performance difference between the Performance-only approach and the other two reveals that a trade-off exists between the IBP and the TA optimization. This trade-off has also been discussed in (Goldenberg et al., 2004) where primarily optimizing monetary cost can degrade network performance and vice versa. However, the determination of relative weights between cost and performance optimizations is far from trivial, particularly when the units of the two metrics have different scales. It is thus in many cases difficult to express in terms of weights the trade-off between the two metrics. Therefore, we assume that from business point of view, an AS considers the IBP cost optimization as more important than the TA performance optimization. Based on this assumption and our simulation study, we conclude that the Cost-Performance approach, which uses our proposed GA and the greedy-penalty heuristic algorithm, performs well both in terms of the total IBP cost and the total bandwidth consumption, in comparison with the Cost-only and the Performance-only approaches.

The Cost-Performance approach can be used by INPs to achieve an effective provisioning of end-to-end bandwidth guarantees. Moreover, since the TA problem has dealt with the selection of inter-AS route and explicit intra-AS route within the network, the Cost-Performance approach could be effectively applied to BGP/MPLS virtual private network provisioning (Rosen & Rekhter, 1999), a subject which is attracting a great deal of attention.

6.6.4 Impact of inter-AS overprovisioning factor on bandwidth consumption

We evaluate the impact of overprovisioning factor on the total bandwidth consumption achieved by the three TA approaches. The results of this evaluation are based on 1500 inter-AS traffic flows. The values of the inter-AS bandwidth overprovisioning factor examined are 1.25, 1.5, 1.75 and 2.0. As the inter-AS available bandwidth increases, the outbound provider SLA capacity constraint becomes less restrictive to the TA problem. Thus, in this case, we expect that the total bandwidth consumption in the network can be further improved.

Fig. 10 shows that the total bandwidth consumption decreases as the overprovisioning factor increases. This is because a large overprovisioning factor reduces the outbound provider SLA capacity constraint and therefore increases the solution space for the TA algorithm, enabling it to find a result with lower total bandwidth consumption. As expected, the Cost-Performance approach has lower total bandwidth consumption than the Cost-only approach at any considered value of the overprovisioning factor. The total bandwidth consumption of the Performance-only approach for all considered values of the

overprovisioning factor is identical because the approach does not consider IBP. Therefore, its performance is not affected by the overprovisioning factor. Fig. 11 shows the normalized total bandwidth consumption achieved by the three TA approaches. As the overprovisioning factor increases, the relative improvement of the Cost-Performance approach over the Cost-only approach slightly increases from approximately 11% to 13%.

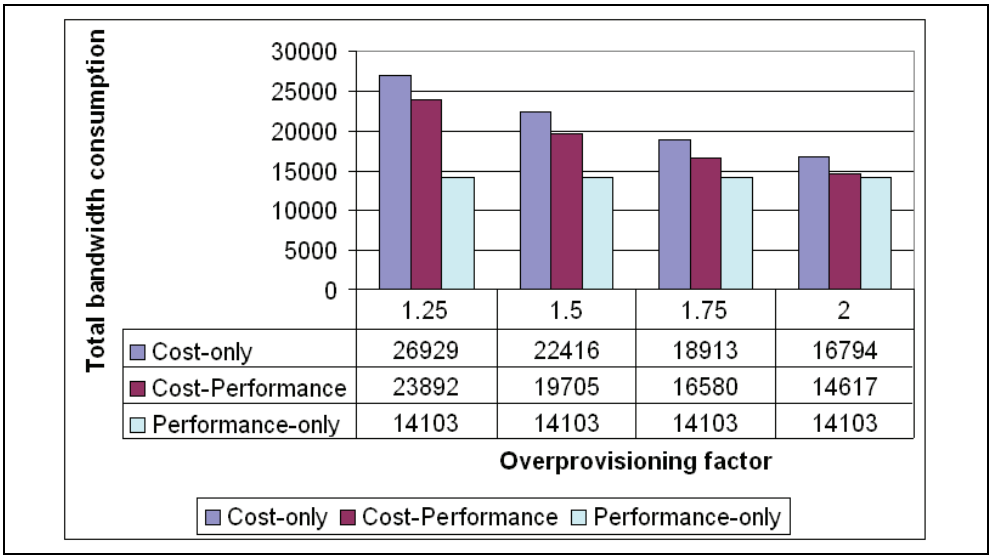


Fig. 10. Total bandwidth consumption achieved by different f_{over}

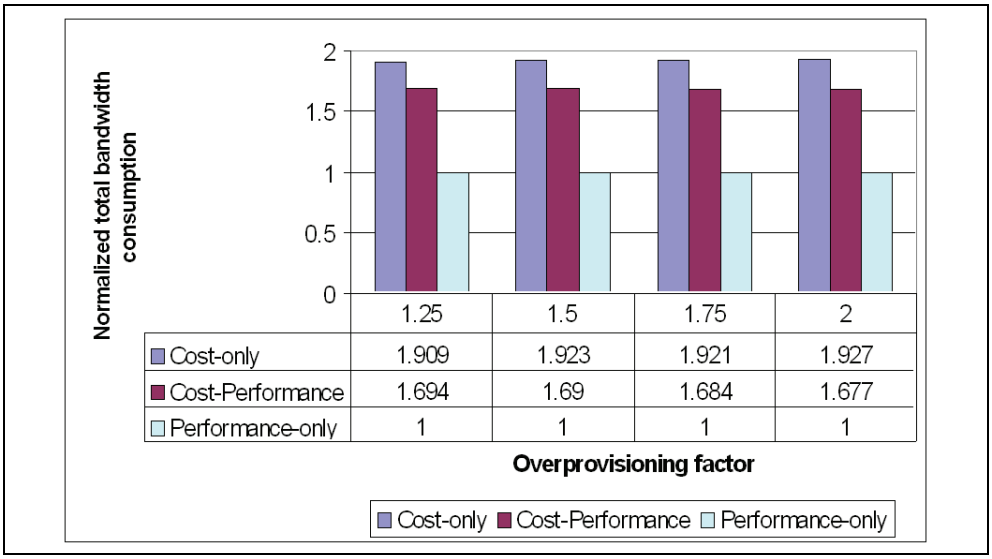


Fig. 11. Normalized total bandwidth consumption achieved by different f_{over}

Fig. 11 also reveals that the performance differences among the three TA approaches are consistent and are insensitive to changes on the overprovisioning factor. The results presented in these figures have revealed the effect of IBP on the TA performance with a different overprovisioning factor. The results confirm our conjecture that as the overprovisioning factor increases, more bandwidth is available in outbound provider SLAs for the TA algorithms to further optimize the total bandwidth consumption.

7. Conclusion

In this chapter we have reviewed a cascaded negotiation model for negotiating and establishing SLAs for bandwidth guarantees between ASes, and a network dimensioning system to solve the inter-AS bandwidth provisioning and the traffic assignment problems systematically.

We formulated the inter-AS bandwidth provisioning problem as an integer programming problem and prove it to be NP-hard. An efficient genetic algorithm was proposed to solve the problem. Our simulation study shows that the genetic algorithm has a near-optimal total inter-AS bandwidth provisioning cost. This cost is approximately 5%-30% and 75%-90% less than the cost achieved by a conventional greedy heuristic algorithm and a random-based algorithm respectively under two customer-peering scenarios.

We formulated the traffic assignment problem as an integer programming problem and prove it to be NP-hard. An efficient greedy-penalty heuristic algorithm was proposed to solve the problem. Our simulation study showed that the greedy-penalty heuristic algorithm achieved on average 10% less total bandwidth consumption than the random-based TA heuristic algorithm.

Finally, we evaluated the effects of different overprovisioning factor values on the total bandwidth consumption. The more the inter-AS bandwidth is overprovisioned, the less the total bandwidth is needed to carry the supported traffic across the network.

A limitation of our work is performance robustness. In case where the derived traffic matrix deviates significantly from the real traffic demands or link failures happen, the performance of IBP and TA may be affected since these network conditions have not been taken into account during the optimization. As future work, we will make the IBP and TA problems robust to traffic demand uncertainty and link failures. Although this may result in trade-offs between performance and robustness, we attempt to achieve good and balance solutions with respect to these two metrics.

8. References

Awduche, D. et al. (2002). Overview and Principles of Internet Traffic Engineering, *IETF RFC 3272*.

BRITE: Boston University Representative Internet Topology Generator. Website http://www.cs.bu.edu/brite/.

Broido, A. et al. (2004). Their Shares: Diversity and Disparity in IP Traffic, *Proceedings of Passive and Active Measurement Workshop*, pp. 113-125, Antibes Juan-les-Pins, April 2004, Springer.

Chu, P.C. & Beasley, J.E. (1997). A Genetic Algorithm for the Generalized Assignment Problem. *Computers and Operations Research*, Vol. 24, No, 1, January, pp. 17-23.

Fang, W. & Peterson, L. (1999). Inter-AS Traffic Patterns and Their Implications, *Proceedings of IEEE GLOBECOM*, pp. 1859-1868, Rio Je Janeiro, December 1999, IEEE.

Feamster, N. et al. (2003). Guidelines for Interdomain Traffic Engineering. *ACM SIGCOMM Computer Communications Review*, Vol. 33, No. 5, October, pp. 19-30.

Feigenbaum, J. et al. (2002). A BGP-based Mechanism for Lowest-Cost Routing, *Proceedings of ACM Symposium on Principles of Distributed Computing*, pp. 173-182, Monterey, July 2002, ACM.

Feldmann, A. et al. (2001). Deriving Traffic Demands for Operational IP Networks: Methodology and Experience. *IEEE/ACM Transactions on Networking*, Vol. 9, No. 3, June, pp. 265-279.

Feldmann, A. et al. (2004). A Methodology for Estimating Interdomain Web Traffic Demand, *Proceedings of ACM IMC*, pp. 322-335, Taormina, October 2004, ACM.

Goldenberg, D.K. et al. (2004). Optimizing Cost and Performance for Multihoming, *Proceedings of ACM SIGCOMM*, pp. 79-92, Portland, August 2004, ACM.

Guerin, R.; Ahmadi, H. & Naghshineh, M. (1991). Equivalent Capacity and Its Applications in High-speed Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 7, September, pp. 968-981.

Holland, J.H. (1975). *Adaptation in Natural and Artificial System*, The University of Michigan Press, ISBN 0262581116.

Howarth, M. et al. (2005). Provisioning for Inter-domain Quality of Service: the MESCAL Approach. *IEEE Communications Magazine*, Vol. 43, No. 6, June, pp. 129-137.

Martello, S. & Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*, Wiley-Interscience, ISBN 0471924202.

Lin, X.H.; Kwok, Y.K. & Lau, V.K.N. (2003). A Genetic Algorithm Based Approach to Route Selection and Capacity Flow Assignment. *Computer Communications*, Vol. 26, No. 9, June, pp. 961-974.

Nucci, A. et al. (2005). Increasing Link Utilization in IP over WDM Networks Using Availability as QoS. *Photonic Network Communication*, Vol. 9, No. 1, January, pp. 55-75.

Osborne, E. & Simha, A. (2002). *Traffic Engineering with MPLS*, Cisco Press, ISBN 1587050315.

Rosen, E. & Rekhter, Y. (1999). BGP/MPLS VPNs, *IETF RFC 2547*.

Shi, S.Y. & Turner, J.S. (2002). Multicast Routing and Bandwidth Dimensioning in Overlay Networks. *IEEE Journal on Selected Areas in Communications*, Vol. 20, No. 8, October, pp. 1444-1455.

Sridharan, A.; Guerin, R. & Diot, C. (2005). Achieving Near-optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks. *IEEE/ACM Transactions on Networking*, Vol. 13, No. 2, April, pp. 234-247.

Teixeira, R. et al. (2005). Traffic Matrix Reloaded: Impact of Routing Changes, *Proceeding of Passive and Active Measurement Workshop*, pp. 251-264, Boston, March/April 2005, Springer.

Zhang, R. et al. (2004). MPLS Inter-Autonomous System Traffic Engineering Requirements, *IETF RFC 4216*.

28

Solving the High School Scheduling Problem Modelled with Constraints Satisfaction using Hybrid Heuristic Algorithms

Ivan Chorbev, Suzana Loskovska, Ivica Dimitrovski and Dragan Mihajlov
Faculty of Electrical Engineering and Information Technologies
Republic of Macedonia

1. Introduction

Constraint Programming is a methodology for problem solving which allows the user to describe data and constraints of the problem without explicitly solving in the declarative phase. Constraint Satisfaction Problems (CSP) can simply be defined as a set of variables and a set of constraints among the values of the variables. Typical method of solving CSP models is building the solution by backtracking approach in which a partial assignment to the variables is incrementally extended, while maintaining feasibility of the current solution. The constraints are kept satisfied throughout the solving process.

Many optimization problems of practical as well as theoretical significance consist of finding "the best" configuration of values for a set of variables. Such problems where the solution is modelled using discrete variables belong to combinatorial optimization (CO). The problems of combinatorial optimization consist of a set of variables, their domains, constraints among variables and a goal function that requires to be optimized. School scheduling is a typical example of a CO problem.

High school schedule generation includes both temporal and spatial scheduling. It is a computation demanding and usually a complex task. It is a NP hard optimization problem that requires a heuristic solving approach (Zhaohui & Lim, 2000).

It is interesting to note that educational institutions rarely use automated tools for schedule generation, although the area has been researched for a long time. A survey in British universities (Zervoudakis 2001) showed that only 21% of the universities use a computer in the generation of exam timetables. Only 37% of the universities use the computer as assistance in the process, while 42% do not use a computer at all. Generation of schedules in some schools in Japan takes up to 100 man hours a year. In bigger schools, schedule generation begins in April and does not end until June, two months after the beginning of the school year, almost 150 work days.

Constraint satisfaction is usually not the first choice for modelling scheduling problems, due to their high complexity. Only the final schedule (hopefully) satisfies all imposed constraints. During schedule generation, most of the constraints will be dissatisfied at some point. We created a system where the extent of constraint satisfaction is measured and compared, so CSP can be successfully used in scheduling (Chorbev et al. 2007). When a measurement of constraint satisfaction is included, the system becomes a Constraint Optimization Problem (COP).

A very general approach to solve combinatorial optimization problems like scheduling is to generate an initial, suboptimal solution and then to apply heuristics to improve the solution. This method is somewhat incompatible with the standard backtracking method in constraints programming. In NP hard problems, building the solution by backtracking approach in which a partial assignment to the variables is incrementally extended is virtually impossible. For example, when the number of queens in the N-Queens problem exceeds certain number (for example 50), the Generalised Arc Consistency – Conflict Based Back-jumping (GAC-CBJ) algorithm fails to give a result in reasonable time (Jolevski et al., 2005b). The duration of the search quickly grows beyond any reasonable amount.

Initially, the aim of our research was to achieve a successful symbiosis of constraint programming and heuristic algorithms. Additionally, we aimed to create hybrid heuristic algorithms that would use the advantages of known algorithms. Therefore, we developed hybrid combinations of different heuristic approaches. Our solving approach begins with an initial suboptimal solution followed by heuristic repair to achieve the final correct solution. Ideas from algorithms like Simulated Annealing, Tabu Search, and Guided Search were incorporated to achieve quicker and more accurate solving. Heuristic algorithms demand a constraint satisfaction system that can measure the level of constraint satisfaction and provide heuristics for solution improvement (Leenen et al. 2003).

Most of the heuristic decisions in our solving process are made in the process of repairing the first suboptimal solution. We had to implement mechanisms to avoid trapping at local optima, avoid deadlocks and achieve convergence. The functions that generate the proper next solution based on the previous one are the key to successful, quick and accurate solving. They use knowledge about the problem and reuse information of the specific inconsistent constraints to generate an improved solution in the neighbourhood of the current one. Exact generation of improved solutions based on previous inconsistencies showed useful up to the moment when a deadlock occurs. A stochastic component in the solving process proved effective in avoiding deadlocks and guiding toward the final solution.

To test and implement the hybrid algorithms and their use over constraint modelled problems, a broader software framework was necessary. Therefore we developed and implemented a universal Constraint Solving Engine (CSE) and a Constraint Programming Library (CPL). We developed a set of constraint types for modelling different problem types and a mechanism for selection an optimal algorithm for the given problem. CPL contains different algorithms, including Simulated Annealing, Tabu search, Arc consistency etc., as well as their hybrids.

This approach offers several advantages. The tool can be applied on different problems by cost of very little to no further programming at all. For the user it is only necessary to model a new problem with given constraint types, choose an algorithm, and initiate the solution process. The engine can be easily implemented in any commercial problem solving software. Our solving engine has substantial theoretical implications, too. The use of object-oriented approach provides a mechanism for adding and testing new algorithms based on the same problem description. This system enables comparing efficiency, and results of different algorithms as well.

Part 2 of the chapter gives an overview of Constraint Programming and Constraint Optimization where the origins, the definitions and the basic concepts are explained. Part 3 of the chapter gives an overview of the concept of hybridization of heuristic algorithms. Some strategies are explained and examples are given. Part 4 of the chapter gives a short

description of the constraint solving engine with multiple optimization algorithms that we developed and used for simulations. Part 5 gives the model of the high school scheduling problem expressed in terms of mathematical constraints. The hybrid heuristic algorithm that we developed is explained in part 6 of the chapter. Finally, part 7 of the chapter contains the closing remarks and ideas for future work.

2. Constraint programming and constraint optimization

During the seventies of the 20th century, David Waltz within one of his algorithms set the basic concept of the technique of Constraint Propagation (Kumar, 1992). Ever since, the concept has evolved surpassing the boundaries of artificial intelligence and affecting wide range of research areas. Today, an increasing number of explorers in the area of programming logic, knowledge representation, expert systems, theoretical computer science, operational research, and other similar fields explore the use of constraint programming techniques, both as theoretical basis as well as true practical applications. In time, it is understood that constraint satisfaction is the main problem of a wasp area of problems like time reasoning, spatial planning, configuration planning, timetable generation, telecommunications, even in databases (Der-Rong & Tseng, 2001). The main reason for the increased interest and success of constraints processing techniques is their ability for good declarative formulation of problems as well as efficient solving (Meyer, 1994).
A constraint is simply a logical relation among several unknowns (or variables), each taking a value in a given domain (Bartak, 1999). More formal definition states:

Definition 1: (Gavanelli, 2002)
A Constraint Satisfaction Problem (CSP) is a triple P = {X, D, C} where:
$X= \{X_1, X_2, ..., X_n\}$ is a set of unknown variables,
$D = \{D_1, D_2, ..., D_n\}$ is a set of domains and
$C = \{c_1, c_2, ..., c_n\}$ is a set of constraints.
Each $c_1(X_{i1}, ..., X_{ik})$ is a relation, i.e., a subset of the Cartesian product $D_{i1} \times \cdots \times D_{ik}$.
An assignment $A=\{X_1->d_1, \ldots, X_n->d_n\}$ (where $d_1 \in D_1, \ldots, d_1 \in D_{n1}$) is a solution if it satisfies all constraints.

In the declaration phase of Constraint Programming, the user describes the data and the constraints of the problem without explicitly solving it. When using constraints, the problems can simply be defined with a set of variables and a set of constraints. The constraints specify a certain relation over a subset of variables. The relations limit the values that the variable can have. Different constraints engage different variables making a network of constraints. The problem that requires to be solved is finding a relation over the entire network of variables that simultaneously satisfies all constraints. The derived problem type is named Constraint Satisfaction Problem – CSP. This methodology is perfectly suited for schedule generation, since the entities engaged can be defined and the expected correct schedule can be declaratively expressed. If the object-oriented approach is added, the result will be general, in the same time having the possibilities for exact specialization.
Constraint programming is a term close to mathematical programming (Sedgewick, 1983). Mathematical programmes contain a set of variables interconnected by a set of mathematical equations called constraints and an objective function that calculates the quality of the solution represented by certain combination of values for the variables. If all equations are

only linear combinations of variables, the problem is a special case named linear programming.
After the problem is modelled with constraints, the state space derived from the variable domain and the constraints requires to be searched for the best solution. The algorithms for searching the state space are a key phase in the solving process.
Constraint Optimization Problems (COP), also known as Constraint Relaxation Problem – CRP (Yoshikawa, 1996), can be defined as common problems of constraint satisfaction in which the level of satisfaction of every constraint can be measured. The goal is to find a solution that maximises the sum of constraint satisfactions. Also, constraint optimization problems can be defined as constraint satisfaction problems upgraded with several local cost functions. The goal of the optimization is finding a solution whose cost, evaluated as a sum of all cost functions, should be maximal or minimal. Regular constraints are called hard, while cost functions are known as soft constraints. Names illustrate that hard constraints must be satisfied, while the soft ones only express preferability toward some solutions.

3. Metaheuristic algorithms and their hybridization

In recent decades we have witnessed the development of a new kind of approximative algorithms that combine basic heuristic methods in frameworks designed for efficient and effective search of the state space. These methods are named metaheuristics. The term was suggested by Glover in 1986, based on the ancient words: "heuristic" meaning "to discover", and the prefix "meta" meaning "above, higher level". These groups include, among others: Ant Colony Optimization (ACO), Evolutionary Computation (EC) – like Genetic Algorithms (GA), Iterated Local Search (ILS), Simulated Annealing (SA), Tabu Search (TS), Brute-force search, Random optimization, Local search, Greedy algorithm, hill-climbing, Random-restart hill climbing, Greedy best-first search, Branch and bound, Swarm intelligence - Ant colony optimization, Greedy Randomized Adaptive Search Procedure – GRASP etc. There is no strict definition for what metaheuristic is, but main axioms found in literature state that metaheuristics are a group of strategies that guide the search process. They search for an optimal or near optimal solution aproximatively and non-deterministically (Blum & Roli 2003).
The combination of different heuristic can be done in several ways. Various heuristic methods can be chronologically applied in different phases of the search, when their advantages are required the most. Besides chronological sequential application of different search methods, the algorithms themselves can be a hybrid of more metaheuristic or basic optimization approaches.
There are various forms of hybridization of algorithms. The first form advocates integration of components from one metaheuristic into another. The second form includes systems known as cooperative search. They are consisted of various algorithms that exchange information. The third option is integration of approximative and systematic (complete) methods. By emphasizing the advantages and flaws of different metaheuristic approaches, it is evident that hybridization and integration of different heuristic algorithms might result in better solutions to problems.
Since schedule generation is a NP hard problem, methods for exhaustive search are not an option. Algorithms like Tabu search, Genetic Algorithms and Simulated Annealing have been previously applied to such problems. Although they all have advantages, no one solves the problem completely. The goal of our research was to implement combinations of algorithms.

3.1 Exchange of components between metaheuristics

A popular way of hybridization is the use of trajectory methods with populations based methods (Blum et al., 2005). The most successful applications of evolutionary algorithms and ant-colony optimization use procedures for local search. The reasons are obvious when the appropriate advantages of trajectory and population methods are analysed.

The power of population methods is based on recombining solutions to derive new ones. Evolutionary algorithms and Scatter search implement explicit recombination with one or more operators (Glover et al., 2003). In Ant-colony optimization and some evolutions algorithms, the recombination is implicit, because new solutions are generated by using a distribution in the state-space, based on the previous populations. This allows guided steps in the search space that are usually bigger than steps made in trajectory methods. That means the solution based on recombination in population methods is more "different" from the parents as opposed to a solution derived with one move from the previous solution. There can also be "big steps" in trajectory methods, like iterated local search and variable neighbourhood search, but, in these methods the steps are not guided (these steps are called trials or perturbations to emphasise the lack of guidance). In every population based method, there are mechanisms that use the good solutions that have been found to influence the search to find even better solutions. The idea has been explicitly implemented in the Path Relinking algorithm (Blum et al., 2005). There, the basic elements are initial solutions and guiding solutions (the best found so far). New solutions are derived by applying moves to decrease the distance between the resulting and the guiding solution. Evolution algorithms achieve the same effect by keeping the best found solutions so far in the population. The approach is called an evolution process with stable states. Scatter search performs a process with stable states. In some implementations of ant colony optimization, there is a schedule for updating the pheromones that uses only the best found solution when the algorithm converges toward the end. It corresponds with changing the direction of the search process toward a good solution hoping to find better on the way.

The power of trajectory methods is the way that they search the promising regions in the state space. Since local search is the main component, a promising part of the search space is searched in a more structural way than in population based methods. This approach reduces the possibility of missing the optimal solution when the search is near it, as opposed to population methods. The conclusion is that population methods are better in identifying promising regions in the search space, while trajectory methods are better in exploring the promising area. Therefore, metaheuristic methods that combine the advantages of population and trajectory methods are successful.

3.2 Cooperative search

A loose form of hybridization is achieved in joint search (Hogg & Huberman, 1993) which consists of searching by various algorithms that exchange information for states, models, entire sub problems, solutions or other specifics of the search space. Usually, the solving process is based on parallel execution of algorithms with different level of communication. The algorithms may be entirely different, or instances of the same algorithm functioning on different models or different configuration parameters. The algorithms that make the joint search can be aproximative or complete, or a mixture of aproximative and complete methods. Cooperative search receives increased interest because of the interest in parallelisation of metaheuristic.

3.3 Integration of metaheuristic and systematic methods

The approach of integration of metaheuristic and systematic methods is quite effective when used over practical problems. The discussion about similarities, differences and possible integration of metaheuristic and systematic methods can be found in (Glover & Laguna, 1997). Recent research papers suggest that integration of metaheuristic and constraint programming is proving especially useful and successful. (Duong & Lam, 2004), (Gomes et al., 2005), (Crawford et al., 2007)

3.4 Hybridization with parallelization

Some of the heuristic algorithms are inherently easily executed in parallel, while others require sophisticated strategies for parallel execution. Generally, genetic algorithms (GA) are easy to execute in parallel, while SA is sequential by nature. On the other hand, there is a mathematical proof that SA slowly, but surely converges toward the final solution. Since there is no such proof for the GA, a hybrid SA with operators from genetic algorithms is a good approach.

There are various Parallel Genetic Simulated Annealing Algorithms – HGSA. In literature [Ohlidal 2004] there are a couple of published versions:

- S. W. Mahfoud and D. E. Goldberg suggest a concept of a GA using a Metropolis algorithm in the selection process.
- M. Krajic describes a hybrid parallel SA based on genetic operators (mutation and cross-reference).
- N. Mori, J. Yoshida and H. Kita use a thermodynamic rule for selection.

Czech describes a parallel implementation of SA without GA. (Czech et al., 2006)

3.4.1 Parallel SA with a Boltzmann synchronization function

Within our research we experimented with parallel execution of SA and developed parallel SA with a Boltzmann synchronization function. (Chorbev et al., 2006)

The cooperation of more processors can be used either to speed up the sequential annealing algorithm or to achieve a higher accuracy of solutions to a problem. In this work we considered both goals. The accuracy of a solution is meant as its proximity to the global optimum solution.

We designed a system with *r* available processors and each of them is capable of generating its own annealing process. The architecture includes a master computer and given number of slave computers, interconnected in a Local Area Network. The starting - master computer P_1 imports the initialization data, generates the first proposed solution and passes data to *r-1* remaining computers. All remaining computers – processors start independent annealing process after receiving the initial data. All processors communicate by exchanging current best solutions during annealing processes, at a chosen rate. The scheme of communication is given in the figure 1.

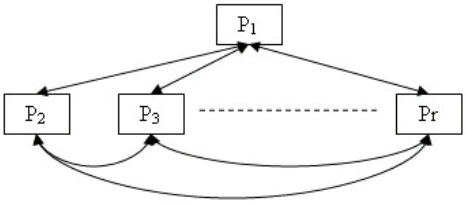


Fig. 1. Processor communication

The communication model used is synchronized point-to point. Before the temperature is decreased, every process sends its best found solution to the remaining *r*-2 processes, and waits the best found solutions from all other processes, too. Once all data is received, each process calls its acceptance function to decide whether to accept the best solution from all other processes or continue with the one found by the process itself. With this architecture, the master computer only starts the solving process and eventually, collects best solutions from slave computers. It serves no purpose during solving iterations and information exchange; therefore its functions could be performed by some of the slaves. Keeping master's functions limited excludes it being a bottleneck in the architecture.
We analyzed a possible problem of certain faster converging processes to wait for slower processes to send their best solutions. This architecture is only as fast as the slowest of the included computers. However, we consider this not to be a setback. All computers used in the network are of same type, design and performance. Also, all computers execute the same annealing algorithm; use the same temperature decrement coefficient, the same number of iterations during each temperature and the same metropolis function. The only difference is the independent random generation of the next proposed solution in every computer. This provides different search paths through the solution space in every parallel process and increase diversity of the search. Therefore, all computers are expected to make at average the same number of acceptance and declination of new proposed solutions (due to the metropolis function). The cumulative result is roughly the same computational effort (time of execution) in each computer. Sometimes some processors might converge faster toward a local optimum, but the necessary broad search of the domain that this parallel architecture brings is worth waiting.
The acceptance function (the decision in every processor to accept the best solution from others or continue with its own) was also a subject of interest in our research. We tried: always accepting the best solution from all others, randomly accepting any of the given solutions from other processes and eventually accepting solutions using the Boltzmann distribution. We got the best results using the Boltzmann distribution. This probability function is fundamental for SA and it seems natural for it to be part of SA's parallelization.
There are other points among the algorithm steps where parallel processes could communicate, i.e. different rates of communication. Data could be exchanged within the inner annealing iteration at every n^{th} iteration or after certain number of temperature decreasing iterations. In our parallel SA the processes P_2, P_3, …, P_r cooperate among each other at every temperature decreasing iteration.
Implementation of the parallel SA is the following:

Process P_0:
INITIALIZE;
Dispatch initial solution to processes Pp, p=2, 3, …, r
Wait until final solutions from processes Pp, p=2, 3,…, r are received
Choose and display the best solution from processes Pp, p=2, 3,…, r
Process Pp, p=2, 3,…, r:
INITIALIZE;//receive initially proposed solution
repeat
    repeat
        PERTUB(solution(i) -> solution(j), Δcostij);
        if METROPOLIS(Δcostij) **then** accept

```
            if accept then UPDATE(solution(j));
        until predefined number of iterations;
        Send current solution s(p) to other processes Pq,  q=2, 3,…, r, q≠p
        Receive solutions from all proc. Pq, q=2, 3,…, r, q≠p
        Choose the best solution s(q) from received solutions
        if exp(-Δcost_pq/TEMP) > random[0; 1) //Boltzmann
        then accept;
        if accept then UPDATE (solution s(j));
        TEMP_+1 = f (TEMP); //Decrease temperature
until stop criterion = true (system is frozen);
```

A crucial component when designing a parallel algorithm is finding the best tradeoff between the amount of communication and every processor's independence. Communication of the parallel processes within the inner annealing cycle causes extensive communication slowing the overall performance. On the other hand, delaying the communication for every n^{th} temperature iteration gave worse solution quality because of lack of sufficient information exchange. The experimental results given in figure 2 show that best results are attained when communicating at every temperature iteration. This graph is generated with 5, 10 and 20 processors.

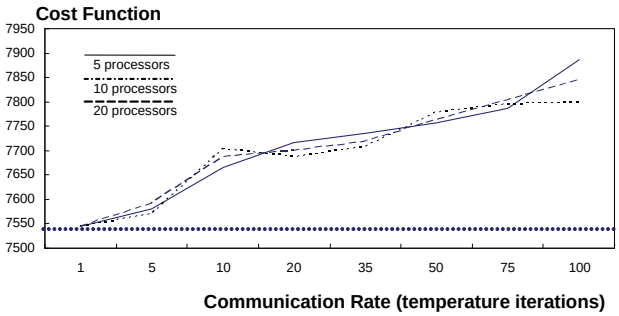


Fig. 2. Course of solution quality versus the rate of communication among processes. The horizontal axis is the number of temperature iterations between the processes communication. The vertical axis is the solution cost. The dashed line is the optimal solution

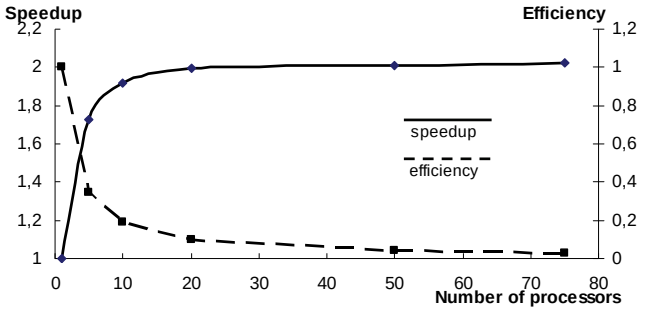


Fig. 3. Course of speedup and efficiency versus the number of processors.

Besides increasing solution quality, main reason for parallelization is the expected speedup. Speedup is defined as the ratio of solving time using single processor versus multiple parallel processors solving time. Efficiency is defined as the ratio of speedup versus the number of processors used. Efficiency gives the utilization of the processors. According to experimental results in figure 3, the speedup is obviously increasing when going from one processor toward five or ten. Further increasing of the number of processors brings no advantage since large amount of communication among processes slows the overall performance. According to experimental results, our parallel SA implementation achieves best speedup at 10 – 20 processors. However, if we take the efficiency into consideration, using more than 10 processors is highly inefficient.

4. Constraint solving engine with multiple optimization algorithms

The research presented in this chapter is performed using a Constraint Programming Library (CPL) (Jolevski et al. 2005a). The software library is consisted of a set of classes – generic constraint types for modeling different problem types and a mechanism for selection an optimal algorithm for the given problem. This approach is required because the Constraint Solving Engine (CSE) is developed to enable solution of problems with different nature. The engine is modular, allowing specific heuristics for certain problems to be implemented in overridden functions. Every step of the problem solving process could be implemented either with existing components or with newly added modules overriding those already contained in the basic object-oriented system.

The CSE is based on the concept of variables and their domains. The domains are bounded by the existing constraints in the moment of their creation, making the search space smaller. Later in the process of proposing new solutions, the constraints evaluate the extent of satisfaction and measure the progress toward the best (final) solution.

The main constraint class provides an integrated interface to all its children. The inherited interface enables algorithms to use the constraints, gives them their variables and checks the consistency of conditions. The constraints return Boolean or in some cases a quantitative measure of the constraint satisfaction.

In our model, the solution cost originates from the level of satisfaction of every constraint. Every constraint has an implemented function for calculation of the amount of its dissatisfaction. Additional multipliers to the dissatisfaction levels exist, to increase the influence of certain constraint over others in the total cost.

The set of given constraints is appropriate for modeling different problems. That is so because most of the problems can be divided into smaller and simpler ones that later can be modeled and solved. From mathematical point of view a broad variety of common, appearing different from the outside, problems are turned into the same or similar tasks.

When modeling a problem it can always be expressed through a mathematical language. Very often, the problem can be expressed as a couple of arrays of integer values that comply with certain rules. For the user, those model arrays are converted into understandable solution data like the shortest path for a traveling salesman or into the most optimal high school schedule. In the background, in the mathematical model, the problem rules transform into constraints like: "no two elements of the array can have the same value" or "the sum of all elements of the array must always be a constant value." All rules and value checks are done by methods within the constraints classes.

5. Constraints modelling of the high school scheduling problem

Scheduling covers a wide area of problems with temporal and spatial distribution of resources. Three broad families of scheduling problems can be distinguished depending on the degrees of freedom in positioning resource supply and resource demand intervals in time (Abramson, 1991): pure scheduling problems, pure resource allocation and finally, joint scheduling and resource allocation problems. High School scheduling is a composite problem.

In case of School Scheduling, the model includes means for temporal and spatial distribution of resources. It is required to implement priorities among constraints, providing methods for satisfying primarily more important rules followed by less significant.

The main interest of constraint programming lies in actively using the constraints to reduce the computational effort required to solve a problem, in the same time achieving good declarative problem formulation. Constraints are used not only to test the validity of a solution, but also in a constructive mode to deduce new constraints and detect inconsistencies. This process is called constraint propagation.

This problem domain falls within the category of Constraint Optimization Problems (COP), where the constraint(s) satisfaction requires to be evaluated (in opposition to those problems where they can only be satisfied or unsatisfied, called constraint satisfaction problems, CSP) (Penya et al., 2005). Application of algorithms like Simulated Annealing (SA) demands a solution cost function that the algorithm will tend to decrease (Leenen et al., 2003). Therefore, the implemented constraints of the model are capable of producing a numerical measurement of their satisfaction. Abramson (Abramson, 1991) in his model separates the total cost to three parts: teacher cost, class and room cost as a result of clashes in the trial solutions on those three bases.

Schedule generation has been formalized as a problem of optimizing constraints, or Constraint Relaxation Problem – CRP by (Yoshikawa et al., 1996). They focused on using the min-conflict heuristics to generate an initial solution for solving both school and university timetabling problems. After a fairly good-quality initial solution is generated by an Arc-Consistency algorithm, their proposal relies on a heuristic billiard-move operator to iteratively repair the current solution and complete assignment of lessons for school/university timetabling. The min-conflicts heuristic (MCH) tries to examine each variable to assign a value with the minimum number of constraint violations. Tam and Ting (Tam & Ting, 2003) combine the min-conflicts and look-forward heuristics used in local search methods to effectively solve general university timetabling problems.

5.1 Notation

The problem in our case is modeled as follows: There are G·D·N (G – Groups, D – Days, N – lessons per day) variables (items) that define the assignment of lessons to groups and rooms. Variables are grouped in blocks of D·N variables. Every block corresponds to the timetable for one group.

Let's denote the set of all lessons in a timetable by $T = \{t_0, t_1, \ldots, t_{T-1}\}$, where $T = |T| = G \cdot D \cdot N$ is the number of lessons in a timetable. Lessons are grouped in blocks of N lessons that are all in the same day. Lessons are ordered in an increasing day order.

The next stage, before actually turning the constraints into program code, is creating the mathematical model. For that purpose, an exact notation was required, part of which is defined as follows:

$\mathbf{\Pi} = \{\pi_0, \pi_1, \ldots, \pi_{P-1}\}$ set of teachers;
$\mathbf{B} = \{\beta_0, \beta_1, \ldots, \beta_{B-1}\}$ set of subjects;
$\mathbf{\Psi} = \{\psi_0, \psi_1, \ldots, \psi_{Y-1}\}$ set of school years;
$\mathbf{\Delta} = \{\delta_0=0, \delta_1=1, \ldots, \delta_D=D-1\}$ set of working days;
$\mathbf{\Gamma} = \{\gamma_0, \gamma_1, \ldots, \gamma_{G-1}\}$ set of groups;
$\mathbf{X} = \{\chi_0, \chi_1, \ldots, \chi_{C-1}\}$ set of rooms;
$\mathbf{I} = \{I_{min}, I_{min}+1, \ldots, N\}$ set of number of lessons in a day;
I_{min} minimum number of lessons in a day; and
N maximum number of lessons in a day.

5.2 Data

The algorithm works using existing data. The data has to be previously entered in the program (in the relational database through an intuitive user interface of the scheduling software (Jolevski et al., 2005d)), and prepared in the specified format. Certain data is exploitive in more than one constraint. Our model contains eight previously entered data structures. They are:

1. Number of weekly lessons x per subject β per group γ is defined by the following set of ordered triples: $\mathbf{B\Gamma} = \{(\beta_i, \gamma_j, x) \mid i = 0, \ldots, B\text{-}1; j = 0, \ldots, G\text{-}1\}$.
   Function $x = X_{\beta\gamma}(\beta, \gamma)$: $\mathbf{B\times\Gamma} \rightarrow \mathbf{Z}_+$ returns the required number of lessons for subject β for group γ.
   Function $x = w_\gamma(\gamma)$: $\mathbf{\Gamma} \rightarrow \mathbf{Z}_+$ returns the required number of weekly lessons for group γ.
2. All combinations of groups and subjects that a particular teacher can teach are given in the following set. Teacher π who teaches subject β for group γ is defined by the following set of ordered triples: $\mathbf{\Pi B\Gamma} = \{(\pi_k, \beta_i, \gamma_l) = \mid k = 0, \ldots, P\text{-}1; i = 0, \ldots, B\text{-}1; l = 0, \ldots, G\text{-}1\}$.
   Function $\pi = P_{\beta\gamma}(\beta, \gamma)$: $\mathbf{B\times\Gamma} \rightarrow \mathbf{\Pi}$ returns the teacher π for subject β for group γ.
3. Subjects β that can be taught in a room χ is defined by the following set of ordered pairs: $\mathbf{BX} = \{(\beta_i, \chi_j) \mid i = 0, \ldots, B\text{-}1; j = 0, \ldots, C\text{-}1\}$.
   Function $T_{\beta\chi}(\beta, \chi)$: $\mathbf{B\times X} \rightarrow \{0, 1\}$ returns 1 if subject β can be taught in room χ, otherwise returns 0.
4. Maximum number m of groups that can share a room χ is defined by the following set of ordered pairs:
   $\mathbf{X}_M = \{(\chi_i, m) \mid i = 0, \ldots, C\text{-}1\}$.
   Function $M_\chi(\chi)$: $\mathbf{X} \rightarrow \mathbf{Z}_+$ returns the maximum number of groups that can share room χ at any point in time.
5. If a subject β should be taught in blocks of 2 consecutive lessons, than there is an appropriate element in the set $\boldsymbol{C} = \{\beta \mid \beta \in \boldsymbol{B}\}$
   The function $c_\beta(\beta)$: $\boldsymbol{B} \rightarrow \{0,1\}$ returns 1 if the subject β can be taught in blocks of 2 lessons, because $\beta \in \boldsymbol{C}$, else returns 0 meaning $\beta \notin \boldsymbol{C}$.
6. The availability of the teacher in a particular time slot during the week is kept in this set. A teacher π_i available to teach a class a on a day δ_j in a shift μ_k is defined with the following set:

$$\boldsymbol{A} = \{(\pi_i, \delta_j, \mu_k, a) \mid i = 0, \ldots, P - 1; j = 0, \ldots, D - 1; k = 0, \ldots, M - 1\}$$

If elements exist for a professor in the set $\boldsymbol{\Pi\Delta M}$, then these elements hold the availability of the teacher. If there is no element for the particular teacher in the set $\boldsymbol{\Pi\Delta M}$ the teacher is available at any time in the week.

The function $a_\alpha(\pi_i, \delta_j, \mu_k, a)$: $\boldsymbol{A} \rightarrow \{0,1\}$ returns 1 if the teacher π_i is available to teach the subject a in the day δ_j in shift μ_k , that is $(\pi_i, \delta_j, \mu_k, a) \in \boldsymbol{A}$, else returns 0 meaning $(\pi_i, \delta_j, \mu_k, a) \notin \boldsymbol{BA}$.

7. The subject β can not be taught in a lesson j:

$$\boldsymbol{BA} = \{(\beta, j) \mid \beta \in \boldsymbol{B}; j \in \{1, \ldots, N\}\}.$$

The function $B_\alpha(\beta, j)$: $\boldsymbol{B} \rightarrow \{0,1\}$ returns 1 if the subject β can be taught in the lesson j, meaning $(\beta, j) \notin \boldsymbol{BA}$, else returns 0 meaning $(\beta, j) \in \boldsymbol{BA}$.

8. The set of elective subjects is $\boldsymbol{E} = \{\beta \mid \beta \in \boldsymbol{B}\}$.

The function $e_\beta(\beta)$: $\boldsymbol{B} \rightarrow \{0,1\}$ returns 1 if β is elective subject or $\beta \in \boldsymbol{E}$, else returns 0 meaning $\beta \notin \boldsymbol{E}$.

5.3 Constraints

The problem is modeled by representing it through 16 constraints. They are defined as follows:

1. $\sum_{k=i}^{i+D\cdot N-1} (b_\tau(\tau_k) = \beta_j) = X_{\beta\gamma}(\beta_j, \gamma_i)$ for $i = 0, D\cdot N, 2\cdot D\cdot N, \ldots, (G\text{-}1)\cdot D\cdot N$ and $j = 1, 2, \ldots, B$, where the sum represents the number of weekly lessons for subject β_j in group γ_i. The number of weekly lessons per subject in a group is defined by a set in the entered data.

When this constraint was coded with the given tools in the Constraint Solving Engine, in the implementation, the generic constraint class CSetCover was used. The generic constraint classes are developed universally so that they could be used in different forms to express different real problem constraints. The CSetCover constraint class, as all other in the CSL, inherits from the basic constraint class (Chorbev et al., 2007). Its task is to check the number of occurrences of a certain value for a given variable coordinate in the array of variables (the "subject" value of the complex time slot variable in this case). When the number of occurrences of the given value in the variable is adequate, the constraint is "covered". The set of values to be covered was equal to the set of courses **B**. Every set value requires to be covered exactly $X_{\beta\gamma}(\beta,\gamma)$ times. Since the set to be covered can be different for different groups, a separate instance of the CSetCover class is necessary to be created for every group.

2. $\sum_{k=i}^{i+D-1} (l_\nu(\nu_k) - f_\nu(\nu_k) + 1) = w_\gamma(\gamma_m)$ for $i = 0, D, 2\cdot D, \ldots, (G\text{-}1)\cdot D$, m = i/D, where the sum is the number of weekly lessons for group γ_m. Number of weekly lessons per group is defined by a set in the entered data.
3. $T_{\beta\chi}(b_\tau(\tau_i), r_\tau(\tau_i)) = 1$ for $i = 0, 1, 2, \ldots, G\cdot D\cdot N\text{-}1$

Subjects are always taught in appropriate rooms as defined by the input data.

4. $a_\tau(\tau_j) = 0$ for $i \le j < i+f$ and $i+l+1 < j \le i+N\text{-}1$ for $k = 0, 1, \ldots, G\text{-}1$ and $l = 0, 1, \ldots, D\text{-}1$, $i = k\cdot D\cdot N + l\cdot N$, where $n = k\cdot D+l$, $f = f_\nu(\nu_n)$, $l = l_\nu(\nu_n)$. There can be no timetable breaks. Empty lessons are determined by variables ν_n.
5. $a_\tau(\tau_j) = 1$ for $i+f \le j \le i+l$ for for $k = 0, 1, \ldots, G\text{-}1$ and $l = 0, 1, \ldots, D\text{-}1$, $i = k\cdot D\cdot N + l\cdot N$, where $n = k\cdot D+l$, $f = f_\nu(\nu_n)$, $l = l_\nu(\nu_n)$. There can be no timetable breaks. Non-empty lessons are determined by variables ν_n.

6. $\sum_{i=1}^{G} \begin{cases} 0 & if \quad r_\tau(\tau_{(i-1)\cdot D\cdot N+l}) \neq \chi_j \\ 1 & if \quad r_\tau(\tau_{(i-1)\cdot D\cdot N+l}) = \chi_j \end{cases} \leq M_\chi(\chi_j)$ for $j = 1, 2, \ldots, C$ and $l = 1, 2, 3, \ldots, D\cdot N$

   At any point in time l, room χ_j can have at most $M_\chi(\chi_j)$ groups as defined by the set $\mathbf{X}_M$ in the input entered data.

   As it was described in the previously published material (Chorbev et al., 2007, Jolevski et al., 2005b) the constraints classes implemented in the Constraint Solving Library (CSL) were created to be universal and applicable to different problems. Therefore, in this constraint the class CSetCover (generic constraint) was used. The set of values to be covered was $\mathbf{X}_M$. In the constraint-variable network, practically the model of the problem, there were $D\cdot N$ copies of this constraint. There was a copy for each time slot (lesson) in the work week. However, every copy is in fact the same instance of the generic CSetCover constraint, since the set of values to be covered was always $\mathbf{X}_M$. In every different copy, the classroom coordinate of the variables for different groups for that lesson was taken in consideration.

7. $b_\tau(\tau_i) \in \mathbf{A} \wedge i \neq l_\nu(\nu_k) \wedge (b_\tau(\tau_i) \neq b_\tau(\tau_{i-1}) \vee i = f_\nu(\nu_k)) \Rightarrow b_\tau(\tau_{i+1}) = b_\tau(\tau_i)$ for $i = 0, 1, \ldots, G\cdot D\cdot N\text{-}1$, and $k = i$ div N.

   If subject $b_\tau(\tau_i)$ must be taught in blocks of 2 lessons, and τ_i is not the last lesson in the day k, and subject $b_\tau(\tau_i)$ is different from the previous subject $b_\tau(\tau_{i-1})$ or τ_i is the first lesson in a day, then the next lesson needs to be for the same subject $b_\tau(\tau_{i+1})$.

8. $b_\tau(\tau_i) \neq b_\tau(\tau_j)$ for $k = 0, 1, \ldots, G\text{-}1$ and $l = 0, 1, \ldots, D\text{-}2$, where $i = l_\nu(\nu_n)$ and $j = f_\nu(\nu_{n+1})$, $n = k\cdot D+l$. n is the index of the variable ν_n that defines the index i for the last lesson τ_i in day l for group γ_k.

   Subject $b_\tau(\tau_i)$ for the last lesson in a day except for the last day in the week (usually Friday) and subject $b_\tau(\tau_j)$ for the first lesson in the previous day for any group must be different.

9. $b_\tau(\tau_{i+f}) \neq b_\tau(\tau_{i+f+1}) \neq \ldots \neq b_\tau(\tau_{i+l})$ for all $b_\tau(\tau) \notin \mathbf{A}$, for $k = 0, 1, \ldots, G\text{-}1$ and $l = 0, 1, \ldots, D\text{-}1$, $i = k\cdot D\cdot N + l\cdot N$, where $n = k\cdot D+l$, $f = f_\nu(\nu_n)$, $l = l_\nu(\nu_n)$. n is the index of the variable ν_n that defines the indices for the first and last lessons τ_{i+l} and τ_{i+l} in day l for group γ_k.

   Subjects for all lessons τ_{i+f} to τ_{i+l} in a day l must be different for all subjects that can not be taught in blocks of two lessons $b_\tau(\tau) \notin \mathbf{A}$.

10. $a_\alpha(p_\tau(\tau_i), j, m_\lambda(\lambda_l), k) = 1$ for $l = 0, 1, \ldots, G\text{-}1$; $j = 0, 1, 2, \ldots, D\text{-}1$; $k = 0, 1, 2, \ldots, N\text{-}1$; where $i = l\cdot D\cdot N + j\cdot N + k$.

    For all groups ($l = 0, 1, \ldots, G\text{-}1$), all days ($j = 0, 1, \ldots, G\cdot D\text{-}1$), and all lessons ($k = 0, 1, 2, \ldots, N\text{-}1$), the teacher $p_\tau(\tau_i)$ must be available for the lesson τ_i.

11. $B_\alpha(b_\tau(\tau_{i\cdot N+j}), j) = 1$ for $i = 0, 1, \ldots, G\cdot D\text{-}1$; $j = 1, 2, \ldots, N$.

    For all groups and all days ($i = 0, 1, \ldots, G\cdot D\text{-}1$), the subject $b_\tau(\tau_{i\cdot N+j})$ taught at any lesson ($j = 1, 2, \ldots, N$) must be allowed by the set **BA**.

12. $e_\beta(b_\tau(\tau_{f\cdot D\cdot N + j\cdot N+k})) = e_\beta(b_\tau(\tau_{(f+1)\cdot D\cdot N + j\cdot N+k})) = \ldots = e_\beta(b_\tau(\tau_{l\cdot D\cdot N + j\cdot N+k}))$ for $i = 0, 1, \ldots, Y\text{-}1$; $j = 0, 1, \ldots, D\text{-}1$; $k = 0, 1, 2, \ldots, N\text{-}1$; where $f = f_\psi(\psi_i)$ and $l = l_\psi(\psi_i)$.

    For all school years ($i = 0, 1, \ldots, Y\text{-}1$) and all days ($j = 0, 1, \ldots, D\text{-}1$), and all lessons in a day ($k = 0, 1, 2, \ldots, N\text{-}1$), the elective/non-elective property of the subject is equal for all groups.

13. $y_\gamma(\gamma_i) = y_\gamma(\gamma_j) \Rightarrow s_\lambda(\gamma_i) = s_\lambda(\gamma_i)$ for $i = 0, 1, \ldots, G\text{-}1$; $j = 0, 1, \ldots, G\text{-}1$; $i \neq j$

    If two groups γ_i and γ_j are in the same school year $y_\gamma(\gamma_i) = y_\gamma(\gamma_j)$, then they are in the same shift $s_\lambda(\gamma_i) = s_\lambda(\gamma_i)$.

14. $r_\tau(\tau_{i+k}) = r_\tau(\tau_{j+k}) \wedge s_\lambda(\gamma_m) = s_\lambda(\gamma_n) \Rightarrow b_\tau(\tau_{i+k}) = b_\tau(\tau_{j+k})$ for $i = 0, D{\cdot}N, 2{\cdot}D{\cdot}N, \ldots, (G{\cdot}D\text{-}1){\cdot}N$; $j = 0, D{\cdot}N, 2{\cdot}D{\cdot}N, \ldots, (G{\cdot}D\text{-}1){\cdot}N$; $i \neq j$; $m = i/(D{\cdot}N)$; $n = j/(D{\cdot}N)$; $k = 0, 1, \ldots, D{\cdot}N\text{-}1$.
If two groups γ_m and γ_n from the same shift $s_\lambda(\gamma_m) = s_\lambda(\gamma_n)$ are scheduled to share room $r_\tau(\tau_{i+k}) = r_\tau(\tau_{j+k})$ during lesson k, then lesson's subject will be same for both groups.
15. $r_\tau(\tau_{i+k}) \neq r_\tau(\tau_{j+k}) \wedge s_\lambda(\gamma_m) = s_\lambda(\gamma_n) \Rightarrow p_\tau(\tau_{i+k}) \neq p_\tau(\tau_{j+k})$ for $i = 0, D{\cdot}N, 2{\cdot}D{\cdot}N, \ldots, (G{\cdot}D\text{-}1){\cdot}N$; $j = 0, D{\cdot}N, 2{\cdot}D{\cdot}N, \ldots, (G{\cdot}D\text{-}1){\cdot}N$; $i \neq j$; $m = i/(D{\cdot}N)$; $n = j/(D{\cdot}N)$; $k = 0, 1, \ldots, D{\cdot}N\text{-}1$.
If two lessons τ_{i+k} and τ_{j+k} that happen at the same time k and in the same shift $s_\lambda(\gamma_m) = s_\lambda(\gamma_n)$ are held in different rooms $r_\tau(\tau_{i+k}) \neq r_\tau(\tau_{j+k})$, the teachers must be different $p_\tau(\tau_{i+k}) \neq p_\tau(\tau_{j+k})$.
16. Timetable breaks for teachers are minimized.

6. Implementation of a hybrid simulated annealing algorithm

When solving the school scheduling problem, we attempted to add additional functionalities from other optimization algorithms in Simulated Annealing (SA). We started by SA knowing of its power to avoid local optima and its theoretical guaranty to find the global optimum. SA has been extensively researched and has shown satisfactory results in solving problems of combinatorial optimization and temporal and spatial scheduling (Duong & Lam, 2004), (Abramson, 1991), (Aarts et al., 2003), (Czech et al., 2006). In the software library, we implemented a combined version of the SA algorithm. It has elements of memory from the Tabu Search as well as a complex neighborhood function for local search similar to the Guided Search algorithm.
Detailed explanation of the algorithm follows after the pseudo code:

```
initialSol ← ConstructAsCorrectInitSolutAsPossible();
SolutionNeighborhood.SetSolution( initialSol );
{listOfVarsToChange, currentEnergy} ← CalculateEnergy(initialSol);
temp = InitialTemperature;
do
do
SolutionNeighborhood.GenerateCandidateSol ( currentSol, listOfVarsToChange);
FindAffectedConstraints();
{listOfVarsToChange,newEnergy}  ←  CalcEnergy(SolNeighborhood.Candidate);

if  ( Metropoliten(newEnergy - currentEnergy,temp))
                    SolutionNeighborhood.AcceptCandidate();
else
                    SolutionNeighborhood.RefuseCandidate();
endif

while ( !stopSearch and (trials < saMaxTrials )
          and successfulTrials < saMaxSuccessfulTrials
          and smallestEnergy > lowerEnergyBound );

temp = TempSchedule.GetNewTemperature();

while ( !stopSearch
```

and numberOfTempDecreases < MaxTempDecreases
and SolutionCount < MaxNumberOfSolutionsToFind
and smallestEnergy > lowerEnergyBound
and consecutiveNoSuccess < MaxConsecNoSuccess **)**

return SolutionNeighborhood.ReturnBestFoundSolution();

The algorithm initially constructs the solution in a way that as many as possible constraints are satisfied, and the cost - energy is reduced to minimum. In the particular high school schedule generation, for every group, in the available time slots, the appropriate number of classes per subject is filled. A teacher is assigned for every subject. All lessons are inserted in the time slots continually, until the appropriate numbers of classes per subject per group are achieved. The remaining task for the algorithm is to move the items (group, subject, teacher) in other time slots during the week, so that the remaining constraints are satisfied (not repetition of the same subject twice in the same day etc.)
In the meantime, an object from the CNeighborhood class is generated (Jolevski et al., 2005a). This object holds the current solution, and when asked for, generates a new proposal solution in the neighborhood of the current one. This is the place where the local search in the search space is performed.
After the neighborhood function generates a new solution, the algorithm invokes the function FindAffectedConstraints(). It detects the constraints whose satisfaction has been changed during the previous solution perturbation. Having this information, the function CalculateEnergy() calculates only the participation of the affected constraints within the overall energy, as opposed to recalculating the entire cost. Additionally, the function CalculateEnergy() generates a new list listOfVarsToChange. The list states which variables to change in the next solution perturbation so that a better solution is derived.
The Metropoliten() function implements the Metropolis probability distribution function. Considering the difference of energies of the previous and actual solution, as well as the temperature parameter, it decides whether to accept or reject the new solution.

$$P_T(\Delta E(X)) = \begin{cases} 1, & \Delta E(X) <= 0 \\ \exp(-\frac{\Delta E(X)}{T}), & otherwise \end{cases}$$

The inner iterations continue executing at constant temperature until one of the given conditions is met. Execution stops when the predefined maximal number of iterations per temperature is achieved, the maximally allowed number of accepted solutions is achieved, or the expected minimal energy is evaluated.
After ending the internal iterations, the function myTempSchedule.GetNewTemperature() is invoked. Depending on the chosen temperature schedule, a new value for the parameter temperature is derived. Having the new temperature, a new cycle of the internal iterations follows. If the conditions for ending the entire solving are met, the algorithm returns the best found solution. The ending conditions consist of achieving predefined number of temperature iterations, achieving minimal energy or achieving a predefined number of iterations where the metropolis function has not accepted any solution proposals.
In the presented implementation of SA we included functionalities from other metaheuristic algorithms. Memorizing and using the list of previously affected variables in the new

solution proposal in our algorithm includes memory in the solving. Memory is an element from Tabu Search. The list of affected variables helps in guiding the search and avoiding cycles. Generating a new solution proposal based on knowledge of the problem and previous experience adds elements of Guided search.

6.1 Neighborhood function for generation of an improved solution

Despite the effort to build a universal solving engine, there are certain parts of the system that seriously benefit from specialization. We estimated that the best part to implement specialization, in respect to both modularity and performance, is the neighborhood function. A neighborhood generation function is required to generate a new solution similar to, or in the "neighborhood" of, the previous one. The new solution is expected to have a lower cost (energy), meaning that the level of constraint satisfaction is higher. Presumably, the next solution should keep the qualities of the former and hopefully correct its weaknesses. The new solution proposal sometimes might have worse qualities than the previous one, but it still might be accepted as base of future solutions. Such acceptance has to be allowed to escape local in the pursuit of the global optima.

There are numerous ways to implement new trial solution generation. We used several combined approaches. The first method that we used, mainly during the first iterations of the solving process, is random based variable permutation. The method at the beginning considers how big part of the current solution should be changed in each iteration. We experimented with numbers from two variables up to 10% of the overall variables. The number of improvements in the solution during algorithm iteration increased as we decreased the number of changed variables. This behavior implies that a small change in the new solution is easier to find wrong and undo, while massive changes in the solution might compensate both corrections and faults, to a minor collective change in overall cost. Therefore, changing a small number of variables in each iteration makes the cost function an objective measure of progress or regress.

The neighborhood function randomly chooses which variables to change, keeps an array of changed variables and randomly assigns new values to the chosen variables from their domain. The array of changed variables is later used to calculate the new cost as a difference with the former one. Other approaches found in literature (Abramson, 1991) consist of swapping values of two variables. Nevertheless, for the sake of speed, the tendency is to always achieve calculation of the new cost as a difference with the last one.

6.2 Intelligent generation of the initial solution

The first place to implement intelligence in the solving process is the initial solution of the search. Significant number of iterations is avoided if the initial solution is not generated randomly, but in respect to the imposed constraints. Naturally, not all constraints could be satisfied at the start, otherwise, no search would have been necessary. We decided to satisfy as many constraints as possible at the beginning. Later during the solving process, initially satisfied constraints are not even checked for consistency, sparing many calculations in the evaluation step. This behavior is only possible if the neighborhood function includes special precaution. The functions must not dissatisfy these initial rules while generation of the next solution.

For example, all groups are initially given the right number of classes per subject. Therefore, no additional checks are necessary for this constraint, assuming the neighborhood function

only swaps positions of those classes in the available time and space slots. The neighborhood function must not add or subtract new classes per subject per group. Also, the classes are initially placed in rooms that they can be taught in.

6.3 Guided generation of new trial solutions

Different algorithm hybridizations based on SA for solving constraint modeled non-linear problems have been previously proposed. An example algorithm is CSAGA by Wah et. al (Wah & Chen, 2001). They combine SA and Genetic Algorithms (GA). The participation of the GA is in the creation of new solution proposals. A new generation of solution proposals is generated with genetic operators in each SA iteration. Every proposal is evaluated by multiplying its Lagrange-multipliers. The best proposal is selected with a GA.

In our SA implementation, additional intelligence was implemented in the neighborhood function. In the generation of the next solution, the neighborhood function uses listOfVarsToChange, the list of variables that it should change. The list of variables is derived when checking the constraint satisfaction and the particular variables that participate in the unsatisfied constraints. During the evaluation of the previous solution, the algorithm remembers which constraint generated the most of the unwanted cost. Knowing the variables that participate in the given constraint, the algorithm knows which variables should be changed to correct the current solution. This is how intensification of the search is achieved. Diversification is achieved by occasionally invoking a more complex neighborhood function, in a way explained later in the text.

One could favor the idea of generating the new solution by forcing changes in the variables that make the most of the unwanted cost at that point. However, we face two setbacks: the increased intelligence in the neighborhood function will decelerate the iterations; and the danger of trapping in local minima is increased. Nevertheless, random generation not always succeeds to find the final correct solution and therefore guided search is required to be implemented. Guided local search has already proven effective in solving the scheduling problem (Tsang et al., 1999).

We implemented guided search that includes different algorithms of swapping variable values. Depending on the constraint that has been dissatisfied, adequate neighborhood sub function is invoked. The neighborhood function swaps values of variables either in informed manner, generating a better solution, or in random position. For instance, if an empty class is spotted in-between classes, this empty time slot has to be filled with a class, so the last class of that day for the group is placed in that position, pushing the empty classes at the end. If two identical classes are found next to each other, one of them has to go in a different day. Swapping is performed between this class and a class from the next day, separating identical classes in different days.

Figure 4 presents the invoking of particular neighborhood sub-functions during algorithm iterations. The horizontal axis represents the algorithm iterations, while the vertical axes contains all 16 constraints that model the problem. Clearly, some constraints that were satisfied in the construction of the initial solution never have their sub-functions invoked later. The remaining constraints (5, 6, 7, 8, 15) are invoked with variable frequencies.

Figure 5 shows the distribution of calls of the particular constraint's sub-functions during the solutions search. The horizontal axis represents the constraints invoked during solving, and the vertical axis gives the number of calls of the given constraint. The most invoked constraint is the 15th, the collision when one teacher is placed to teach in two classrooms in

the same time. Because of its frequent invocations, a rather simple algorithm for correction has been developed. Two lessons from two time slots of one of the groups in the collision are chosen and swapped. For example, one of the collided lessons and another random lesson from the same group exchange their time slots. It is important to emphasize the random component that forces diversification in the solution search.

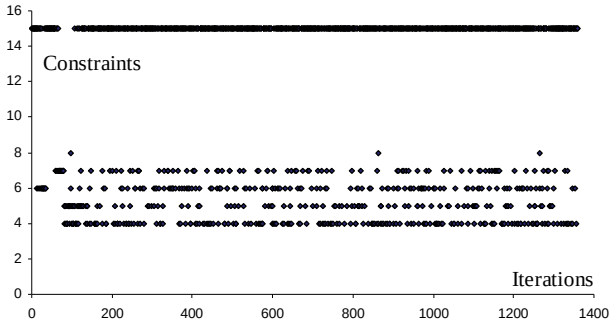


Fig. 4. Calls for evaluation of particular constraints of the model in every iteration of the solution search

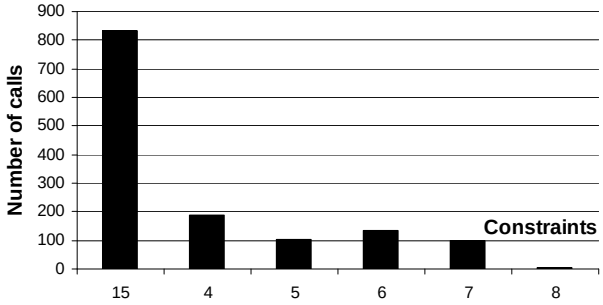


Fig. 5. Distribution of calls toward particular constraint's sub-functions during solving

6.4 Deadlocks

Practice has shown that simple variable swapping often brings to deadlocks. Initially a certain constraint is dissatisfied causing the appropriate swapping neighborhood function to swap values. The new solution dissatisfies a different constraint causing its swapping function to make the previous swap rollback. At this point a cyclic deadlock starts, with no possibility of ending. We solved this problem rather simply, but effectively. Every neighborhood function contains more than one swapping algorithm that triggers randomly, with different probability.

Certain more effective and easily calculable swapping methods are executed more often. Since they tend to cause deadlocks, once in a while, another different neighborhood function is triggered for the same constraint dissatisfaction (equation 1). For instance, if *rnd* is a randomly generated number such that $1 \leq rnd \leq 1000$, then:

$$NewSol = \begin{cases} SimpleSwap(OldSol), if\ 1 < rnd < 1000 \\ ComplexSwap(OldSol), if\ rnd = 1 \end{cases}$$

Varying the probability of triggering different neighborhood functions evidently influences solving time and result quality. Table 1 shows the dependences of solving speed and solution quality from the probability of using a more thorough permutation opposite to simple swap in the neighborhood function. A measure of solving quality is achieved minimal cost, number of made improvements, number of temperature decreases (SA parameter) and duration of the search.

Probabil.	Min. Cost	Number Improve.	Nr.Temp Decreases	Duration (msec)
1/5	315	68	2926	335857
1/10	355	80	5558	306890
1/100	326	86	3434	137983
1/1000	201	99	4273	133300
1/5000	297	99	4440	238560
1/10000	300	83	3214	259314

Table 1. Dependences of solving from the probability of using a complex neighborhood

Extremely high probabilities (0.2, 0.1) do not achieve the lowest solution cost, and solving duration is prolonged. Here the complex and thorough permutation is triggered too often and convergence toward the final solution is interrupted even when there is no deadlock. Extremely low probabilities (0.0001) also give unsatisfactory results because with such low probabilities, the thorough permutation is not triggered even when there is a deadlock going on. We chose to use a probability 1/1000 = 0.001, because it seems, the required neighborhood function triggers exactly when a deadlock happens.
Figure 6 shows the dependences of cost - energy in terms of SA, number of improvements and temperature decreases from the probability of using a complex neighborhood. It is visually evident that the chosen probability of 1/1000 = 0.001 gives the maximal number of improvements and the minimal energy - cost.

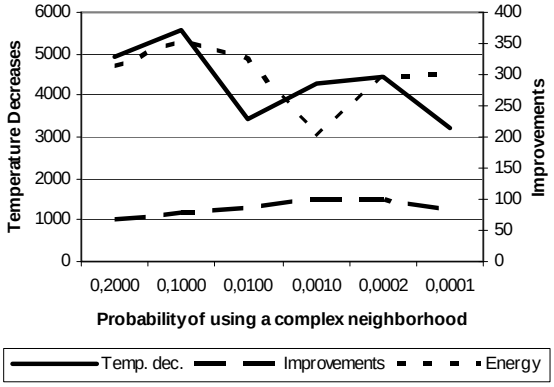


Fig. 6. Dependences of improvements, cost and energy from the probability of using a complex permutation.

Some of the parameters of SA must be functionally dependant from the input parameters of the problem. For instance, the number of iterations of the inner loop of SA or the number of overall temperature decrements of SA is a multiple of 100 and the number of variables in the problem model. The maximal number of algorithm executions without accepting new solutions is 100.

6.5 Impact of guided search

Our search through the solution space is guided in a manner such that the neighborhood function accepts the old solution proposal, along with the constraint that generates most of the cost function. Since the neighborhood function includes specific algorithms for solution improvement of each constraint, the appropriate one is triggered. The neighborhood function generates a new solution in the "neighborhood" of the current one, precisely marking the changed variables. Only the participation of the changed variables is recalculated into the overall solution cost. The recalculation function marks the constraint that contributes most to the cost.

Every constraint involves a different number of variables that create the problem model. Hence, every new solution generation changes a different number of variables depending on the constraint scope. We already determined (Chorbev et al., 2007) that changing a smaller amount of variables in every iteration gives better results. Still, letting the affected constraint decide the degree of change in the new iteration, allows appearing of occasional jumps in the search space. Massive change among the variables means jumping to new solution neighborhoods, when previous local search is exhausted.

The experimental results are generated from solving a school scheduling problem with 60 groups, 75 teachers, 37 rooms (utilized on two shifts) and 45 different subjects. Figure 7 shows the variations of the solution cost (energy) during the first 100 iterations of the solving process.

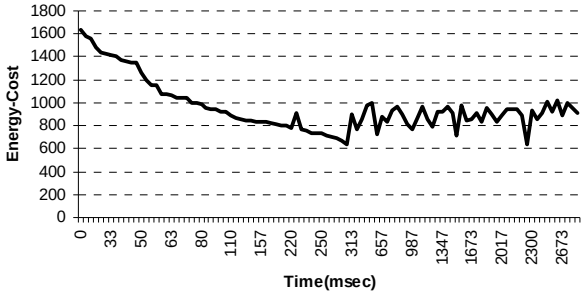


Fig. 7. Variation of solution cost in time during the first 100 iterations

The graph on figure 7 shows that informed - guided search in the beginning quickly directs the solution toward a lower solution cost. At that point the algorithm performs exact solving rather than heuristic search in each iteration. The neighborhood function exactly changes the dissatisfying variable for the most influential constraint to an accurate value. However, this process does not last long, quickly getting to a situation where every intervention among variables causes another equally influential constraint to become dissatisfied. In each succeeding iteration of the optimization algorithm, a triggered neighborhood corrects one, but dissatisfies other constraints.

This period of averagely constant cost, with a very small overall cost decrement lasts for most of the solving process (figure 8). During this period, swift jumps and depressions characterize the graph. The neighborhood function easily jumps to often completely new solutions. This behavior is inherited from Simulated Annealing, because high "temperature" allows the probability for acceptance of worse solutions to remain significantly big. As temperature decreases in SA and many of the neighborhoods have been tested, jumps start happening rarely. The search is directed in the neighborhood that has shown best potential for the final solution. The entire solving process can be seen at figure 8.

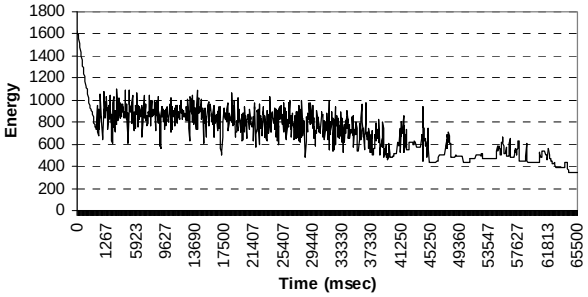


Fig. 8. Variation of solution cost in time during entire schedule generation

In this process, resulting from the nature of the algorithm SA, worsened solutions are accepted as bases for further improvement, therefore managing to escape the emerging deadlocks. Although the search is guided, and the functions change specific error variable values, randomness must still be present. The neighborhood functions swap values between time slots in the schedule, but often new positions for swapping values are given randomly. The randomness is controlled with the domain of the variables and the goal to satisfy other constraints. The combination of worse solution acceptance and randomness in new solution generation is the key to avoiding trapping in local optima and escaping deadlocks.

7. Conclusion

The chapter deals with description of the heuristic algorithm that we implemented to build a school scheduling software. The scheduling software is based on a Constraint Solving Engine (CSE) and a Constraint Programming Library (CPL) which we previously developed (Chorbev et al., 2007). Various simulations and tests of the solving process implied the required corrections to the model (Jolevski et al., 2005c) and the algorithms.

Every mentioned heuristic algorithm has certain advantages that come in handy in specific circumstances and specific problems. The goal in our research was to extract the best ideas and develop a novel hybrid algorithm that will achieve better performances. In this particular case, we tried to add additional functionalities from other optimization algorithms in Simulated Annealing as basis. We started from SA knowing of its power to avoid local optima and its theoretical guaranty to find the global optimum. We combined the memory from Tabu search, the intelligence of guided search and the completeness of GAC-CBJ. Initially we tested and fine tuned the algorithms on trivial problems like the traveling salesman, quadratic assignment or the n-queens problem. Eventually we took the schedule generation problem as a way to give practical implementation of the developed hybrid algorithms.

The area of constraint programming is quite perspective in the sense that it can use a lot of knowledge previously gathered from analysis in logical programming (Prolog). Furthermore, new better heuristic algorithms are implemented for solving problems modeled with constraints turning the constraint satisfaction into a perspective strategy. Even in worse case scenario, if solving through constraints does not give the promised results, the universal mathematical modeling of problems is a contribution by itself. Having the problem modeled in a reusable way is a base for implementing new ideas in future.
The modularity of the concept, the clear distinction between the model and the algorithm leaves room for separate independent corrections and enhancements. The concept of universality that results from the modularity is exceptionally useful. Having the model and the algorithm separated, they can both be replaced. The algorithm can be used for another problem, or the modeled problem can be solved with another algorithm, with only minor additional interventions. Enhancing and optimizing the clearly distinct model is easier. The separated model can be used for testing new algorithms and getting better results.
When building a solving engine, the universal algorithms and functions are implemented manage to give a solution. However, certain customizations and integration of small heuristic drastically accelerate the solving process. Adding guided search of the solution space showed significant improvement opposed to simple random solution proposal. However, a stochastic component is useful to avoid deadlocks and trapping in local optima.
By limiting the involvement of appropriate heuristic for the given problem type to modular components in the engine, the universality of the library is maintained. At the same time the performance is significantly increased. Finding a balance between universal optimizing functions and problem dependent heuristics, improves the engine for further more or less similar tasks.

7. References

Aarts, E. H. L., Korst, J. H. M., Laarhoven, P. J. M. V. (2003). *Simulated annealing In Local Search in Combinatorial Optimization*, Princeton University Press, ISBN: 0691115222, Princeton, New Jersey 08540 USA.

Abramson D. (1991) Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms, *Management Science*, Volume 37, Issue 1, pages 98 – 113

Bartak, R. (1999): Constraint Programming: In Pursuit of the Holy Grail, *Proceedings of WDS99*, Part IV, June 1999, pp. 555-564, MatFyzPress, Prague.

Blum C., Roli A., (2003) Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, Vol. 35, No. 3, pp. 268–308.

Blum C., Roli A., Alba E., (2005), *Parallel Metaheuristics*, *Wiley Book Series on Parallel and Distributed Computing*, John Wiley & Sons, ISBN: 9780471739388

Cave A., Nahavandi S., Kouzani A. (2002) Simulation Optimization for Process Scheduling through Simulated Annealing, *Proceedings of the 2002 Winter Simulation Conference*, 2002, San Diego, California, USA, pages 1269-1273.

Chorbev I., Dimitrovski I., Mihajlov D., Loskovska S. (2007) Hybrid Heuristics for Solving the Constraints Modeled High School Scheduling Problem, *Proc. of IEEE Region 8 Eurocon 2007 Conf.*, pages 2242-2249, ISBN: 978-1-4244-0813-9, Warsaw, Poland.

Chorbev I., Dimitrovski I., Loskovska S., Mihajlov D.(2006), A parallel implementation of Simulated annealing with a Boltzmann synchronization function and its application to solve the traveling salesman problem, Proc *IS2006*, pp 14-17, Ljubljana, Slovenija

Crawford B., Castro C., Monfroy E., (2007) Integration of Constraint Programming and Metaheuristics, *Lecture Notes in Computer Scienc, Abstraction, Reformulation, and Approximation,* Springer Berlin / Heidelberg, pp 397-398, ISBN 9783540735793

Czech Z., Wieczorek B.(2006) Solving bicriterion optimization problems by parallel simulated annealing, *Proceedings of the 14th Euromicro Workshop on Parallel, Distributed and Network –based Processing,* pp:7-14, ISBN ~ ISSN:1066-6192 , 0-7695-2513-X, 15-17 Feb. 2006, IEEE Computer Society Washington, DC, USA

Der-Rong Din, Tseng, S.S. (2001): Heuristic and Simulated Annealing Algorithms for Extended Cell Assignment Problem in Wireless ATM Network; *Journal of Information Science and Engineering*, Vol.17 No.4, pp.647-665

Duong T., Lam K. (2004), Combining Constraint Programming and Simulated Annealing on University Exam Timetabling, *Int. Conf. RIVF'04*, pages 205-210, Hanoi, Vietnam.

Gavanelli, M. (2002): Interactive Constraint Satisfaction Problems for Artificial Vision, in Ph.D. thesis, Department of Engineering, University of Ferrara, Italy.

Glover F. and M. Laguna (1997), *Tabu Search*, Kluwer Academic Publishers, Boston, ISBN 0 7923 8187 4.

Glover F., Laguna M. and Martí R., (2003), *Advances in Evolutionary Computation: Theory and Applications*, A. Ghosh and S. Tsutsui (Eds.), pp. 519-537, Springer-Verlag, ISBN:3-540-43330-9

Gomes N., Vale Z., Ramos C. (2005) Combining metaheuristics and constraint programming to solve a scheduling problem, *P. of the 4th WSEAS Int. Conf. on Applied Mathematics and Computer Science*, Article No. 5, ISBN:960-8457-17-3, Rio de Janeiro, Brazil, 2005

Hao J., Pannier J., (1998) Simulated Annealing and Tabu Search for Constraint Solving, in *Fifth Intl. Symposium on Artificial Intelligence and Mathematics*,

Hogg, T. and Huberman, A. (1993), Better than the best: The power of cooperation. *1992 Lectures in Complex Systems*, pp. 163–184, Addison-Wesley, Reading, MA.

Jolevski I., Loskovska S., Chorbev I., Mihajlov D., (a 2005) An Overview of a Constraint Solving Engine with Multiple Optimization Algorithms, *Proc. of the 27th International Conference on Information Technology Interfaces*, pp: 602- 608, ISBN: 953-7138-02-X, Cavtat, Croatia, June 20-23, 2005.

Jolevski I., Loskovska S., Chorbev I., Mihajlov D., Murgovski N., (c 2005) Constraints Modeling of the High School Scheduling Problem, *The Int. Conf. on Computer as a Tool, EUROCON 2005*, pp: 748-751, ISBN: 1-4244-0049-X, Belgrade, Serbia and Montenegro, 2005.

Jolevski I., Loskovska S., Chorbev I., Mihajlov D.,(b 2005) A Solution of N Queen Problem using a Constraint Solving Engine with Multiple Optimization Algorithms, *Proc. of ETAI 2005*, pages I56-I61, Ohrid, R. of Macedonia, 2005.

Jolevski I., Loskovska S., Murgovski N., Chorbev I., Mihajlov D.,(d 2005) Development of a user interface for a school scheduling application, ETAI, Ohrid, R. of Macedonia, *Proc. of ETAI 2005*, pages I4-4,I90–I95, Ohrid, R. of Macedonia, 2005.

Kumar, V. (1992): Algorithms for Constraint - Satisfaction Problems: A Survey, in *AI Magazine,* Volume 13, Issue 1, Spring 1992, pp: 32 - 44, American Association for Artificial Intelligence, ISSN:0738-4602, Menlo Park, CA, USA,

Leenen L., Venter L., Britz K.,(2003) A Pre-processing Algorithm for solving Constraint Satisfaction Optimization Problems, *Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on*

Enablement through technology, p.11-15, ISBN:1-58113-774-5, September 17-19, 2003, South African Institute for Computer Scientists and Information Technologists

Meyer, M., (1994), Finite Domain Constraints: Declarativity meets Efficiency, in Doctor Dissertation

Ohlidal M., Schwarz J. (2004), Hybrid Parallel Simulated Annealing Using Genetic Operations, *Mendel 10th Internacional Conference on Soft Computing*, pp. 89-94, ISBN: 80-214-2676-4, Brno, Czech Republic, FSI VUT, 2004.

Penya Y. K., Jennings N. R., Neumann G. (2005) An Optimal Distributed Constraint Optimization Algorithm for Efficient Energy Management, *Proc. of Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, pp: 17- 182, ISBN: 0-7695-2504-0, 28-30 Nov. 2005, Addison-Wesley Publishing Company, Inc.

Tam V. and Ting D.,(2003) Combining the Min-Conflicts and Look-Forward Heuristics to Effectively Solve A Set of Hard University Timetabling Problems, *Proc. of the 15th IEEE International Conference on Tools with Artificial Intelligence*, p. 492, 1082-3409/03

Tsang, Wang, Davenport, Voudouris & Lau.(1999) A family of stochastic methods for constraint satisfaction and optimization, *The First International Conference on The PACLP*, London, pages 359-383

Wah B., Chen Y.,(2001) Hybrid Constrained Simulated Annealing and Genetic Algorithms for Nonlinear Constrained Optimization, *Congress on Evolutionary Computation*, IEEE, pages. 925-932, Volume 2, 2001

Yoshikawa M., Kaneko K., Yamanouchi T., Watanabe M.,(1996) A Constraint Based High School Scheduling System, *Intelligent Systems and Their Applications*, pp 63 - 72, Vol. 11, Issue 1, ISSN:0885-9000, IEEE Educational Activities Department, NJ, USA

Zervoudakis K., Stamatopoulos P..(2001) A Generic Object-Oriented Constraint Based Model for University Course Timetabling, *Proc. of the 3rd International Conference on the Practice and Theory of Automated Timetabling PATAT 2000,* Lecture Notes In Computer Science; Vol. 2079, pp 28 - 47, ISBN:3-540-42421-0, Springer-Verlag London, UK, 2000

Zhaohui F. and A. Lim, (2000) Heuristics for the Exam Scheduling Problem, *Proc. of the 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'00),* pp 172-175, ISBN: 0-7695-0909-6.

Toward Improving b-Coloring based Clustering using a Greedy re-Coloring Algorithm

Tetsuya Yoshida[1], Haytham Elghazel[2], Véronique Deslandres[2],
Mohand-Said Hacid[3] and Alain Dussauchoy[2]
[1] *Graduate School of Information Science and Technology, Hokkaido University,*
[2] *Université de Lyon, Lyon, F-69003, France ; université Lyon 1, EA4125, LIESP*
[3] *Université de Lyon, Lyon, F-69003, France ; université Lyon 1, LIRIS,*
[1]*Japan*
[2,3]*France*

1. Introduction

Clustering is an important task in the process of data analysis which can be viewed as a data modeling technique that provides an attractive mechanism to automatically find the hidden structure of large data sets (Jain et al., 1999). Informally, this task consists of the division of data items (objects, instances, etc.) into groups or categories, such that all objects in the same group are similar to each other, while dissimilar from objects in the other groups. Clustering plays an important role in data mining applications such as Web analysis, information retrieval, medical diagnosis, and many other domains.

Recently, we have proposed a clustering method based on the concept of b-coloring of a graph (Irving & Manlov, 1999). A graph b-coloring is an assignment of colors to the vertices of the graph such that:

i. no two adjacent vertices (vertices joined by an weighted edge representing the dissimilarity between objects) have the same color (*proper coloring*)
ii. for each color, there exists at least one vertex which is adjacent (has a sufficient dissimilarity degree) to all other colors. This vertex is called a *dominating vertex*; there can be many within the same class.

Both (i) and (ii) are the constraints in b-coloring of a graph.

The b-coloring based clustering method enables to build a fine partition of the dataset into clusters even when the number of clusters is not specified in advance. The previous clustering algorithm in (Elghazel et al., 2006) conducts the following two steps in greedy fashion:

1. initalizes the colors of vertices so that the colors satisfy proper coloring, and
2. removes, by a greedy procedure, the colors that have no dominating vertices, until each color has at least one dominating vertex.

These steps correspond to the above two constraints in b-coloring. Although it returns a b-coloring of a graph, it does not explicitly consider the quality of the clusters in the algorithm. Thus, besides satisfying the above constraints, it was difficult to explicitly generate *better* clusters of the given data items.

In order to alleviate this weakness, we have proposed a greedy algorithm which realizes the re-coloring of data items (vertices) to improve the quality of the constructed partition (Elghazel et al., 2007). Informally, our algorithm selects at each stage the vertex with the maximum degree of "outlier" and which do not affects the dominant vertices in the b-coloring. The color of the selected vertex is changed while guaranteeing that the quality of the re-colored partition is monotonically improved. The selection of vertices as well as that of the assigned colors are conducted in greedy fashion. Our greedy algorithm exhibits the following characteristics:

1. it realizes the update of b-coloring based clustering while satisfying the constraints in b-coloring,
2. it monotonically increases the quality of clusters (the quality clusters needs to be measured by some objective function). This enables to realize a compromise between the intra-cluster cohesion and intercluster separation, and
3. it employs a simple greedy strategy, in order to reduce its time complexity.

Thus, the proposed greedy algorithm can complement the weakness of the previous method by improving the constructed partition.

To evaluate the effectiveness of the proposed greedy algorithm, we tested it over benchmark datasets from the UCI repository (Blake & Merz, 1998). The detailed results of the evaluations are reported and discussed in this paper. Through this evaluation, the effectiveness of the proposed greedy algorithm is confirmed.

This paper is organized as follows. Section 2 presents the related work. Section 3 explains the approach of b-coloring based clustering and validity indices for estimating the quality of clustering in general. Section 4 describes the details of the proposed greedy algorithm. Section 5 reports the results of the experiments to evaluate the proposed algorithm. Section 6 discusses our approach in terms of the greedy strategy and other possible improvements. Section 7 gives a brief conclusion.

2. Related work

Generally speaking, clustering of data can be divided into two approaches: a hierarchical approach and a partitioning approach. The hierarchical approach builds a cluster hierarchy, or a tree of clusters (which is called a dendrogram) whose leaves are the data points and whose internal nodes represent nested clusters of various sizes (Guha et al., 1998). On the other hand, the partitioning approach give a single partition of the data by fixing some parameters (number of clusters, thresholds, etc.). Each cluster is represented by its centroid (Hartigan & Wong, 1979) or by one of its objects located near its center (e.g., monoid) (Ng & and Han, 2002). When the distances (or, dissimilarities) among all the pairs of data can be estimated, these can be represented as a weighed dissimilarity matrix in which each element stores the corresponding dissimilarity. Based on the dissimilarity matrix, the data can also be conceived as a graph where each vertex corresponds to a data item and each edge corresponds to a pair of data items with their dissimilarity as its label.

Other techniques for realizing the clustering of data include graph-theoretic clustering approaches. Many graph-theoretic clustering algorithms basically consist of searching for certain combinatorial structures in the similarity graph. In this case, some hierarchical approaches are related to graph-theoretic clustering. The best-known graph-theoretic divisive clustering algorithm (the single-link algorithm) is based on construction of the Minimal Spanning Tree (MST) of the data (Zahn, 1971), and then deleting the MST edges

with the largest lengths to generate single-link clusters. The complete-link algorithms are also reduced to a search for a maximal complete subgraph, namely a clique which is the strictest definition of a cluster. Some authors have proposed to use the vertex coloring of graphs for the hierarchical classification purpose. (Guenoche et al, 1991) proposed a divisive classification method based on dissimilarity tables, where the iterative algorithm consists, at each step, in finding a partition by subdividing the cluster with the largest diameter into two clusters in order to exhibit a new partition with the minimal diameter. By mapping each data item to the corresponding vertex, the subdivision is obtained by a 2-coloring of the vertices of the maximum spanning tree built from the dissimilarity table. The derived classification structure is a hierarchy.
On the other hand, the partitioning methods are also related to graph-theoretic clustering. The method in (Hansen & Delattre, 1978) reduced the partitioning problem of a data set into clusters with minimal diameter, to the minimal coloring problem of a superior threshold graph. The edges of this graph are the pairs of vertices distanced from more than a given threshold. In such a graph, each color corresponds to one cluster and the number of colors is minimal. Unfortunately, while this method tends to build a partition of the data set with effectively compact clusters, it does not give any importance to the cluster-separation.

3. b-coloring based clustering

We use a bold italic capital letter to denote a set. For instance, $\boldsymbol{V}$ represents a set of vertices. In addition, $|\boldsymbol{V}|$ represents the cardinality of $\boldsymbol{V}$, i.e., the number of vertices in $\boldsymbol{V}$.
Our approach for the clustering of data assumes that some dissimilarity function for a pair of data to be handled is specified. By denoting the set of data to be handled as $\boldsymbol{V}$, the dissimilarity of a pairs of data $v_i, v_j \in \boldsymbol{V}$ is calculated by some function d: $\boldsymbol{V} \times \boldsymbol{V} \rightarrow \boldsymbol{R}^+$. We also assume that this function is symmetric.
Based on the dissimilarity function, the set of data $\boldsymbol{V}$ can be transformed into the corresponding graph-structured data by:

1. mapping each data to a vertex, and
2. connecting each pair of vertices v_i and $v_j \in \boldsymbol{V}$ by the edge (v_i, v_j) with label $d(v_i, v_j)$.

The above transformation results in an undirected complete edge-weighted graph. The b-coloring of this complete graph is not interesting in terms of the clustering problem. Indeed, each data will be assigned to one and the only one cluster, which is meaningless as the clustering of data. To alleviate this, we also require another parameter θ. This parameter works as the threshold value for defining the edges in the graph. Formally, a pair of vertices $(v_i, v_j \in \boldsymbol{V})$ are connected with the edge (v_i, v_j) in the graph iff $d(v_i, v_j) > \theta$ for the specified threshold θ. The constructed graph $G(\boldsymbol{V},\boldsymbol{E})$ is called a superior threshold graph.
The above notations are summarized in Table 1.

Symbol	Description
$\boldsymbol{V}$	a set of vertices (each vertex corresponds to a data item)
θ	a threshold value
$\boldsymbol{E}$	the set of edges among $\boldsymbol{V}$ for $d(,)$ and θ
$\boldsymbol{P}$	a set of clusters
$d(v_i, v_j)$	a dissimilarity function between vertices v_i and v_j

Table 1. Notations for a threshold graph

3.1 An example

Suppose a set of data with the dissimilarities in Table 2 is given, which is represented as a dissimilarity matrix for the data. Fig. 1 shows the superior threshold graph for Table 2 where the threshold θ is set to 0.15. The edges are labeled with the corresponding dissimilarities in the matrix.

vertex	A	B	C	D	E	F	G	H	I
A	0								
B	0.20	0							
C	0.10	0.30	0						
D	0.10	0.20	0.25	0					
E	0.20	0.20	0.15	0.40	0				
F	0.20	0.20	0.20	0.25	0.65	0			
G	0.15	0.10	0.15	0.10	0.10	0.75	0		
H	0.10	0.20	0.10	0.10	0.05	0.05	0.05		
I	0.40	0.075	0.15	0.15	0.15	0.15	0.15	0.15	0

Table 2. A weighted dissimilarity matrix

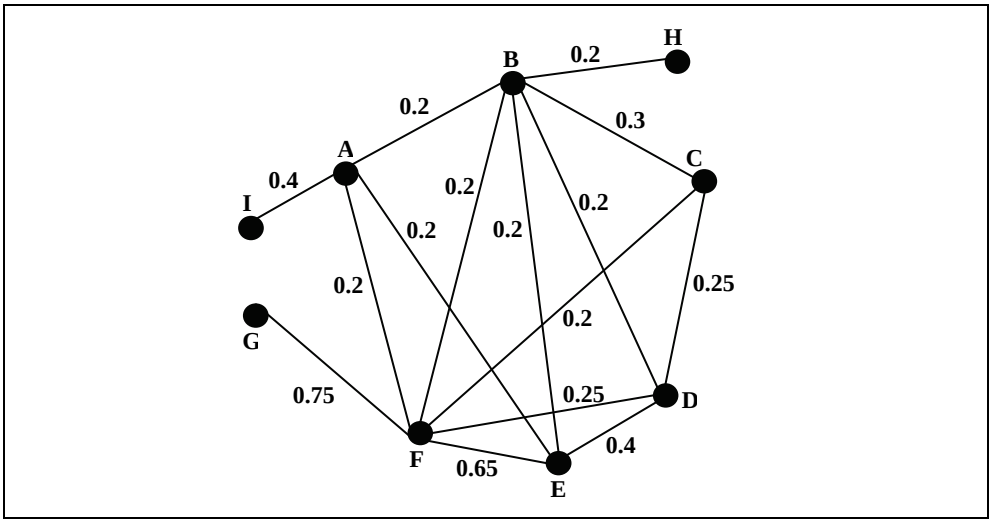


Fig. 1. A threshold graph for Table 1 (θ=0.15)

The previous b-coloring based clustering algorithm works with the following two stages:

1. initializing the colors of vertices so that the colors satisfy proper coloring, and
2. removing, by a greedy procedure, the colors without any dominating vertex.

Here, these stages are conducted with greedy fashion, because finding the maximum number of colors for b-coloring of a graph is known to be computationally too expensive. Utilization of a greedy strategy is a realistic approach for dealing with real-world data, especially for large scale data.

For instance, the proper coloring in Fig.2 is obtained from the graph in Fig.1 with step 1). After that, a b-coloring of the graph is obtained by step 2). The result is illustrated in Fig. 3. The vertices with the same color (shape) are grouped into the same cluster. This realizes the

clustering of data in Table 1. In this example, the sets of vertices {A,D}, {B}, {C,E,G,I}, {F} are the clusters.

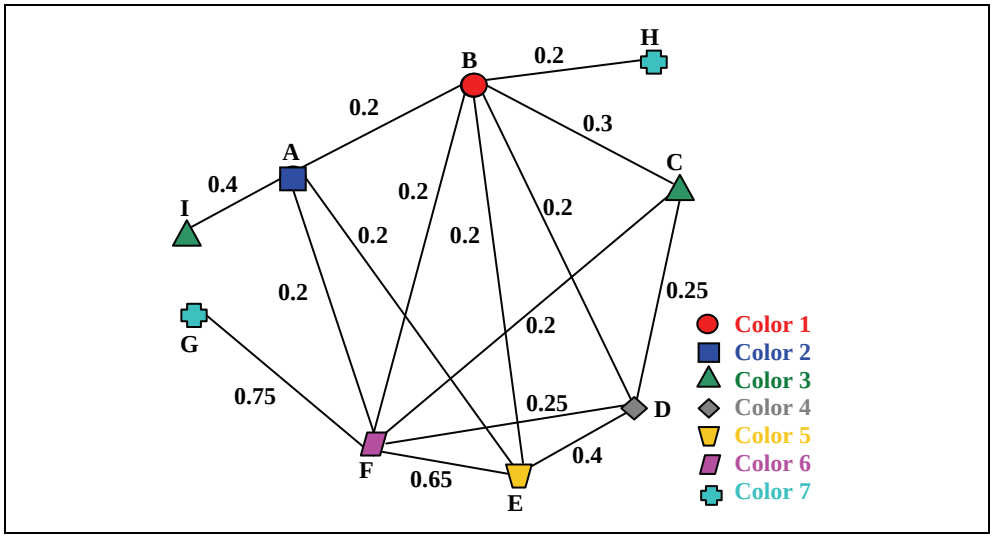


Fig. 2. A proper coloring of the graph in Fig.1

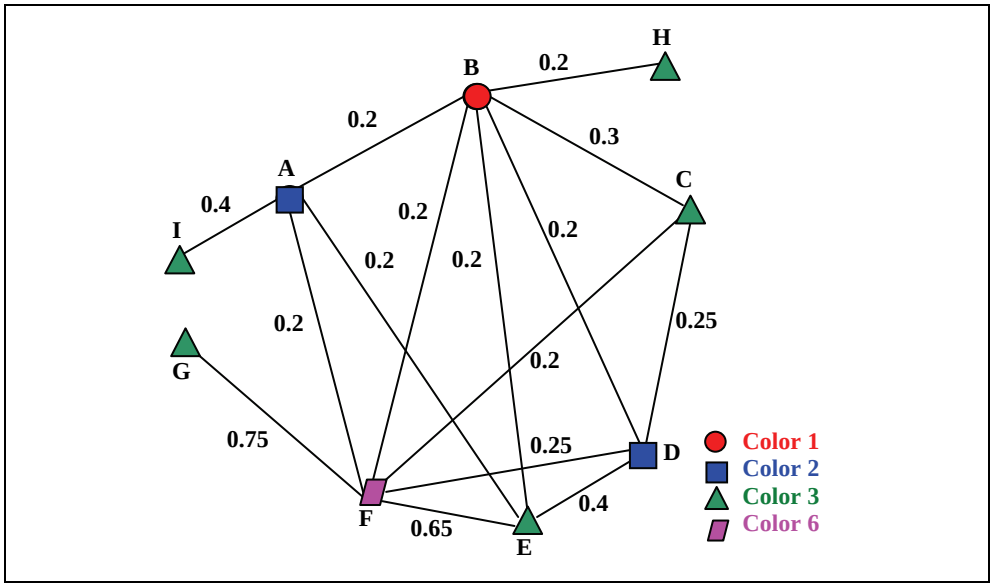


Fig. 3. A b-coloring of the graph in Fig.1 based on Fig.2.

3.2 Validation Indices

Many validation indices for clustering have been proposed (Bezdek & Pal, 1998) and adapted to the symbolic framework (Kalyani & Sushmit, 2003). Among them, we focus on a

validation index called generalized Dunn's index. This index is denoted as $Dunn_G$ hereafter in this paper. This index is designed to offer a compromise between the *inter-cluster separation* and the *intra-cluster cohesion*. The former corresponds to the compactness of the clusters, the latter corresponds to what extent the clusters are well-separated each other.

Suppose a set of vertices $\boldsymbol{V}$ (which correspond to the data items) are clustered or grouped into a partition $\boldsymbol{P} = \{C_1, C_2, \ldots, C_k\}$. Here, each cluster or group is denoted as C_i, and the partition $\boldsymbol{P}$ satisfies the constraint: $\forall C_i, C_j \in \boldsymbol{P}$, $C_i \cap C_j = \varnothing$ for $i \neq j$. We abuse the notation of $\boldsymbol{P}$ to represent both a set of clusters as well as a set of colors, because each cluster $C_i \in \boldsymbol{P}$ corresponds to a color in our approach and no cluster share the same color.

For $\forall C_h \in \boldsymbol{P}$, an average within-cluster dissimilarity is defined as

$$S_a(C_h) = \frac{1}{\eta_h(\eta_h - 1)} \sum_{o=1}^{\eta_h} \sum_{o'=1}^{\eta_h} d(v_o, v_{o'}) \tag{1}$$

where $\eta_h = |C_h|$, $v_o, v_{o'} \in C_h$.

For $\forall C_i, C_j \in \boldsymbol{P}$, an average between-cluster dissimilarity is defined as

$$d_a(C_i, C_j) = \frac{1}{\eta_i \eta_j} \sum_{p=1}^{\eta_i} \sum_{q=1}^{\eta_j} d(v_p, v_q) \tag{2}$$

where $\eta_i = |C_i|$ and $\eta_j = |C_j|$, $v_p \in C_i$, $v_q \in C_j$.

Dunn's generalized index for a partition P is defined as

$$Dunn_G(P) = \frac{\min\limits_{i,j,i\neq j} d_a(C_i, C_j)}{\max\limits_{h} S_a(C_h)} \tag{3}$$

where $C_h, C_i, C_j \in \boldsymbol{P}$.

Basically, the partition $\boldsymbol{P}$ with the largest $Dunn_G$ ($\boldsymbol{P}$) is regarded as the best clustering.

The above notations are summarized in Table 3.

Symbol	Description
$S_a(C_h)$	an average within-cluster dissimilarity of a cluster C_h
$d_a(C_i, C_j)$	an average between-cluster dissimilarity between C_i and C_j
$Dunn_G$ ($\boldsymbol{P}$)	generalized Dunn's index of a partition P

Table 3. Notations for evaluating a partition.

4. A greedy re-coloring algorithm

4.1 A motivating example

As explained in Section 3, for the data in Table 2, the previous approach returns the partition in Fig. 3 as its best b-coloring of the corresponding superior threshold graph. However, even for the same number of clusters, the graph in Fig. 1 can have other different b-colorings with better quality of clustering *(e.g.,* with larger value of $Dunn_G$ index). Actually, there is another b-coloring with better quality of clusters. An example is shown in Fig. 4. Obviously, the colors in Fig.4 satisfy the constraints in b-coloring and thus it is a b-coloring of the graph in Fig.1. Furthermore, the partition in Fig. 4 is better than that in Fig.3, since it has the value $Dunn_G$ =1.538, which is larger than the previous value (1.522) in Fig. 3.

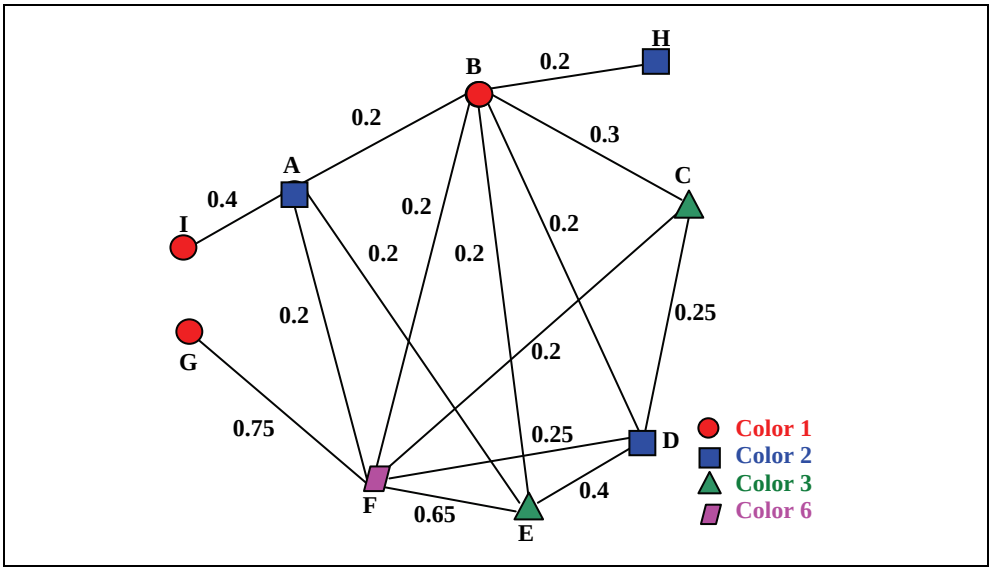


Fig. 4. Another b-coloring with better quality

As illustrated in the above example, even when the previous approach described in Section 3 returns a partition $\boldsymbol{P}$ based on the b-coloring of a given graph $G(\boldsymbol{V},\boldsymbol{E})$, there can be other partitions for the same graph with better quality, while satisfying the constraints in b-coloring. To construct a better partition, it is also important to find a partition with better quality, while satisfying the constraints b-coloring.

However, as described above, *directly* trying to find out a better b-coloring can be computationally too expensive. Even if a better partition can be obtained, it will not scale up for large data. To cope with this problem, we take the following approach: instead of directly finding out a better partition, by utilizing a partition which satisfies the constraints, try to find out better ones. This approach is formalized as follows.

[Definition 1] Re-Coloring Problem in b-Coloring based Clustering

For a given graph $G(\boldsymbol{V},\boldsymbol{E})$ and a b-coloring partition $\boldsymbol{P}$ of $G(\boldsymbol{V},\boldsymbol{E})$, find another b-coloring partition $\boldsymbol{P}'$ of $G(\boldsymbol{V},\boldsymbol{E})$ such that $\boldsymbol{P}'$ is equal to or better than $\boldsymbol{P}$ for some clustering validity index.

In our current approach, the quality of a partition $\boldsymbol{P}$ is measured with $Dunn_G(\boldsymbol{P})$ in Section 3.2. In the following, we describe the details of our approach to tackle this problem.

4.2 Notations

In addition to the notations in Table3, to characterize a vertex $v \in \boldsymbol{V}$ in a graph $G(\boldsymbol{V},\boldsymbol{E})$, we use the following functions for in the description of our greedy algorithm. A function $N(v)$ returns the set of neighboring vertices in $G(\boldsymbol{V},\boldsymbol{E})$. A function $c(v)$ returns the assigned color of the vertex v. A function $N_c(v)$ returns the set of neighborhood colors for the vertex v. Furthermore, a function $C_p(v)$ is defined as $C_p(v) = \boldsymbol{P} \backslash N_c(v)$ for v. Here, $\boldsymbol{P} \backslash N_c(v)$ represents the set difference. Note that $C_p(v)$ contains the originally assigned color $c(v)$ of the vertex v. These are summarized in Table 4.

Symbol	Description
$N(v)$	the set of neighborhood vertices for a vertex $v \in \boldsymbol{V}$ in $G(\boldsymbol{V},\boldsymbol{E})$
$c(v)$	the assigned color of a vertex $v \in \boldsymbol{V}$ in $G(\boldsymbol{V},\boldsymbol{E})$
$N_c(v)$	the set of neighborhood colors for a vertex $v \in \boldsymbol{V}$ in $G(\boldsymbol{V},\boldsymbol{E})$
$C_p(v)$	$C_p(v) = \boldsymbol{P} \setminus N_c(v)$ for a vertex $v \in \boldsymbol{V}$ in $G(\boldsymbol{V},\boldsymbol{E})$

Table 4. Several functions for a vertex in a graph

4.2.1 Types of vertex

A set of vertices $\boldsymbol{V_d}$ contain the dominating vertices in a b-coloring of a graph $G(\boldsymbol{V},\boldsymbol{E})$. For each vertex $v \in \boldsymbol{V_d}$, if a vertex v_s is the only vertex with the color $c(v_s)$ in $N_c(v)$, v_s is called a supporting vertex of v.

We divide the set of vertices $\boldsymbol{V}$ into two disjoint subsets $\boldsymbol{V_c}$ and $\boldsymbol{V_{nc}}$ such that $\boldsymbol{V_c} \cup \boldsymbol{V_{nc}} = \boldsymbol{V}$ and $\boldsymbol{V_c} \cap \boldsymbol{V_{nc}} = \emptyset$. Each vertex in $\boldsymbol{V_c}$ is called a critical vertex, and each vertex in $\boldsymbol{V_{nc}}$ is called a non-critical vertex. The vertices in $\boldsymbol{V_c}$ are critical in the sense that their colors cannot be changed (or, would not be changed in our current approach). On the other hand, the vertices in $\boldsymbol{V_{nc}}$ are not critical and thus considered as the candidates for the re-coloring. $\boldsymbol{V_c}$ is further divided into three disjoint sets of vertices $\boldsymbol{V_d} \cup \boldsymbol{V_s} \cup \boldsymbol{V_f}$. Here, $\boldsymbol{V_f}$ is called a set of finished vertices, and contains the already checked vertices for re-coloring. More detailed discussions about why it is important to define these sets of vertices are given in Section 4.3, with respect to the greedy nature of our algorithm. Furthermore, for $\forall v \in \boldsymbol{V}$, $\forall C_i \in \boldsymbol{P}$, an average dissimilarity between a vertex v and a cluster C_i is defined:

$$d_a(v, C_i) = \frac{1}{\eta_i} \sum_{p=1}^{\eta_i} d(v, v_p) \tag{4}$$

where $\eta_i = |C_i| v_p \in C_i$.

The above notations are summarized in Table 5.

Symbol	Description
$\boldsymbol{V_c}$	the set of critical vertices
$\boldsymbol{V_{nc}}$	the set of non-critical vertices
$\boldsymbol{V_d}$	the set of dominating vertices
$\boldsymbol{V_s}$	the set of supporting vertices
$\boldsymbol{V_f}$	the a set of finished vertices
$d_a(v, C_i)$	an average dissimilarity between a vertex v and a cluster C_i

Table 5. Notations for the vertices in a graph

4.3 A greedy re-coloring algorithm

4.3.1 Why greedy algorithm?

By definition, since each dominating vertex is connected to the vertices with all the other colors (clusters), it is far away from the other clusters (at least greater than the specified threshold θ. This means that, dominating vertices contribute to increase the inter-cluster dissimilarity, which is important for better clustering.

Likewise, by definition, each supporting vertices is necessary (important) to ``keep'' some dominating vertex, since without its color the dominance property will be lost. Thus, these vertices also contributes to increase the inter-cluster dissimilarity.

Based on the above reasons, in our current approach, the colors of the vertices in $\boldsymbol{V_d}$ and $\boldsymbol{V_s}$ are fixed (not changed) to sustain the inter-cluster dissimilarity. Furthermore, to guarantee the termination of the processing, re-coloring of vertices is tried at most once. To realize this, when a vertex is tested (checked) for re-coloring, the vertex is moved into the finished vertices $\boldsymbol{V_f}$ in order to avoid the repetition.
In summary, we consider re-coloring of the vertices in $\boldsymbol{V} \setminus \{\boldsymbol{V_d} \cup \boldsymbol{V_s} \cup \boldsymbol{V_s}\}$, namely the set of non-critical vertices $\boldsymbol{V_{nc}}$. In addition, whenever a vertex is checked for re-coloring, it is moved into the finished vertices $\boldsymbol{V_f}$ so that its color is fixed in the latter processing. Thus, since the size $|\boldsymbol{V_{nc}}|$ is monotonically decreased at each re-coloring of some vertex in $G(\boldsymbol{V},\boldsymbol{E})$, the termination of the processing is guaranteed.
Since the color of a vertex v is fixed once it is inserted into $\boldsymbol{V_c}$, and other possibilities are not explored in later processing, our algorithm works in a greedy fashion. This is important, both for reducing the time complexity of the algorithm and to guarantee its termination. Admittedly there can be other approach to solve the re-coloring problem. For instance, it might be possible to incorporate some kind of back-tracking for the re-coloring, *e.g.*, to consider further re-coloing of the vertices in $\boldsymbol{V_f}$. However, in compensation for the possibly better quality, such an approach will require much more computation time and more dedicated mechanism to guarantee the termination.

4.3.2 A vertex selection criterion

As explained in Section 4.3.1, our approach considers the vertices in $\boldsymbol{V_{nc}}$ for re-coloing. The next question is, which vertices should be considered for re-coloring and in what order. Our criterion for the vertex selection is as follows.
Among the vertices in $\boldsymbol{V_{nc}}$, we select the vertex with the maximal average within-cluster dissimilarity. Thus, the following vertex v^* is selected.

$$v^* = \arg\max_{v \in V_{nc}} d_a(v, c(v)) \tag{5}$$

Here, the value of $d_a(v,c(v))$ defined in equation (4) corresponds to the degree of "outlier" of the vertex v, because it represents the average within-cluster dissimilarity when it is assigned to the cluster $c(v)$ (note that a color also corresponds to a cluster).
On the contrary, suppose other vertex v' which is not with the maximal value in equation (4) is selected and re-colored. In that case, the size of the cluster $|C_p(v)|$ can decrease, since some other vertex might be moved into the set of critical vertices $\boldsymbol{V_c}$ due to the re-coloring of v'. This amounts to putting more constraints into the re-coloring processing and restricting the possibilities of new color for v^*. For instance, the increase in the size of neighboring vertices $|N_c(v^*)|$ means more constraints, and thus leads to decreasing the possible colors for v^*.
Based on the above argument, among the vertices in $\boldsymbol{V_{nc}}$, we select the vertex with the maximal average within-cluster dissimilarity for re-coloring.

4.3.3 A color selection criterion

After selecting a vertex as the candidate for re-coloring, the next question is, which color the vertex should be assigned. Note that our objective is to increase the quality of a partition, while preserving the constraints. The second constraint, namely the preservation of the dominating vertices is guaranteed by our vertex selection strategy in Section 4.3.1. Thus, we need to select the color which satisfy the first constraint, namely the proper coloring, and which leads to the better quality of the resulting partition.

Our color selection criterion is as follows. When the vertex v^* is selected for re-coloring, we check the colors in $C_p(v^*)$, since it represents the colors which satisfy the proper coloring constraint for v^*. Among these colors, we select the one with the maximal $Dunn_G$ in equation (3), since it evaluates the quality of the resulting partition.

4.3.4 The algorithm

We need to take into account the fact that the color of non-critical vertices $\boldsymbol{V_{nc}}$ might be changed through their re-coloring in the latter processing. This means that, reflecting the colors of $\boldsymbol{V_{nc}}$ to evaluate the quality of the current partition can be an unreliable. Thus, we exclude the non-critical vertices to evaluate the quality of the current partition in the re-coloring process, and utilize only the fixed colors in critical vertices $\boldsymbol{V_c}$.

Furthermore, when the color $c(v)$ of a vertex v is re-colored to some other color c, some vertex $v_{nc} \in \boldsymbol{V_{nc}}$ might become new critical vertices. This is because some other vertices can become dominating vertices or supporting ones, due to the re-coloring of v. To reflect the change of colors in the graph $G(\boldsymbol{V,E})$ due to the re-coloring of the vertex v, we also define a set of vertices $\boldsymbol{V_c^{tmp}}(v, c)$. $\boldsymbol{V_c^{tmp}}(v, c)$ represents the set of vertices which become *new* critical vertices induced from this re-coloring. In addition, we denote the resulting partition of $G(\boldsymbol{V,E})$ as $\boldsymbol{P}(v, c)$. Here, in $\boldsymbol{P}(v, c)$, only the originally assigned color $c(v)$ of the vertex v is re-colored to c, and the colors of the other remaining vertices are not changed. These notations are summarized in Table 6.

Symbol	Description
$\boldsymbol{V_c^{tmp}}(v, c)$	the a set of new critical vertices by changing the color of v to c
$\boldsymbol{P}(v, c)$	the new partition by changing the color of v to c

Table 6. additional notations for the algorithm

Based on the above, our greedy re-coloring algorithm is summarized as the Algorithm re-coloring() in Fig. 5. In the selection of vertex or color, there can be multiple candidates with exactly the same value. In that case, since the candidates are indistinguishable with respect to our criteria, one of them is selected at random.

4.3.5 Properties of the algorithm

The proposed algorithm has the following desirable properties for clustering. We explain the properties with their proofs in this subsection.

[Proposition 1]

Algorithm re-coloring() returns a proper coloring of $G(\boldsymbol{V,E})$ from $\boldsymbol{P}$.

Proof

Algorithm re-coloring() can change the color of a vertex v^* only to some color in $C_p(v^*)$. By definition of the function $C_p()$ in Table, all the colors in $C_p(v^*)$ satisfy the proper coloring for the vertex v^*.

[Proposition 2]

Algorithm re-coloring() returns a b-coloring of $G(\boldsymbol{V,E})$ from $\boldsymbol{P}$.

Proof

From Proposition 1, proper coloring is guaranteed. We need to show that there is at least one dominating vertex for each color. By definition, this property is satisfied in $\boldsymbol{P}$. As explained in Section 4.3.3, since Algorithm re-coloring() does not change the colors of dominating vertices nor those of the supporting vertices, there is at least one dominating vertex for each color.

Algorithm re_coloring()

Input: $G(V,E)$ //*A graph*
P //*a b-coloring partition of* $G(V,E)$
Output: C' //*another partition*

$C' := P$;
Divide V into $V_c \cup V_{nc}$
while $V_{nc} \neq \emptyset$ **do**
$v^* = \arg\max_{v \in V_{nc}} d_a(v', c(v'))$
for $c \in C_p(v^*)$ **do**
calculate $V_c^{tmp}(v^*, c)$ and $P(v^*, c)$ induced from the re-coloring of $c(v^*)$ into c;
For $\forall C_i \in P(v^*, c)$, calculate $d_a(v^*, C_i))$ w.r.t. $V_c \cup V_c^{tmp}(v^*, c)$
end for
$c^* = \arg\max_{c \in C_p(v^*)} Dunn_G(P(v^*, c))$
Re-color $c(v^*)$ into c^*;
$V_{nc} = V_{nc} \setminus \{v^*\}$;;
$V_f = V_f \cup \{v^*\}$;
$V_c = V_c \cup V_c^{tmp}(v^*, c)$;
end while
return C'

Fig.5. the greedy re-coloring algorithm

[Proposition 3]
Algorithm re-coloring() monotonically improve the quality of partition.
Proof
As explained above, the color which maximizes the quality (here, $Dunn_G$ is utilized)[is selected by modifying the originally assigned color. Note that it is allowed that the originally assigned color is selected and thus unchanged. Since this re-coloring is repeated for all the non-critical vertices, when Algorithm re-coloring() terminates, the quality of partition will be monotonically improved.

5. Evaluations

The proposed greedy algorithm (Algorithm re-coloring() in Fig.) was tested by considering two relevant benchmark data sets, viz., *Zoo*, and *Mushroom* from the UCI Machine Learning Repository (Blake & Merz, 1998). To evaluate the quality of the partition discovered by the greedy algorithm (called Improved b-coloring Partition), the results are compared with that of the best partition returned by the previous b-coloring clustering algorithm as the one maximizing the $Dunn_G$ value (denoted as original b-coloring), the Hansen's method based on minimal coloring technique (Hansen & Delattre, 1978) and the Agglomerative Single-link method (Jain et al., 1999).
In addition to the value of Generalized Dunn's index, we also evaluated the results based on a probability matching scheme called *Distinctness* (Kalyani & Sushmita, 2003). This evaluation index is useful in the cluster validation problem, since it is independent of

1. the number of clusters, and
2. the dissimilarity between objects.

For a partition $\boldsymbol{P}$ with p clusters $\{C_1,C_2,..,C_p\}$, the *Distinctness* is defined as the inter-cluster dissimilarity using a probability match measure, namely the variance of the distribution match. The variance of the distribution match between clusters C_k and C_l in a given partition is measured as:

$$Var(C_k,C_l)=\frac{1}{m}\sum_{i}^{m}\sum_{j}\left(P\left(a_i=V_{ij}|\mathrm{C_k}\right)-P\left(a_i=V_{ij}|\mathrm{C_l}\right)\right)^2 \tag{6}$$

where m is the number of attributes a_i characterizing the objects. $P(a_i=V_{ij}\,|\,C_k)$ is the conditional probability of a_i to take value V_{ij} in class C_k.

The above equation assumes that each data has only one value per attribute (represented by $j \in a_i$). The greater this value, the more dissimilar are the two clusters being compared. Thus, the concepts they represent are also dissimilar.

The *Distinctness* of a partition $\boldsymbol{P}$ is calculated as the average variance between clusters as:

$$Distinctness=\frac{\sum_{k=1}^{p}\sum_{l=1}^{p}Var(C_k,C_l)}{p(p-1)} \tag{7}$$

When comparing two partitions, the one with larger distinctness would be considered as better one, with respect to this index, since the clusters in such a partition represent more distinct concepts.

5.1 Zoo dataset

The Zoo dataset includes 100 instances of animals with 17 features and 7 output classes. The name of the animal constitutes the first attribute. There are 15 boolean features corresponding to the presence of hair, feathers, eggs, milk, backbone, fins, tail; and whether airborne, aquatic, predator, toothed, breathes, venomous, domestic, catsize. The numeric attribute corresponds to the number of legs.

Table 7 summarizes the clustering results. The Distinctness measure indicates better partitioning for the clusters generated by the b-coloring clustering approach (for the original partition as well as for the improved partition). This confirms that the utilization of dominating vertices finds more meaningful and well-separated clusters. In the other hand, the improved partition has the larger value. This indicates the pertinence of the greedy algorithm to improve the original b-coloring partition.

method	# Clusters	Distinctness	$Dunn_G$
re-coloring based	7	*0.652*	*1.120*
original b-coloring	7	0.612	1.071
agglomerative single-link	2	0.506	0.852
Hansen	4	0.547	1.028

Table 7. the result of Zoo dataset.

5.2 Mushroom dataset

Each data record contains information that describes the 21 physical properties (e.g., color, odor, size, shape) of a single mushroom. A record also contains a *poisonous* or *edible* label for the mushroom. All attributes are categorical; for instance, the values that the size attribute takes are narrow and broad, while the values of shape can be bell, at, conical or convex, and odor is one of spicy, almond, foul, fishy, pungent etc. The mushroom database has many data items (the number of data items is 8124).The number of *edible* and *poisonous* mushrooms in the data set is 4208 and 3916, respectively. There are 23 species of mushrooms in this data set. Each species is then identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one.
Table 8 summarizes the results of the clustering obtained, over the mushroom data using the different clustering approaches.

method	# Clusters	Distinctness	$Dunn_G$
re-coloring based	17	*0.728*	*0.995*
original b-coloring	17	0.713	0.891
agglomerative single-link	20	0.615	0.866
Hansen	19	0.677	0.911

Table 8. he result of Mushroom dataset.

Furthermore, we also analyzed the assigned objects in the clusters. Table 9 and Table 10 show the membership differences among the clusters by the previous b-coloring approach and the proposed approach in this paper. The clusters with bold italic characters represent the so-called *non-pure* clusters. These clusters are called non-pure, since they contain both poisonous and edible data items (mushrooms), and fail to separate them solely based on their features.

Cluster ID	# of Edible	# of Poisonous	Cluster ID	# of Edible	# of Poisonous
1	0	36	11	139	0
2	***96***	***464***	12	18	0
3	***695***	***72***	13	0	1296
4	768	0	14	224	0
5	1510	0	15	0	1728
6	220	0	***16***	***48***	***32***
7	145	0	17	192	0
8	0	288			
9	144	0			
10	9	0			

Table 9. details of cluster assignment for Mushroom dataset by the original approach

From these tables, we observe that almost all the clusters generated by both approaches are pure, except for the three clusters (Cluster 2, 3 and 16). This result also confirms that the

utilization of dominating vertex contributes to generating to more meaningful and well-separated clusters.

Cluster ID	# of Edible	# of Poisonous	Cluster ID	# of Edible	# of Poisonous
1	0	36	11	107	0
2	***96***	***464***	12	16	0
3	***475***	***72***	13	0	1296
4	768	0	14	288	0
5	1728	0	15	0	1728
6	192	0	***16***	***48***	***32***
7	145	0	17	192	0
8	0	288			
9	144	0			
10	9	0			

Table 10. details of cluster assignment for Mushroom dataset by the proposed approach

6. Discussions

In our current approach we employ a greedy strategy tackle the re-coloring problem defined in Section 4.1. The major reasons for utilizing a greedy strategy is, as in other many approaches based on some greedy algorithms, we believe that it is useful as well as crucial for handling real world data, especially for large scale data. Based on this hypothesis, both the selection of vertex to be re-colored and the selection of the color to be assigned, is conducted in greedy fashion.

The other side of our greedy algorithm is that, besides it tries to improve the quality of partition while satisfying the constraints, there can still be better solutions for the re-coloring problem. If finding out better solutions is the most important (and, the only) interest, then, it would be possible to seek other much more expensive approaches. For instance, it might be possible to incorporate some kind of back-tracking for the re-coloring of the vertices. Such a recursive approach might be useful, both for the conceptual simplicity of the algorithm as well as the quality of the obtained solutions, in compensation for the incurred computational complexity.

In addition, there are many interesting issues to pursue:

1. more experiments and comparison for our algorithm on real world datasets, and
2. extension of our re-coloring approach for the critical vertices

As for (1), medical datasets or large scale image datasets seem interesting. As for (2), relaxing the constraints on the critical vertices seems promising for finding out better partition.

7. Conclusions

This paper has proposed a new greedy algorithm to improve the quality of clustering, while satisfying the constraints in the b-coloring of a specified graph. The previous b-coloring

based clustering approach enables to build a fine partition of the data set (classical or symbolic) into clusters even when the number of clusters is not pre-defined. However, since it does not consider the quality of the clusters, besides obtaining the clusters in terms of the b-coloring of the graph, it was difficult to obtain better clusters explicitly. The proposed algorithm in this paper can complement this weakness. It conducts the re-coloring of the vertices (which correspond to data items) to improve the quality of the clusters, while satisfying the constraints. A greedy strategy is employed in the re-coloring process, both for the selection of vertex to be re-colored and the selection of the color to be assigned. We believe that utilization of a greedy strategy is useful as well as crucial for handling real world data, especially for large scale data.

The proposed greedy algorithm was tested over benchmark datasets from the UCI repository. The detailed results of the evaluations are reported and discussed. Through this evaluation, the effectiveness of the proposed greedy algorithm is confirmed. Especially, the results of experiments indicate that our approach is useful to offers a compromise between the inter-cluster separation and the intra-cluster cohesion.

8. Acknowledgments

The first author was supported by Canon Foundation in Europe Research Fellowship for his stay in France. The second author was supported by JSPS, Japan (PE 07555) for his stay in Japan. The authors are grateful to these grants. This work is partially supported by the grant-in-aid for scientific research (No. 20500123) funded by MEXT, Japan.

9. References

Bezdek, J.C. & Pal, N.R. (1998). Some new indexes of cluster validity. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 28, No.3, 1998, pp.301-315

Elghazel, H.; Deslandres, V., Hacid, M.S., Dussauchoy, A. & Kheddouci, H. (2006). A new clustering approach for symbolic data and its validation: Application to the healthcare data. *Proceedings of ISMIS2006*, pp.473–482, Springer Verlag

Elghazel, H.; Yoshida, T., Deslandres, V., Hacid, M.S. & Dussauchoy, A. (2007). A new Greedy Algorithm for improving b-Coloirng Clustering. *Proceedings of GbR2007*, pp.228-239, Springer Verlag

Guenoche, A.; Hansen, P. & Jaumard, B. (1991). Efficient algorithms for divisive hierarchical clustering with the diameter criterion. *Journal of Classification*, Vol.8, pp.5-30

Guha, S.; Rastogi, R. & Shim, K. (1998). Cure: An efficient clustering algorithm for large databases. *Proceedings of the ACM SIGMOD Conference*, pp.73-84

Hansen, P. & Delattre, M. (1978). Complete-link cluster analysis by graph coloring. *Journal of the American Statistical Association*, Vol.73, pp.397-403

Hartigan, J. & Wong, M. (1979). Algorithm as136: A k-means clustering algorithm. *Journal of Applied Statistics*, Vol.28, pp.100-108

Blake, C.L. & Merz, C.J. (1998). UCI repository of machine learning database. University of California, Irvine. http://www.ics.uci.edu/~mlearn/MLRepository.html

Irving, W. & Manlov, D. F. (1999). The b-chromatic number of a graph. *Discrete Applied Mathematics*, Vol.91, pp.127-141

Jain, A.K.; Murty, M.N. & Flynn, P.J. (1999). Data clustering: A review. *ACM Computing Surveys*, Vol.31, pp.264-323

Kalyani, M. & Sushmita, M. (2003). Clustering and its validation in a symbolic framework. *Pattern Recognition Letters*, Vol.24, No.14, pp.2367-2376

Ng, R. & and Han, J. (2002). Clarans: a method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, Vol.14, No.5, pp.1003-1016

Zahn, C.T. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, Vol.20, pp.68-86

30

WDM Optical Networks Planning using Greedy Algorithms

Nina Skorin-Kapov
Faculty of Electrical Engineering and Computing, University of Zagreb
Croatia

1. Introduction

Optical networks have been established as the enabling technology for today's high-speed communication networks. Wavelength Division Multiplexing (WDM) enables the efficient utilization of optical fibers by dividing its tremendous bandwidth into a set of disjoint wavelength bands, referred to as wavelengths. Each wavelength supports one communication channel which corresponds to an end user operating at an arbitrary speed, e.g. peak electronic speed. This helps to overcome the opto-electronic mismatch between the multiple terabit-per-second bandwidth of optical fibers and the gigabit-per-second electronic processing speeds at end users.

In wavelength-routed WDM networks, all-optical directed channels, called *lightpaths*, can be established between pairs of nodes which are not necessarily neighboring in the physical topology. A set of lightpaths creates a so-called virtual topology over the physical interconnection of fibers. Packet-switched traffic is then routed over this virtual topology, independent of the physical topology. Traffic send via a lightpath is transmitted in the optical domain with no opto-electronic conversion at intermediate nodes. Establishing a lightpath requires a transmitter and receiver at the source and destination nodes, respectively, and includes routing it over the physical topology and assigning to it a wavelength.

One of the main challenges in wavelength-routed WDM networks is to successfully solve the Virtual Topology Design (VTD) problem. This problem is usually divided into the following four sub-problems. The first is to determine the set of lightpaths which is to form the virtual topology. This set of lightpaths can be static, scheduled or dynamic. Static lightpaths are established semi-permanently and chosen on the basis of a traffic matrix representing the estimated average traffic floes between node pairs. Scheduled lightpaths, on the other hand, try to exploit the periodic nature of traffic by defining a schedule for establishing and tearing down lightpaths based on periodic traffic trends. Lastly, dynamic lightpaths are established as connection requests arrive with no *a priori* information regarding traffic demands. Unless specified otherwise, the VTD problem usually refers to the static case which we will be discussing in the remainder of this chapter. Thus, we use these terms interchangeably.

The second sub-problem in VTD is to find for each lightpath a corresponding route in the physical topology, while the third is to assign to each a wavelength subject to certain constraints. Lightpaths routed over the same physical links at the same time cannot be

assigned the same wavelength. This is called *the wavelength clash constraint*. If there are no wavelength converters available, which is often the case due to their high prices, the entire lightpath must be established the same wavelength. This is known as the *wavelength continuity constraint*. Sub-problems two and three are commonly referred to as the Routing and Wavelength Assignment (RWA) problem. The RWA problem is often solved separately with the objective to minimize wavelengths and/or lightpath congestion, or maximize the number of established lightpaths subject to a limited number of wavelengths. An example of a 4-node wavelength-routed network, an RWA scheme, and its corresponding virtual topology with five established lightpaths is shown in Fig. 1.

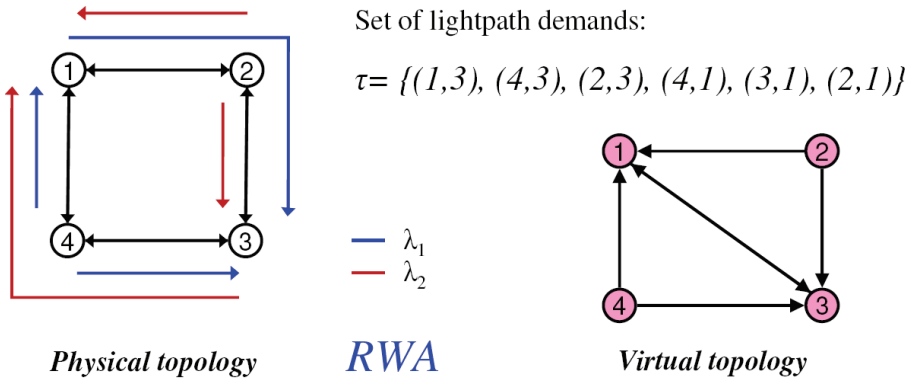


Fig. 1. An example of solving the Routing and Wavelength Assignment problem

Finally, after determining the set of lightpaths and successfully solving the RWA problem, packet-switched traffic must be routed over the virtual topology which is the fourth sub-problem in VTD. Objectives include minimizing the average packet and virtual hop distances, the number of transceivers used, and congestion.

The Virtual Topology Design problem, as well as the RWA problem, is NP-complete. Thus, heuristic algorithms are needed to find sub-optimal solutions for larger problem instances. In this chapter we discuss greedy algorithms based on bin packing for static RWA. Furthermore, we present greedy approaches for solving the first three sub-problems of Virtual Topology Design, which we refer to as the VRWA problem, in conjunction with a linear program for traffic routing (the fourth sub-problem of VTD).

2. The RWA problem

2.1 Problem definition

The Routing and Wavelength Assignment problem is as follows. Given is a graph $G=(V,E)$, where V is the set of nodes and E is the set of bidirectional edges representing a fiber in each direction. Since we are considering the static case, we are given a set of lightpath demands, $\tau = \{(s_1,d_1), \ldots, (s_n,d_n)\}$, where s_i, d_i in V, $i=1,\ldots,n$, are the source and destination nodes, respectively. These lightpaths are to be established semi-permanently. To solve the RWA problem, we need to find a set of directed paths $P=\{P_1,\ldots,P_n\}$ in G, each corresponding to one lightpath demand and assign to each a wavelength subject to the following constraints. Two paths that share a common physical link (in the same direction) cannot be assigned the same wavelength (*the wavelength clash constraint*). Furthermore, we assume that there are no

wavelength converters and thus the entire physical path corresponding to a single lightpath must be assigned a unique wavelength (*the wavelength continuity constraint*). Furthermore, we constrain the length in hops of the paths in P by a parameter H.
Our objective is to minimize the number of wavelengths needed to establish the given set of lightpath demands. A secondary objective we consider is minimizing the physical lengths of the lightpaths which is desirable due to transmission impairments and delay.

2.2 Related work

The Routing and Wavelength Assignment problem has been widely studied in the literature. This problem has been proven to be NP-complete (Chlamtac et al., 1992) and several heuristic approaches have been developed to help solve it sub-optimally. Variations have been studied, such as the static, scheduled and dynamic cases, with (un)limited wavelengths, with(out) wavelength converters and/or considering physical impairments in optical fibres ((Choi et al., 2000), (Jia et al., 2002), (Mukherjee, 1997), (Murthy & Gurusamy, 2002)).
In (Ramaswami & Sivarajan, 1995), a mixed integer linear formulation is given for the RWA problem which is highly intractable and, thus, heuristics are needed. Alternative formulations are given in (Ozdaglar & Bertsekas, 2003) which consider a quasi-static view and introduce a cost function which is such that it tends to give integral solutions even when the problem is relaxed.
Most heuristic approaches divide the problem into two sub-problems solved subsequently: the first is to route the set of lightpaths and the second is to assign wavelengths. Given a routing scheme, wavelength assignment is equivalent to the graph coloring problem so existing heuristics for graph coloring are often used. In (Banerjee & Mukherjee, 1996), the authors suggest a multi-commodity flow formulation for routing which is relaxed and then rounded using a randomized approach. Wavelength assignment is solved using graph coloring heuristics. Local random search is used to solve the routing sub-problem in (Hyytia & Virtamo, 1998) while a greedy graph coloring algorithm assigns wavelengths for the obtained routing solution. In (Noronha & Ribeiro, 2006), a tabu search algorithm suggested for color-partitioning is used to perform wavelength assignment on a set of previously calculated alternative routes. Two-step algorithms, such as those mentioned above, can give good results but may have longer execution times than one-step algorithms.
A one-step approach is suggested in (Lee et al., 2002) which gives an integer formulation solved using column generation. This, however, is not practical for larger problems. A simple yet highly efficient greedy algorithm, called *Greedy_EDP_RWA* is suggested in (Manohar et al., 2002). This approach is based on edge disjoint paths and runs as follows. The algorithm creates a partition of the set of lightpaths where each element of the partition contains a subset of the given lightpaths routed on mutually edge disjoint paths which can, thus, be assigned the same wavelength. Hence, the number of wavelengths required is equal to the number of elements in the partition. This algorithm has been shown to give better results than (Banerjee & Mukherjee, 1996) and yet is much faster. We suggested improved greedy algorithms based on bin packing in (Skorin-Kapov, 2006.a) which will be described in more detail in the next subsection. Efficient implementations of these greedy bin packing algorithms were suggested in (Noronha et al., 2008).

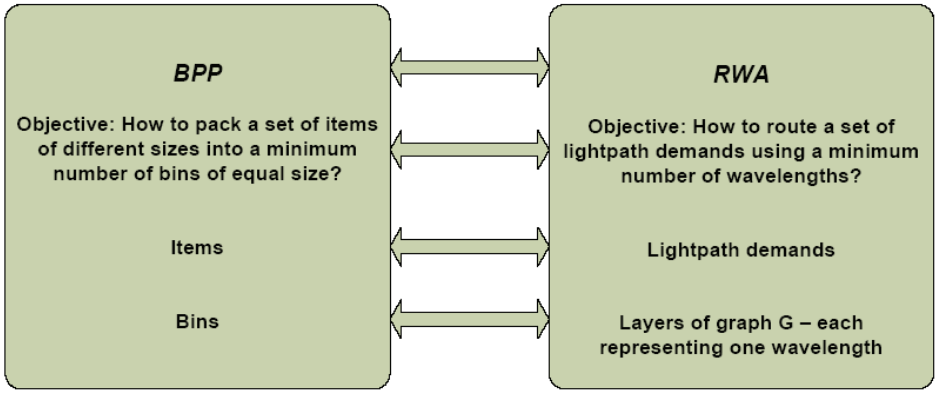


Fig. 2. Analogies between the Bin Packing Problem and Routing and Wavelength Assignment

2.3 Greedy algorithms based on bin packing

In order to efficiently solve RWA using fast greedy algorithms, we adapt classical bin packing heuristics to meet the specific demands of our problem. Bin packing is a well-known NP-hard optimization problem which attempts to pack a given set of items of various sizes into the minimum number of bins of equal size. Various heuristic algorithms have been proposed for bin packing and surveys can be found in (Coffman et el., 1996) and (Coffman et al., 2002). Widely-used greedy heuristics for this problem are the First Fit (FF), Best Fit (BF), First Fit Decreasing (FFD), and Best Fit Decreasing (BFD) algorithms. The First Fit algorithm packs items into the first bin into which it fits, while the Best Fit algorithm pack items into the bin which leaves the least room left over after including the item. Both algorithms pack items in random order, and as such can be used as *online* algorithms which pack items in the order that they appear.

The FFD and BFD algorithms, on the other hand, must have *a priori* knowledge of the entire set of items to be packed. Namely, they sort items in non-increasing order of their size and then pack them according to the FF or BF strategies, respectively. The motivation for this is that first packing the larger items, which are more difficult to pack, and then filling up remaining spaces with smaller items often lead to fewer bins needed. The FFD and BFD algorithms can only be used as *offline* algorithms since they require complete knowledge of the problem (i.e. the set of items), but give much better results than the corresponding online algorithms.

We apply these ideas to help develop efficient greedy algorithms for the static RWA problem. We call these heuristics the FF_RWA, BF_RWA, FFD_RWA, and BFD_RWA algorithms. To apply the Bin Packing Problem (BPP) to RWA, we have to define items and bins in terms of optical networks which we do as follows. Items represent lightpath demands while bins represent layers or copies of the physical topology, i.e., graph *G*, each corresponding to one wavelength. Our objective is to route all the lightpath demands on the minimum number of layers such that lightpaths routed on the same layer are edge disjoint.

2.3.1 The FF_RWA algorithm

The First Fit Routing and Wavelength Assignment (FF_RWA) algorithm runs as follows. Lightpath demands (i.e., *items*) are selected at random and routed on the lowest-indexed layer[1] of graph G (i.e., *bin*) that has a feasible path available and assigns to it the wavelength corresponding to that layer. If there is no feasible path available on any existing layer, i.e. a path shorter than the hop bound H, a new layer is added. Once a path is found, its corresponding edges are deleted, i.e., are marked as used for that wavelength. Note that, using this approach, a lightpath may be routed on a longer path on a lower-indexed layer than might be available on a higher layer. Lightpaths in RWA, as opposed to items in BPP, are not of fixed size but depend on the available links in each layer. This algorithm is basically equivalent to the *Greedy_EDP_RWA* algorithm from (Manohar et al., 2002), differing only in the order in which some steps are executed, but yielding the same results.

2.3.2 The BF_RWA algorithm

The Best Fit Routing and Wavelength Assignment (BF_RWA) algorithm also starts with a single layer and routes lightpath demands in random order. However, instead of routing lightpaths on the first layer on which there is an available path, lightpaths are routed on the layer on which it '*fits best*'. By best fit, we do not mean the layer with the least room left over as in BPP, but rather the one on which the lightpath can be routed on the shortest path. If there are multiple layers which can offer routes of the same path length, the lowest–indexed one is chosen. If there is no feasible path available on any layer, a new one is added. The main motivation for this approach is to use fewer resources for individual lightpaths leaving more room for future demands and ultimately minimizing the number of wavelengths used. Additionally, this approach helps to minimize the physical lengths of the lightpaths.

2.3.3 The FFD_RWA and BFD_RWA algorithms

The First Fit and Best Fit Decreasing Routing and Wavelength Assignment (FFD_RWA and BFD_RWA) algorithms sort the lightpath demands in non-increasing order of the lengths of their shortest paths in G and then proceed according to the FF and BF strategies, respectively. We use a lightpath's shortest path in G as a measure of its size, even though the lightpath will not necessarily be routed on this path. The motivation for this method of sorting is that if *'longer'* lightpaths (i.e. those that are harder to route) are routed first, when most resources are still available, they can be routed on their shortest paths using up less space. *'Shorter'* lightpaths are then more easily routed over the remaining links which can ultimately lead to fewer wavelengths used.

2.4 Lower bounds

To asses the value of the obtained solutions we compare with simple lower bounds which can be easily calculated even for larger problems. A lower bound on the number of wavelengths is:

$$LB_w = \max\left\{\max_{i\in V}\left\lceil\frac{\Delta_l}{\Delta_p}\right\rceil, \left\lceil\frac{\sum_{j=1}^{n} l(SP_j)}{2\,|E|}\right\rceil\right\} \quad (1)$$

[1] Initially, only one layer of G is considered.

The first element represents the maximum ratio of logical in (or out) degree Δ_l to physical in (or out) degree Δ_p rounded to the highest integer. The second element represent the sum of the lengths in hops of the shortest paths $l(SP_j)$ for all lightpath demands, divided by the total number of edges $|E|$, multiplied by 2 (since they are bidirectional).
A simple lower on the average physical lengths is simply the sum of all the shortest paths $l(SP_j)$ divide by the number of lightpaths n:

$$LB_H = \frac{\sum_{j=1}^{n} l(SP_j)}{n} \tag{2}$$

2.5 Computational results

The *Greedy_EDP_RWA* algorithm from (Manohar et al., 2002) and the BF_RWA, FFD_RWA, and BFD_RWA were implemented in C++ and run on a PC powered by a P4 2.8GHz processor.[2] Series of 5 random 100-node networks were created with average degrees of 3, 4, and 5. Sets of random lightpath requests were generated where the probability P_l of there being a lightpath between two nodes ranged from 0.2 to 1, in 0.2 increments. The upper bound on the physical hop length H was set to $max(diam(G), \sqrt{|E|})$ as in (Manohar et al., 2002). All algorithms were run with 10 different seeds for each test case.

Test Netw.	P_l	Lightpath requests	LB_W	*Greedy_EDP_RWA (FF_RWA)*	*BF_RWA*	*FFD_RWA*	*BFD_RWA*
1		2116	21	24.3 (*23,25*)	21.7 (*21,24*)	**21*** (*21,21*)	**21*** (*21,21*)
2		2081	25	28.1 (*26,30*)	25.4 (*25,27*)	**25*** (*25,25*)	**25*** (*25,25*)
3	0.2	2067	24	25.8 (*24,27*)	24.4 (*24,26*)	**24*** (*24,24*)	**24*** (*24,24*)
4		2054	29	29.9 (*29,31*)	29.4 (*29,31*)	**29*** (*29,29*)	**29*** (*29,29*)
5		2125	32	33.8 (*32,36*)	**32*** (*32,32*)	**32*** (*32,32*)	**32*** (*32,32*)
1		4063	39	44.3 (*43,46*)	39.9 (*39,43*)	**39*** (*39,39*)	**39*** (*39,39*)
2		4047	46	50.9 (*49,53*)	47.2 (*47,48*)	**47** (*47,47*)	**47** (*47,47*)
3	0.4	4064	50	53.4 (*52,55*)	50.1 (*50,51*)	**50*** (*50,50*)	**50*** (*50,50*)
4		4063	47	51.4 (*50,52*)	48.3 (*47,50*)	**47*** (*47,47*)	**47*** (*47,47*)
5		4099	50	55.8 (*53,59*)	50.3 (*50,51*)	**50*** (*50,50*)	**50*** (*50,50*)
1		6017	61	66.9 (*63,69*)	61.1 (*61,62*)	**61*** (*61,61*)	**61*** (*61,61*)
2		5995	69	74.6 (*72,78*)	69.1 (*69,70*)	**69*** (*69,69*)	**69*** (*69,69*)
3	0.6	6054	67	75.4 (*73,78*)	67.8 (*67,71*)	**67*** (*67,67*)	**67*** (*67,67*)
4		6054	71	77.5 (*75,81*)	71.2 (*71,72*)	**71*** (*71,71*)	**71*** (*71,71*)
5		6113	66	77.9 (*76,83*)	67.6 (*66,70*)	**66*** (*66,66*)	**66*** (*66,66*)
1		7960	80	86.7 (*84,89*)	80.1 (*80,81*)	**80*** (*80,80*)	**80*** (*80,80*)
2		7988	81	89.8 (*88,93*)	81.8 (*81,83*)	**81*** (*81,81*)	**81*** (*81,81*)
3	0.8	8052	88	99.4 (*96,103*)	89.8 (*88,93*)	**88*** (*88,88*)	**88*** (*88,88*)
4		8014	86	94.4 (*91,99*)	86.6 (*86,89*)	**86*** (*86,86*)	**86*** (*86,86*)
5		8017	88	101.4 (*97,106*)	88.9 (*88,90*)	**88*** (*88,88*)	**88*** (*88,88*)
1		9900	99	108.3 (*106,110*)	99.9 (*99,102*)	**99*** (*99,99*)	**99*** (*99,99*)
2		9900	99	110.7 (*108,113*)	100.4 (*99,102*)	**99*** (*99,99*)	**99*** (*99,99*)
3	1.0	9900	99	120 (*118,123*)	105 (*103,109*)	**99*** (*99,99*)	**99*** (*99,99*)
4		9900	99	110.8 (*108,112*)	99.1 (*99,100*)	**99*** (*99,99*)	**99*** (*99,99*)
5		9900	99	122.7 (*119,125*)	106.7 (*105,109*)	**103.6** (*103,104*)	104.5 (*104,105*)

Table 1. The number of wavelengths obtained by the greedy RWA algorithms and the lower bound for 100-node networks with an average degree of 4.

[2] The FF_RWA algorithm was not implemented due to its basic equivalency with *Greedy_EDP_RWA*.

Test Network	P_l	Lightpath requests	LB_{PH}	*Greedy_EDP_RWA (FF_RWA)*	*BF_RWA*	*FFD_RWA*	*BFD_RWA*
1	0.2	2116	2.92	3.86	2.97	3.87	**2.93**
2		2081	2.95	3.88	2.98	3.87	**2.96**
3		2067	2.96	3.89	2.99	3.90	**2.96***
4		2054	3.05	4.03	3.11	4.03	**3.05***
5		2125	3.23	4.25	3.28	4.26	**3.24**
1	0.4	4063	2.91	3.84	2.93	3.85	**2.91***
2		4047	2.97	3.87	2.98	3.88	**2.97***
3		4064	2.97	3.87	2.98	3.88	**2.97***
4		4063	3.05	4.02	3.10	4.00	**3.05***
5		4099	3.24	4.23	3.29	4.25	**3.26**
1	0.6	6017	2.92	3.82	2.93	3.83	**2.92***
2		5995	2.96	3.85	2.97	3.87	**2.96***
3		6054	2.98	3.87	**2.98***	3.88	**2.98***
4		6054	3.05	4.00	3.07	4.00	**3.05***
5		6113	3.24	4.23	3.29	4.23	**3.27**
1	0.8	7960	2.93	3.83	2.94	3.84	**2.93***
2		7988	2.97	3.84	2.98	3.85	**2.97***
3		8052	2.98	3.86	2.99	3.87	**2.98***
4		8014	3.05	4.00	3.08	4.00	**3.05***
5		8017	3.24	4.22	3.27	4.23	**3.26**
1	1	9900	2.94	3.83	**2.94***	3.83	**2.94***
2		9900	2.97	3.85	**2.98**	3.85	**2.98**
3		9900	2.98	3.86	2.99	3.86	**2.98***
4		9900	3.05	4.00	3.07	4.00	**3.05***
5		9900	3.24	4.20	3.28	4.22	**3.27**

Table 2. The average lightpath length (in hops) of the solutions obtained by the greedy RWA algorithms and the lower bound for 100-node networks with an average degree of 4.

In Table 1, the average number of wavelengths of the solutions obtained by the implemented algorithms and the lower bounds for networks with an average degree of 4 are shown. Furthermore, the lowest and highest values for each test case are shown in parenthesis while the best obtained solutions among the tested algorithms are marked in bold. Those solutions which are equal to the lower bound, i.e. that are known to be optimal, are marked as '*'. We can see that the FFD_RWA and BFD_RWA algorithms significantly outperform *Greedy_EDP_RWA* and BF_RWA and give optimal solution for all but two test cases.

In order to further asses the quality of the obtained solutions, we recorded the average path lengths of the lightpaths established for each test case. Table 2 shows the results for networks with an average degree of 4. We can see that here the 'Best Fit' strategy helps obtain significantly shorter lightpaths than the 'First Fit' strategy, while the BFD_RWA algorithm gives the best results in all test cases. The results for networks with average degree of 3 and 5 are omitted for lack of space but can be found in (Skorin-Kapov, 2006.a).

Although all four algorithms are very fast and tractable, running under half a second for the cases tested, the *Greedy_EDP_RWA* and BF_RWA are slightly faster than the FFD_RWA and BFD_RWA algorithms due to the time spent sorting the lightpaths in the latter. However, as a result of sorting lightpaths, FFD_RWA and BFD_RWA usually give the same results for any order of lightpaths (unless all lightpaths are of the same length) and thus only need to

be run once, while *Greedy_EDP_RWA* and BF_RWA should be run as multi-start algorithms in order to obtain good solutions.

3. The VTD problem

3.1 Problem definition

The Virtual Topology Design problem includes determining the set of lightpaths to be established on the basis of a traffic matrix, performing RWA, and lastly routing packet-switched traffic over the established virtual topology. Given is the a graph $G=(V, E)$ representing the physical topology and a long-term traffic matrix Λ representing the estimated average traffic flows between pairs of nodes. Furthermore, we have given a limited number of transmitters and receivers, commonly referred to as transceivers T, a maximum number of wavelengths W, as well as an upper bound on the number of hops H in the physical paths of lightpaths.

Various objectives can be considered. The most common optimization criteria used for Virtual Topology Design are the minimization of congestion and average packet hop distance. Congestion is defined as the maximum traffic load on any lightpath. The average packet hop distance is the average number of lightpaths a packet or unit of traffic traverses on its way from source to destination. Traversing multiple lightpaths incurs additional delay due to opto-electronic and electro-optic conversion encountered when going from one lightpath to the next. Both congestion and average packet hop distance are functions of the virtual topology and the traffic matrix, while they are independent of the physical topology and RWA scheme.

An objective criterion which has been gaining more and more attention lately is the minimization of transmitters and receivers since they make up for most of the network cost. An additional objective was proposed in (Skorin-Kapov, 2007), called the virtual hop distance, which minimizes the average hop distance between any two nodes in the virtual topology. Minimizing this criterion ensures that the virtual topology is well connected for all node-pairs, which can postpone costly reconfiguration in case of changing traffic trends. Minimizing the physical lengths of lightpaths is also desirable due to delay and, more importantly, physical impairments which can cause signal degradation. Considering all these objectives and their trade-offs is important to successfully solving the VTD problem.

3.2 Related work

Several approaches have been proposed to solve VTD or a combination of its sub-problems using mixed-integer linear formulations (MILPs) with various constraints. A formulation for complete VTD with the objective to minimize the average packet hop distance with full wavelength conversion is given in (Banerjee & Mukherjee, 2000). Heuristics for the same problem are given in (Mukherjee et al., 1996). The problem with no wavelength conversion is formulated in (Ramaswami & Sivarajan, 1996) with the objective to minimize congestion, but with no a constraint on the number of wavelengths available. Since the formulation is intractable for larger problems, the authors suggest various heuristic algorithms. One of them is the LP Logical Design Algorithm (LPLDA) which solves a relaxation of the proposed MILP and rounds the virtual topology variables; RWA is not considered. Alternative rounding schemes to obtain better solutions from LP-relaxations were proposed in (Skorin-Kapov, 2007).

Another heuristic suggested in (Ramaswami & Sivarajan, 1996), which is best-known, is the Heuristic Topology Design Algorithm (HLDA). HLDA is a greedy algorithm for the VRWA problem with a limited number of wavelengths and no wavelength conversion. Recall that Virtual topology and Routing and Wavelength Assignment (VRWA) problem consists of the first three sub-problems in Virtual Topology Design. The fourth sub-problem, Traffic Routing (TR), is solved subsequently using an LP formulation with the objective to minimize congestion. HLDA attempts to establish lightpaths between nodes in decreasing order of their estimated traffic, where each lightpath is routed on its shortest path and assigned the lowest-indexed wavelength available. After establishing a lightpaths, the value of its corresponding traffic is decreased by the value of the next highest traffic demand (or set to zero if the next highest traffic demand is higher) and then the traffic demands are re-sorted. This enables multiple lightpaths to be established between pairs of nodes with high traffic. Once the procedure ends, additional lightpaths are set up at random between nodes with left-over transmitters and receivers. This algorithm is simple, and yet performs very well with respect to congestion for which it was tested.
In (Krishnaswamy & Sivarajan, 2001), a MILP formulation for VTD including a limit on the number of wavelengths and allowing no wavelength conversion is given. Since the formulation is intractable, its relaxation is solved iteratively 25 times using a cutting plane, after which the lightpath selection and lightpath routing variables are rounded. Wavelength assignment is performed subsequently using a heuristic, while traffic routing over lightpaths is solved with an LP composed of only the traffic constraints from their MILP for VTD. This method gives good results but can be computationally prohibitive and does not guarantee a solution with the constrained number of wavelengths due to the subsequent wavelength assignment heuristic.
In (Zang & Acampora, 1995), the VRWA problem is solved by constraining potential lightpath routes to their shortest paths, and then assigning wavelength subsequently to as many lightpaths as possible in descending order or traffic, subject to the wavelength clash and continuity constraints. This approach utilizes resources well, but significantly limits possibilities by using predetermined shortest paths. In (Puech et al., 2002) a method to reduce the complexity of the first and last sub-problems of Virtual Topology design, i.e. lightpath selection and traffic routing, are given. In (Kuri et al., 2002), a tabu-search algorithm for lightpath selection and traffic routing is presented, while the trade-offs concerning cost and congestion are studied.

3.3 Greedy algorithms each aimed to optimize different objective criteria

Due to the many aspects and evaluation criteria important for VTD and its sub-problems, it is challenging to develop heuristics which perform well for all criteria. We propose 4 greedy heuristics for the VRWA problem (Skorin-Kapov, 2008), each aimed to optimize various optimization criteria, and then solve Traffic Routing using an LP formulation from (Krishnaswamy & Sivarajan, 2001) which minimizes congestion.

3.3.1 The TSO_SP algorithm

The first greedy algorithm considers Traffic Sorted Overall and routes it on the Shortest Path available (the TSO_SP algorithm). A layered graph approach is used, as in the bin packing

algorithms for RWA, but with a limited number of layers *W*. First the traffic demands are sorted in non-increasing order giving us an ordering of node-pairs which will be considered as potential lightpath demands. For each node pair in the defined order, a lightpath is established on the layer on which there is the shortest path available. If there is no feasible route available on any of the *W* layers, the lightpath between the node-pair in question is simply not established. If a lightpath is set up, the links along its path are deleted from the corresponding layer, i.e. are marked as used. This approach is similar to HLDA except that multiple lightpaths are not established between pairs of nodes and the transmitters and receivers not used initially are not subsequently assigned to random lightpaths since one of our objectives is to minimize transceiver cost.

3.3.2 The TSO_FS algorithm

The TSO_FS algorithm also Sorts Traffic Overall, but routes lightpaths on the First Satisfactory path available. Basically traffic demands are sorted in non-increasing order and corresponding potential lightpaths are routed on the lowest-indexed layer on which there is a satisfactory path available. We consider a path satisfactory of its length is less than the upper bound on the hop length *H*. If there is no satisfactory path available, the lightpath is dropped. The motivation for 'filling up' lower-indexed layers is to leave higher layers empty and potentially minimize the total number of layers used, i.e. the total number of wavelengths used.

3.3.3 The TSBS_SP algorithm

In the TSBS_SP algorithms, Traffic is Sorted By Source and routed on the Shortest Path available. Instead of sorting traffic between all node pairs, or potential lightpaths, in non-increasing order we do the following. For each node separately, we sort the traffic demands originating from that node to all other nodes (i.e. the row in the traffic matrix corresponding to the node in question) in non-increasing order. Then we make a single ordering of traffic demands, i.e. node pairs, by taking the highest traffic demand from each node, starting with the highest one overall and continuing in decreasing order. Then we take the next highest traffic demand, and the third, and so on until all traffic demands are included in the list. An example of such a method of sorting is shown in Fig. 3. Once the traffic demands are sorted, the algorithm tries to establish lightpaths in the specified order by routing them on the layer with the shortest path available. The motivation for this approach, with respect to sorting the lightpaths, is to create a virtual topology which is spread out more evenly and not only concentrated around a few nodes with very high traffic. This could lower the average virtual hop distance as well as prevent unconnected virtual topologies when resources are very scarce which can cause traffic to be blocked between certain nodes, i.e. giving infeasible solutions to the VTD problem.

3.3.4 The TSBS_FS algorithm

The TSBS_FS algorithm also considers Traffic Sorted By Source but routes lightpaths on the First Satisfactory path available. Basically, after sorting the node pairs according to the TSBS strategy, lightpaths are established on the lowest-indexed layer that has a satisfactory path available.

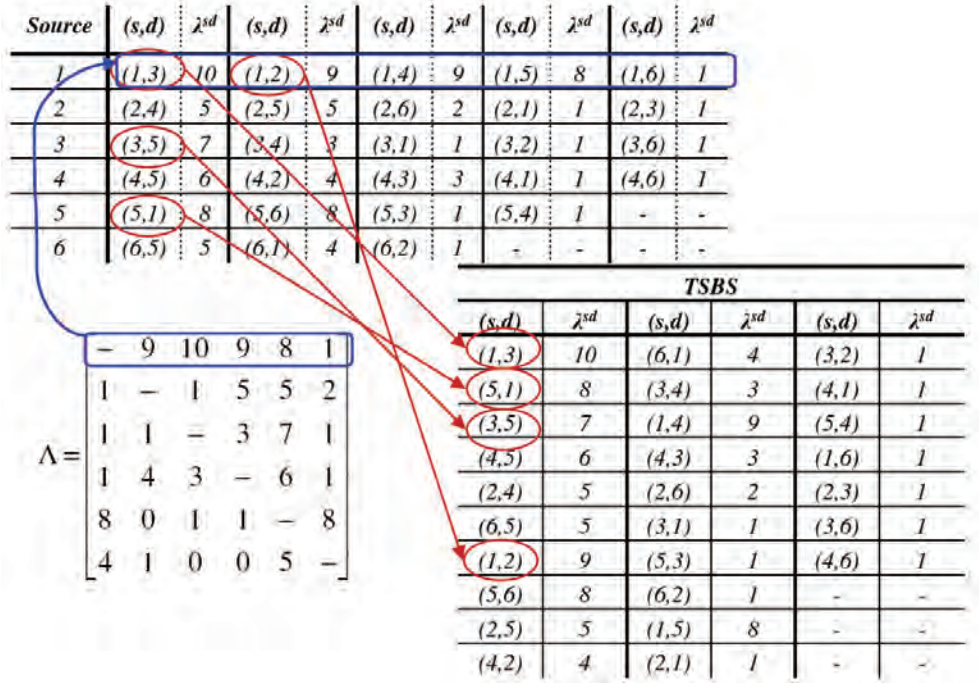


Fig. 3. An example of sorting a traffic matrix using the TSBS method.

3.4 Lower bounds

Lower bounds on the average packet hop distance and congestion were developed in (Ramaswami & Sivarajan, 1996) and are as follows. Assuming $P = (p_{sd})$ is the average traffic distribution matrix, where p_{sd} is the probability that there is a packet from s to d, Π_i for $1 \le i \le N$ is a permutation of $(1,2,\ldots,N)$ such that $p_{i\,\Pi i\,(j)} \ge p_{i\,\Pi i\,(j')}$ if $j \le j'$. If Δ_l is the maximum degree of the virtual topology, the lower bound on the average packet hop distance was shown to be

$$\overline{H_p^{LB}} = \sum_{i=1}^{N} \sum_{k=1}^{m} \sum_{j=n_{k-1}+1}^{N-1} p_{i\pi_i(j)} \tag{3}$$

where m is the largest integer such that

$$N > 1 + \Delta_l + \ldots + \Delta_l^{m-1} = \frac{\Delta_l^m - 1}{\Delta_l - 1} \tag{4}$$

and

$$n_k = \sum_{i=1}^{k} \Delta_l^i, \text{for } i \le k \le m-1, n_m = N-1, n_0 = 0\,. \tag{5}$$

Since we consider a limited number of wavelengths W on each link, the virtual degree cannot exceed $W^*\Delta_p$, where Δ_p is the maximum degree of the physical topology, we define the maximum degree of the virtual topology Δ_l to be

$$\Delta_l = min(Tr, W \cdot \Delta_p). \tag{6}$$

Using the lower bound for the average packet hop distance described above, a lower bound on congestion was derived in (Ramaswami & Sivarajan, 1996) as

$$\lambda_{max}^{LB} = \frac{r \cdot \overline{H_p^{LB}}}{E}, \tag{7}$$

where r is the total arrival rate of packets to the network and E is the number of directed links in the virtual topology.

A lower bound on the average virtual hop distance was derived in (Skorin-Kapov, 2007) as follows. Since the average virtual hop distance is independent of the traffic matrix, the lower bound on the average virtual hop distance from any node s in V to all the other nodes in the network is the same for each node s. As noted in (Ramaswami & Sivarajan, 1996), if a network has a maximum logical degree of Δ_l , for some node s in V there can be at most Δ_l nodes one hop away from s, at most Δ_l^2 nodes two hops away, at most Δ_l^3 nodes three hops away, etc. An ideal virtual topology with respect to virtual hop distance from some node s to the remaining nodes in the network would be such a topology in which node s had Δ_l neighbors, each of which had Δ_l neighbors of their own without creating a cycle, and so on, until all the nodes were connected.

Let m be the largest integer such that $N \geq 1+ \Delta_l +\ldots+ \Delta_l^{m-1} = (\Delta_l^m - 1)/(\Delta_l - 1)$ holds. In the ideal virtual topology with respect to virtual hop distance from node s, Δ_l nodes would be one hop away from s, Δ_l^2 nodes would be two hops away, etc., up until Δ_l^{m-1} nodes that would be $(m-1)$ hops away. The remaining $(N-1)-(\Delta_l +\ldots+ \Delta_l^{m-1})$ nodes would be m hops away. It follows that the lower bound on the average virtual hop distance would be

$$\overline{H_v^{LB}} = \frac{\sum_{k=1}^{m-1} k\Delta_l^k + m[(N-1) - \sum_{k=1}^{m-1}\Delta_l^k}{N-1} = \frac{\Delta_l\left[\frac{(m-1)\Delta_l^m - m\Delta_l^{m-1} + 1}{(1-\Delta_l)^2}\right] + m\left(N - \frac{\Delta_l^m - 1}{\Delta_l - 1}\right)}{N-1} \tag{8}$$

Lower bounds on the number of wavelengths, transceivers and average physical hop lengths of the lightpaths are not relevant for our particular problem, i.e., they would be zero since there is no minimum number of lightpaths which must be established.

3.5 Computational results

The greedy algorithms for the VRWA problem described above were implemented in C++ and run on a PC with a P4 2.8 GHz processor. CPLEXv6 solver was used to solve the LP for Traffic Routing. The algorithms were tested on a 14-node reference European core network topology from (Inkret et al., 2003) shown in Fig. 3. The algorithms were tested for two different traffic matrices, p1 and p2, used in (Ramaswami & Sivarajan, 1996) and (Krishnaswamy & Sivarajan, 2001) to test VTD. In traffic matrix p1, most of the traffic is

concentrated around 42 pairs, while traffic in p2 is more evenly distributed. The number of transmitters and receivers per node ranged from T= 2 to 13 each, while the number of wavelengths ranged from $W=T-1$ to $W=T+1$. The upper bound on the number of physical hops was set to $H = max(diam(G), \sqrt{|E|})$ as for the RWA problem in Section 2.

Fig. 3. A reference European core topology from (Inkret et al., 2003).

In Fig. 4, the (a) congestion, (b) average packet hop distance, (c) average virtual hop distance, and (d) number of transceivers used in the solutions obtained for traffic matrix p2 are shown. The results for traffic matrix p1 are similar and are, thus, omitted for the sake of brevity.[3] All four algorithms give similar results for congestion (Fig. 4.(a)), most of which are very close to the lower bound. The exception is for cases with a very small number of transceivers and wavelengths where the algorithms, particularly the TSO algorithms, gave unconnected virtual topologies. For the average packet hop distance, the bound is not very tight so it is difficult to assess their quality (Fig. 4.(b)). Still, we can see that the TSO algorithms tend to perform slightly better than the TSBS algorithms. On the other hand, the TSBS algorithms give better results than the TSO algorithms with respect to the virtual hop distance (Fig. 4.(c)). Here, the bound is fairly tight so we can see that the results are at least near-optimal. However, to establish better connected virtual topologies, the TSBS algorithms use more transceivers than the TSO algorithms (Fig. 4.(d)).

For the European core network, the algorithms usually terminated when all of the available wavelengths were exhausted, and as a result the same number of distinct wavelengths was used by all the algorithms. Furthermore, since the network is fairly small, the physical paths could not differ significantly between solutions. To assess how the algorithms behave with respect to RWA, further testing was done on 5 randomly generated 30-node networks,

[3] Additional numerical results for these algorithms can be found in (Skorin-Kapov, 2008)

where the probability of there being an edge between two nodes was set to $P_l=0.2$ creating fairly dense networks. Traffic matrices were generated using the method suggested in (Banerjee & Mukherjee, 2000) where a fraction F of the traffic is uniformly distributed over $[0,C/a]$, while the remaining traffic is uniformly distributed over $[0, C*\psi/a]$. The values were set to $C=1250$, $a=20$, $\psi=10$ and $F=0.7$ as in (Banerjee & Mukherjee, 2000).

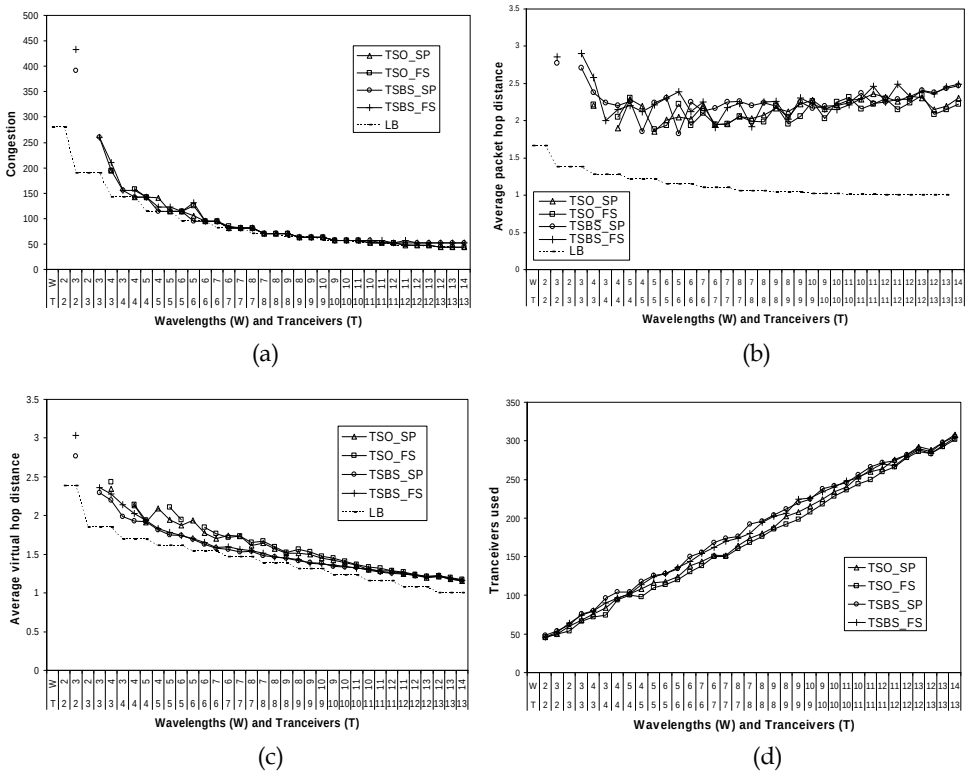
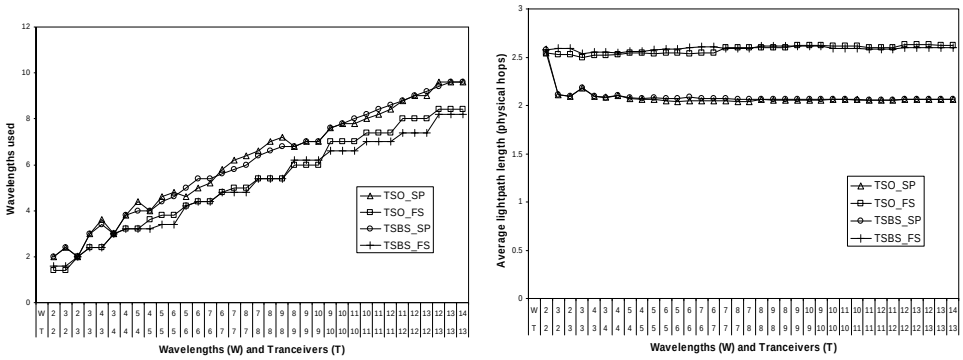


Fig. 4. The (a) congestion, (b) average packet hop distance, (c) average virtual hop distance, and (d) transceivers used by the proposed algorithms for the reference European core network for traffic matrix p2.

The number of wavelengths used and the lengths of the physical paths are shown in Figs. 5 (a) and (b), respectively. We can see that the FS algorithms use significantly fewer wavelengths than the SP algorithms (Fig. 5 (a)). The results for congestion, average packet and virtual hop distance, and the number of transceivers were almost the same for all algorithms indicating that to establish virtual topologies which perform equally well, the FS algorithms use fewer wavelengths leaving more room for expansion of the virtual topology However since the FS algorithms route paths on the first satisfactory path and not the shortest path, they tend to establish longer lightpaths (Fig. 5 (b)).

From the obtained results, we can see that when sorting traffic demands differently (i.e., TSO vs. TSBS), the TSO algorithms obtain slightly better results for congestion and packet hop distance while TSBS obtains better connected virtual topologies overall. Creating better

connected topologies might be desirable if traffic is prone to change since it can perform well, not only for current traffic trends, but for changing traffic. However, this is a trade-off with cost since establishing well-connected virtual topologies usually requires more transceivers, raising the network cost. Furthermore, if transceivers are very scarce, TSBS could help prevent from establishing unconnected virtual topologies which leave some node-pairs completely disconnected.



(a) (b)

Fig. 5. The (a) number of wavelengths used and the (b) physical lengths of lightpaths for the 30-node networks.

The method of routing and assigning wavelengths (i.e., FS vs. SP) does not significantly affect the objective criteria which are functions of the virtual topology (i.e., congestion, average packet and virtual hop distances). The main advantage of the FS algorithms over the SP algorithms is that they use fewer distinct wavelengths for RWA, particularly in dense networks. However, this is a trade-off with the physical length of lightpaths, which might be critical due to physical impairments.

4. Current and future work

Our current work on Routing and Wavelength Assignment is based on developing routing and wavelength assignment schemes aimed to increase robustness against malicious crosstalk and jamming attacks in transparent optical networks. While faults (i.e., component malfunctions) only affect the connections passing directly through them, attacks can spread and propagate throughout the network, making them more destructive and harder to locate and isolate. Our objective is to arrange lightpath demands in such a way as to minimize the propagation capabilities of such attacks. Furthermore, we aim to minimize the upper bound on the number of wavelengths required for successful wavelength assignment and reduce lightpath congestion. We are currently developing greedy algorithms based on bin packing for attack-aware wavelength assignment. Future work will include extending it to include the routing sub-problem to help obtain improved solutions for the RWA problem.

Furthermore, we are investigating the problem of scheduled Virtual Topology Design. Recall that scheduled VTD involves defining an *a priori* schedule for setting up and tearing down lightpaths based on periodic traffic trends. We proposed efficient greedy algorithms for scheduled RWA in (Skorin-Kapov, 2006.b) which route and assign wavelengths to

lightpaths according to a predefined schedule. Currently, in collaboration with P. Pavon-Marino et al. from UPCT, Spain, we are focused on developing algorithms for lightpath selection and scheduling, as well as traffic routing. Preliminary testing of our MILP formulation for the problem was performed using MatPlan WDM (Pavon-Marino et al., 2007). Since this formulation is intractable for larger problem instances, we are working on greedy heuristic approaches to find suboptimal solutions.

5. Conclusions

WDM optical network planning, particularly Virtual Topology Design, is a complex problem and several variations can be considered. Since even the sub-problems of VTD themselves are hard, solving the combined problem for larger instances using exact methods is infeasible. Greedy algorithms have been shown to obtain solutions comparable to those of more complex algorithms in very short time. In the first part of this chapter, we discuss highly-efficient greedy approaches for the static Routing and Wavelength Assignment problem based on bin packing. Suggested are methods of sorting and routing lightpaths which not only reduce the required number of wavelengths, but also reduce the average physical length of established lightpaths. Numerical results indicate that the proposed methods obtain optimal or near-optimal solutions in many cases, and significantly outperform efficient existing algorithms for the same problem. Furthermore, the heuristics are robust and highly tractable and can, thus, be used to solve large problem instances in reasonable time.

In the second part of the chapter, we propose greedy algorithms for the first three sub-problems of Virtual Topology Design, i.e. lightpath selection and RWA, which we call VRWA. Traffic routing is solved subsequently using a linear programming formulation. The greedy algorithms differ with respect to the order in which lightpaths are established, and the method of routing and assigning wavelengths. These variations are intended to improve the performance of the algorithms with respect to different objective criteria such as congestion, average virtual, physical, and packet hop distances, and the number of transceivers and distinct wavelengths used. The fact that they are fast and simple, and can be tailored to meet the needs of the network in question, makes them very attractive for practical use. In general, greedy algorithms have been shown to be very promising candidates for solving complex optical networks planning problems and will play a key role in our future work on scheduled VTD and attack-aware RWA.

6. References

Banerjee, D. & Mukherjee, B. (1996), A Practical Approach for Routing and Wavelength Assignment in Large Wavelength-Routed Optical Newtorks, *IEEE Journal of Selected Areas in Communications*, Vol. 14, (June 1996) pp. 903-908.

Chlamtac, I.; Ganz, A. & Karmi, G. (1992). Lightpath communications: An approach to high-bandwidth optical WANs, *IEEE Transactions on Communications,* Vol. 40, (1992) pp. 1171-1182.

Choi, J. S.; Golmie, N.; Lapeyere, F. ; Mouveaux, F. & Su, D. (2000). A Functional Classification Of Routing and Wavelength Assignment Schemes in DWDM Networks: Static Case, *Proceedings of VII Int. Conf. on Optical Communication and Networks*, Nagoya, Japan, Jan. 2000.

Coffman, E. G.; Garey, M. R. & Johnson, D. S. (1996). Packing Approximation Algorithms: A Survey, In: *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum (Ed.), PWS Publishing Co., Boston, MA.

Coffman, E. G.; Csirik, J. & Woeginger, G. (2002). Bin Packing Theory, In: *Handbook of Applied Optimization*, Pardalos, P. & Resende, M. G. C. (Eds.), Oxford University Press, New York.

Hyytia, E. & Virtamo, J. (1998). Wavelength assignment and routing in WDM networks,
Nordic Telegraffic Seminar 14 (1998) pp. 31-40.

Inkret, R.; Kuchar, A. & Mikac, B., *Advanced Infrastructure for Photonic Networks: Extended Final Report of COST Action 266*, Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia, pp.19-21.

Jia, X.; Hu, X.-D. & Du, D.-Z. (2002). *Multiwavelength Optical Networks*, Kluwer Academic Publishers, Norwell, MA.

Krishnaswamy, R. M. & Sivarajan, K. N. (2001). Design of logical topologies: a linear formulation for wavelength-routed optical networks with no wavelength changers, *IEEE/ACM Transactions on Networking*, Vol. 9, No. 2, (April 2001), pp. 186-198.

Kuri, J.; Puech, N. & Gagnaire, M. (2002). A Tabu search algorithm to solve a logical topology design problem in WDM networks considering implementations costs, *Proceedings of SPIE Asian Pacific Optical Conference*, Shangai, China, Oct. 2002.

Lee, K.; Kang, K. C.; Lee, T. & Park, S. (2002). An Optimization Approach to Routing and Wavelength Assignment in WDM All-Optical Mesh Networks without Wavelength Conversion", *ETRI Journal*, Vol 24, No. 2, (2002) pp. 131-141.

Manohar, P.; Manjunath, D. & Shevgaonkar, R. K. (2002). Routing and Wavelength Assignment in Optical Networks From Edge Disjoint Paths Algorithms", *IEEE Communication Letters*, Vol. 6, No. 5, (May 2002) (pp. 211-213).

Mukherjee, B. (1997). *Optical Communication Networks,* McGraw-Hill, New York.

Murthy, C. S. R. & Gurusamy, M. (2002), *WDM Optical Networks: Concepts, Design, and Algorithms*, Prentice Hall, New Jersey.

Mukherjee, B.; Banerjee, D.; Ramamurthy, S. & Mukherjee, A. (1996). Some Principles for Designing a Wide-area WDM Optical Network, *IEEE/ACM Transactions on Networking*, Vol. 4, No. 5, (Oct 1996), pp. 684-696.

Noronha, T. F. & Ribeiro, C. C. (2004). Routing and wavelength assignment by partition coloring, *European Journal of Operational Research*, Vol 171, No. 3 (June 2006, available online Dec. 2004) pp. 797-810.

Noronha, T.F.; Resende, M.G.C.& Ribeiro C.C. (2008). Efficient implementations of heuristics for routing and wavelength assignment, *Proceedings of 7th International Workshop on Experimental Algorithms (WEA 2008),* C.C. McGeoch (Ed.), LNCS, Springer, vol. 5038,(2008) pp. 169-180.

Ozdaglar, A. & Bertsekas, D. (2003). Routing and Wavelength Assignment in Optical Networks", *IEEE/ACM Transactions on Networking*, Vol 11, No. 2, (April 2003) pp. 259-272.

Pavon-Marino, P.; Aparicio-Pardo, R.; Moreno-Munoz, G.; Garcia-Haro, J. & Veiga-Gontan, J. (2007). MatPlan WDM: An Educational Tool for Network Planning in Wavelength-Routing Networks, *Lecture Notes in Computer Science*, 4534, Springer-Verlag, pp. 58-67.

Puech, N.; Kuri, J. & Gagnaire, M. (2002). Topological Design and Lightpath Routing in WDM Mesh Networks: A Combined Approach, *Photonic Network Communications*, Vol. 4, No. 3/4, (July 2002) pp. 443-456.

Ramaswami, R. & Sivarajan, K. N. (1995), Routine and Wavelength Assignment in All-Optical Networks, IEEE/ACM Transactions on Networking, Vol. 3, No. 5, (Oct. 1995), pp. 489-500.

Ramaswami, R. & Sivarajan, K. N. (1996), Design of Logical Topologies for Wavelength-Routed Optical Networks, *IEEE Journal of Selected Areas in Communications*, Vol. 14, No. 5, (June 1996), pp 840-851.

Skorin-Kapov, N. (2006). Heuristic Algorithms for the Routing and Wavelength Assignment of Scheduled Lightpath Demands in Optical Networks, *IEEE Journal of Selected Areas in Communications*, Vol. 24, No. 8, (August 2006) , pp. 2-15.

Skorin-Kapov, N. (2007a). Routing and Wavelength Assignment in Optical Networks Using Bin Packing Based algorithms, *European Journal of Operational Research*, Vol. 177, Issue 2, (March 2007) , pp. 1167-1179.

Skorin-Kapov, N. (2007b), A New Objective Criterion and Rounding Techniques for Determining Virtual Topologies in Optical Networks, *IEEE Communication Letters*, Vol. 11, No. 6, (June 2007), pp.540-542.

Skorin-Kapov, N. (2008). Virtual Topology Design in WDM Optical Networks: Greedy Algorithms and a Study of Various Objectives, (Submitted to: Telecommunications Systems).

Zang, Z. & Acampora, A. S. (1995). A Heuristic Wavelength Assignment Algorithm for Multihop WDM Networks with Wavelength Routing and Wavelength Re-Use, *IEEE/ACM Transactions on Networking*, Vol 3, No. 3, (June 1995) pp. 281-288.