Bo Chen
Mike Paterson
Guochuan Zhang (Eds.)

LNCS 4614

Combinatorics, Algorithms, Probabilistic and Experimental Methodologies

First International Symposium, ESCAPE 2007
Hangzhou, China, April 2007
Revised Selected Papers

Lecture Notes in Computer Science 4614

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Bo Chen Mike Paterson Guochuan Zhang (Eds.)

Combinatorics, Algorithms, Probabilistic and Experimental Methodologies

First International Symposium, ESCAPE 2007
Hangzhou, China, April 7-9, 2007
Revised Selected Papers

Volume Editors

Bo Chen
University of Warwick, Warwick Business School
DIMAP - Centre for Discrete Mathematics and its Applications
Coventry, CV4 7AL, UK
E-mail: B.Chen@warwick.ac.uk

Mike Paterson
University of Warwick, Department of Computer Science
DIMAP - Centre for Discrete Mathematics and its Applications
Coventry CV4 7AL, UK E-mail: M.S.Paterson@warwick.ac.uk

Guochuan Zhang
Zhejiang University
Department of Mathematics
Hangzhou 310027, China
E-mail: zgc@zju.edu.cn

Library of Congress Control Number: 2007936169

CR Subject Classification (1998): F.2.1-2, F.1.2, H.2.8, I.2.8, C.2, J.1

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-540-74449-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-74449-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12111084 06/3180 5 4 3 2 1 0

Preface

With the Internet, bio-technology, video digital data processing, as well as global geometric data applications, the enormous sizes of datasets have been pushing our limits in data processing. Both theoreticians and practitioners are challenged to an extent that has never been seen before. Growing pressure has been put on different experts, such as computer scientists, combinatorics experts, and statisticians, dealing with their own large data processing problems, to reach out beyond their own disciplines, to search for ideas, methodologies, and tool boxes, to find better, faster and more accurate solutions.

The International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies (ESCAPE) is intended to provide an interdisciplinary forum for researchers across their discipline boundaries to exchange their approaches, to foster innovative ideas as well as to develop a research agenda of common interest. The novelty of ESCAPE is to study practical, large data processing problems with different, and eventually converging, methodologies from major important disciplines.

This volume contains the proceedings of the first ESCAPE symposium, which was held in Hangzhou, China, April 7–9, 2007. We received a total of 362 submissions, of which 184 (50.8%) entered the formal review process and the remaining 178 were declined at an initial screening process as being out-of-scope. Each of the submitted papers that entered the formal review process was reviewed by at least three members of the Program Committee or their deputies on the quality, originality, soundness, and significance of its contributions. Finally 56 papers were accepted: an overall acceptance rate of 15.5%. This proceedings volume contains the 46 papers that were presented at the symposium.

Of the accepted papers, four were selected by the Award Panel as finalists for the Best Paper Award, and these were presented in a special plenary session, after which the final winner was identified. The winner also received a cash prize of US$ 1,000. In this volume the winner's paper appears first, followed by those of the other three finalists.

The huge success of the workshop resulted from the input of many people. We would like first of all to thank all the members of the Program Committee and the Award Panel for their expert evaluation of the submissions. Their service greatly helped raise the academic standard of the symposium.

The local organizers did an extraordinary job, for which we are very grateful. We thank the Department of Mathematics, Zhejiang University, and the National Natural Science Foundation of China for their financial support and for providing an excellent location for the symposium.

April 2007

Bo Chen
Mike Paterson
Guochuan Zhang

Symposium Committees

Program Committee

Susanne Albers	University of Freiburg, Germany
David Avis	McGill University, Canada
Yossi Azar	Tel-Aviv University, Israel and Microsoft Research, USA
Allan Borodin	University of Toronto, Canada
Rainer E. Burkard	Technical University of Graz, Austria
Bo Chen (Chair)	University of Warwick, UK
Guoliang Chen	University of Science and Technology of China
Vladimir Deineko	University of Warwick, UK
Xiaotie Deng	City University of Hong Kong, China
Dingzhu Du	University of Texas at Dallas, USA
Patrick Dymond	York University, Canada
Peter Eades	University of Sydney, Australia
Thomas Erlebach	University of Leicester, UK
Mike Fellows	University of Newcastle, Australia
Ronald Graham	University of California at San Diego, USA
Peter Hammer	Rutgers Center for Operations Research, USA
Winfried Hochstättler	University of Hagen, Germany
Jeff Hong	Hong Kong University of Science and Technology, China
Hiroshi Imai	University of Tokyo, Japan
Kazuo Iwama	Kyoto University, Japan
Klaus Jansen	University of Kiel, Germany
David Kirkpatrick	University of British Columbia, Canada
Rolf Klein	University of Bonn, Germany
Kim Skak Larsen	University of Southern Denmark
Der-Tsai Lee	Academia Sinica (Taipei)
Silvano Martello	University of Bologna, Italy
Helen Meng	The Chinese University of Hong Kong, China
See-Kiong Ng	National University of Singapore
Mike Paterson	University of Warwick, UK
Franz Rendl	University of Klagenfurt, Austria
Mitsuhisa Sato	University of Tsukuba, Japan
Jiri Sgall	Academy of Sciences of Czech Republic
Martin Skutella	University of Dortmund, Germany
Paolo Toth	University of Bologna, Italy
Denis Trystram	ID-IMAG, France
Emo Welzl	ETH Zurich, Switzerland

Gerhard Woeginger	TU Eindhoven, The Netherlands
Masafumi Yamashita	Kyushu University, Japan
Yinyu Ye	Stanford University, USA
Guochuan Zhang (Co-chair)	Zhejiang University, China

Award Panel

Rainer E. Burkard	Technical University of Graz, Austria
Xiaotie Deng	City University of Hong Kong, China
Ronald Graham	University of California at San Diego, USA
Peter Hammer	Rutgers Center for Operations Research, USA
Kazuo Iwama	Kyoto University, Japan
Silvano Martello	University of Bologna, Italy
Mike Paterson (Chair)	University of Warwick, UK

Organizing Committee

Guangting Chen	Hangzhou Dianzi University, China
Zhiyi Tan	Zhejiang University, China
Enyu Yao	Zhejiang University, China
Deshi Ye	Zhejiang University, China
Guochuan Zhang (Chair)	Zhejiang University, China

Table of Contents

The Tight Bound of First Fit Decreasing Bin-Packing Algorithm Is $FFD(I) \leq 11/9OPT(I) + 6/9$

György Dósa

Department of Mathematics, University of Pannonia, Veszprém, Hungary
dosagy@almos.vein.hu

Abstract. First Fit Decreasing is a classical bin packing algorithm: the items are ordered into their nonincreasing order, and then in this order the next item is always packed into the first bin where it fits. For an instance I let $FFD(I)$ and $OPT(I)$ denote the number of the used bins by algorithm FFD, and an optimal algorithm, respectively. We show in this paper that

$$FFD(I) \leq 11/9OPT(I) + 6/9, \quad (1)$$

and that this bound is tight. The tight bound of the additive constant was an open question for many years.

Keywords: first fit decreasing, tight bound.

1 Introduction

We can define the classical bin packing problem in the next way: There are items with sizes $y_1, y_2, ..., y_n$, which are positive real numbers, we are looking for minimum number of unit capacity bins, so that each item is packed into exactly one of the bins, and the sum of the sizes of the items being packed into a bin can not be more than 1. The problem is NP-hard, and FFD (First Fit Decreasing) is a classical bin packing algorithm: the items are ordered into their nonincreasing order, and then in this order the next item is always placed into the first bin, where it fits. For an instance I let $FFD(I)$ and $OPT(I)$ denote the number of the used bins by algorithm FFD, and an optimal algorithm, respectively.

D.S. Johnson in his doctoral thesis in 1973 showed [5] that $FFD(I) \leq 11/9OPT(I)+4$. Later, B.S. Baker [2], in 1985 gave a slightly simpler proof, and showed that the additive constant is not more than 3. Then, in 1991, Yue Minyi [4] proved that $FFD(I) \leq 11/9OPT(I) + 1$, but his proof has some problems. It is not easy to understand, and leaves many gaps to be verified by the reader. Later, in 1997, Li Rongheng and Yue Minyi [1] again tightened the additive constant to be 7/9, but they do not prove the statement, only give a draft about it. In that paper the authors also conjectured that the tight additive constant is 5/9. What is the least value of the additive constant was an open question for many-many years. Now we show that

$$FFD(I) \leq 11/9OPT(I) + 6/9, \quad (2)$$

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 1–11, 2007.

and prove that this bound is already the tight one, the additive constant can not be further decreased. We follow mainly (but not completely) the proof of [4].

Since the tight upper bound was not known previously, even the next question, which seems to be quite trivial, could not be answered: Is there an instance I for the problem for which $OPT(I) = 5$ and $FFD(I) = 7$ hold? In the recent work [6] we give a short proof that there is not such instance. (From statement (2) follows that there can not be such instance, but if we know only $FFD(I) \leq 11/9OPT(I) + 1$, the question can not be answered.) The next example shows that the additive constant can not be less than 6/9: (this example is a simple modification of that being in [3], Chapter 2, page 16). The example consists of six fully packed optimal bins, an the items are packed by FFD into eight bins. Then the upper bound (2) is proven to be tight, since $11/9 \cdot 6 + 6/9 = 72/9 = 8$. The example is as follows: Let $B_1 = \{1/2 + \varepsilon, 1/4 + \varepsilon, 1/4 - 2\varepsilon\}$, and $B_2 = \{1/4 + 2\varepsilon, 1/4 + 2\varepsilon, 1/4 - 2\varepsilon, 1/4 - 2\varepsilon\}$. If there are 4 copies from B_1 and 2 copies from B_2 in the optimal packing, the number of FFD bins will be 8. We also get the tight bound, if the number of the previous optimal bins are $6k + 4$ and $3k + 2$, respectively.

Can we now give (supposing that (2) really holds) for **all** integer m the biggest possible number n, such that $OPT(I) = m$ and $FFD(I) = n$? We need for this purpose one more example: If there are $6k + 1$ copies from B_1 and $3k + 1$ copies from B_2 in the optimal packing, the number of FFD bins will be $11k + 3$. Then we get the next table for the maximum possible values of n, (we denote the difference of the n and m by d):

OPT(I)=m	1	2	3	4	5	6	7	8	9	10
FFD(I)=n	1	3	4	5	6	8	9	10	11	12
n-m=d		1	1	1	1	2	2	2	2	2
m		11	12	13	14	15	16	17	18	19
n		14	15	16	17	19	20	21	22	23
d		3	3	3	3	4	4	4	4	4
m		20	21	22	23	24	25	26	27	28
n		25	26	27	28	30	31	32	33	34
d		5	5	5	5	6	6	6	6	6

For higher values of m, n, and d the table follows in the same way.

For example, there exists such instance, for which the number of the bins in an optimal packing is 5, and the number of bins used by FFD is 6, but this latter value can not be bigger. Without the tight upper bound (2) and previous examples we could not know the maximum value of n in infinite many cases. Then now, it remained only to see the proof for the upper bound. Since the complete detailed proof requires more than 30 pages and we have a page limit, some details can not be treated here, but the author gladly send the manuscript with the whole proof to the interested reader.

2 Preliminaries

In this paper we show the next theorem:

Theorem 1. $9FFD(L) \leq 11OPT(L) + 6$.

Proof: It is trivial, that this statement is equivalent with (2), and since $FFD(I)$ and $OPT(I)$ are integers, it suffices to show that there is not such instance for which

$$9FFD(I) \geq 11OPT(I) + 7 \tag{3}$$

holds. Suppose to the contrary that I is a minimal counterexample, i.e. contains minimum number of items, and (3) holds. It is trivial that then $OPT(I) \geq 2$ and $FFD(I) \geq 3$ must hold.

Let us denote the optimal bins as B_i^* for $i = 1, ..., OPT(I)$, and the FFD bins as B_i for $i = 1, ..., FFD(I)$. The sum of the sizes of items being packed into a bin will be denoted as $Y(B_i)$, and $Y(B_i^*)$, respectively. From the minimality of the counterexample follows that the last FFD bin contains only one, i.e. the last item. Let this item be denoted as X. (The size of this item also will be denoted simply as X.) Let the size of the items be y_k, for $k = 1, ..., n$, we also denote the items as y_k. We suppose w.l.o.g. that the size of the items are nonincreasing, i.e. $y_1 \geq y_2 \geq ... \geq y_n = X$.

We also use the next denotations: Let the j-th item of the i-th optimal bin be denoted as $A_{i,j}^*$ for every $i = 1, ..., OPT(I)$, and let the j-th item of the i-th FFD bin be denoted as $A_{i,j}$ for every $i = 1, ..., FFD(I)$. (We use different denotations for the same items). We assume without loss of the generality that for every i and every $j_1 < j_2$ holds that $A_{i,j_1}^* \geq A_{i,j_2}^*$, and A_{i,j_1}^* comes before A_{i,j_2}^* in the nonincreasing order of the items, and similarly, follows from the FFD rule that for every i and every $j_1 < j_2$ holds that $A_{i,j_1} \geq A_{i,j_2}$, and A_{i,j_1} comes before A_{i,j_2} in the nonincreasing order of the items. A bin is called as i-bin, if it contains exactly i items.

Because all items fit in the optimal packing into $OPT(I)$ optimal bins, follows that $\sum_{k=1}^{n} y_k \leq OPT(I)$. Note that item X does not fit into any previous FFD bin, thus we get

$$Y(B_i) + X > 1, i = 1, ..., FFD(I) - 1. \tag{4}$$

Lemma 1. $X > \frac{FFD(I)-OPT(I)-1}{FFD(I)-2} \geq 2/11$.

Proof. The second inequality is equivalent by (3). From (4) follows that $Y(B_i) > 1-X$ for all $1 \leq i \leq FFD(I)-1$, and $X+Y(B_1) > 1$. Applying these inequalities we get

$$OPT(I) \geq \sum_{k=1}^{OPT(I)} y_k = X + Y(B_1) + \sum_{i=2}^{FFD(I)-1} Y(B_i) > 1 + (1-X)(FFD(I)-2),$$

from which the first inequality follows.

Corollary 1. $X > \frac{\lceil 11/9OPT(I)+7/9\rceil - OPT(I) - 1}{\lceil 11/9OPT(I)+7/9\rceil - 2}$

Proof. We apply (3), the previous lemma, and the facts that $FFD(I)$ is integer, and the ratio $\frac{FFD(I)-OPT(I)-1}{FFD(I)-2}$ is increasing regarding $FFD(I)$.

From now we know that each optimal or FFD bin can contain at most five items.

Definition 1. *We say that bin B_i dominates the optimal bin B_j^* for some i and j, if for every item y_k being in B_j^* there exists an item y_l being in B_i for which $y_k \leq y_l$ and these items in B_i are different.*

Lemma 2. *There are no bins B_i and B_j^* such that B_i dominates B_j^*.*

Proof. Swap one by one the items in B_j^* by items of B_i that dominate them. Then omitting the elements of this bin we get a smaller counterexample, that is a contradiction.

Lemma 3. *Each optimal bin contains at least three items.*

Proof. If an optimal bin contains one element, then by the domination lemma we get a contradiction. Suppose that an optimal bin contains two items, Y and Z, and $Y \geq Z$. Consider the moment when Y is packed. If this item is packed as a first item into an FFD bin, then Z fits into this bin, thus at least one more item, not less than Z will be packed into this bin, which again contradicts to Lemma 2. If Y is not a first item, then the first item is not less than Y, and the second one (i.e. Y) is not less than Z, a contradiction again.

Lemma 4. *Every FFD bin but the last one contains at least two items.*

Proof. Suppose that B_i contains one element, for some $1 \leq i \leq FFD(I) - 1$, let this item be Y. Let this item be packed into the B_j^* optimal bin, then there is an other item in this optimal bin, say Z. Then $Y + Z \leq 1$, follows that $Y + X \leq 1$, thus the last item X fits into this bin, a contradiction.

Lemma 5. $X \leq 1/4$.

Proof. Suppose that $X > 1/4$, then every optimal bin contains at most three items. From Lemma 3 we get that every optimal bin contains exactly three items, thus the number of the items is $3OPT(I)$, and all items are less than $1/2$.

Suppose that there are two consecutive bins B_i and B_j, (then $j = i + 1$), and B_i contains three, and B_j contains two elements. If $A_{i1} + A_{i2} \geq A_{j1} + A_{j2}$, then because A_{i3} fits into the i-th bin, and $A_{i3} \geq X$, follows that X fits into the j-th bin, a contradiction. Thus $A_{i1} + A_{i2} < A_{j1} + A_{j2}$. Because $A_{i1} \geq A_{j1}$, follows that $A_{i2} < A_{j2}$. Thus A_{j2} is packed before A_{i2}, and it did not fit into the i-th bin, thus $A_{i1} + A_{j2} > 1$, thus at least one of them is bigger than a half, a contradiction.

Follows that the first some FFD bins contain two items, the next FFD bins contain three items, and the last FFD bin contains only one item. Let n_i be the number of the FFD i-bins, for $i = 2, 3$. Then $n_2 + n_3 + 1 = FFD(I)$, and the

number of the items is $3FFD(I) - n_2 - 2$. Since the sum of the sizes of any two items is less than 1, the first $2n_2$ items (in the nonincreasing order) are packed pairwise into the first n_2 FFD bins, and follows that

$$y_{2n_2-1} + y_{2n_2} + X > 1. \tag{5}$$

On the other hand, consider the first item in the nonincreasing order, which is a second item in some optimal bin, i.e. consider the largest $A^*_{i,2}$ item. This cannot be later than the $(OPT(I)+1)$-th item, thus $A^*_{i,2} = y_{k_2}$ for some $k_2 \leq OPT(I)+1$. Let $A^*_{i,1} = y_{k_1}$ and $A^*_{i,3} = y_{k_3}$, then $k_1 < k_2 < k_3$, and

$$y_{OPT(I)} + y_{OPT(I)+1} + X \leq y_{k_1} + y_{k_2} + y_{k_3} \leq 1. \tag{6}$$

Comparing (5) and (6) follows that $OPT(I) \geq 2n_2$. We get that the number of items is

$$\begin{aligned} 3OPT(I) = 3FFD(I) - n_2 - 2 &\geq 3FFD(I) - OPT(I)/2 - 2 && (7)\\ &\geq 3\,(11/9OPT(I) + 7/9) - OPT(I)/2 - 2 && (8)\\ &= 19/6OPT(I) + 1/3 > 3OPT(I), && (9) \end{aligned}$$

a contradiction.

At this point we already know that X must lie in interval $(2/11; 1/4]$. In the remaining part of the paper we will divide our investigations into two parts, as $1/5 < X \leq 1/4$, or $2/11 < X \leq 1/5$.

Lemma 6. *For the value of optimum bins holds that $OPT(I) \geq 6$, and in case $X \leq 1/5$ a stronger inequality $OPT(I) \geq 10$ also holds.*

Proof. If $2 \leq OPT(I) \leq 4$, then from Corollary 1 we get that $X > 1/4$, which is contradiction. Reference [6] shows that in case $OPT(I) = 5$ follows that $FFD(I) \leq 6$, which contradicts to (3). Thus follows that $OPT(I) \geq 6$. Similarly we get that if $X \leq 1/5$, then $6 \leq OPT(I) \leq 9$ is not possible, thus $OPT(I) \geq 10$.

We call a bin as open bin if there is at least one item already packed into the bin. A bin is closed if no more item will be packed into this bin. An item which is packed into the last opened bin called as **regular item**, otherwise the item is called as a **fallback item**. Let A be an arbitrary regular item. We say, that B is a **further** item, if it comes (in the nonincreasing order of the items) after A, and will be packed into a later bin. An arbitrary bin is called as (A, B, C) bin, if A, B, and C are items, and exactly these items are packed into that bin. Similar denotations are also used if there are only two, or there are more than three items in a bin. We often will use the next lemma from [4].

Lemma 7. *Let x_i be the last item in the nonincreasing order which is packed into a (L, x_i) FFD-bin, where $L > 1/2$. Then (i), it can be supposed that there is not such item which has size bigger than x_i and not bigger than $1 - L$, (ii) if there is another item x_k with the same size as x_i, then it is packed into a (L', x_k) FFD-bin where L' precedes L and they have equal sizes.*

Proof. Suppose to the contrary that there is an item x_k for which $x_i < x_k \leq 1-L$ holds. Let x_k be the last such item. This is packed by FFD into an earlier bin, which contains exactly one more item $L' \geq L > 1/2$. Then decrease the size of this item to be equal to x_i. Then it will be packed into the same bin, and there will no more item be packed into this bin. This means that we get a new, appropriate counterexample. By induction we get a counterexample which meets property (i). Regarding (ii): If there exists an x_k with the same size as x_i, then it is packed into a (L', x_k) FFD-bin, where $L' \geq L$. Suppose that $L' > L$, and let L' be the last such item. Then we can decrease the size of L' to be equal to L, and we get a new counterexample again.

Lemma 8. *Let x_i be the last item which is packed into a (B_0, x_i) FFD-bin where $\frac{1-X}{2} < B_0 \leq 1/2$, ($x_i$ comes after B_0). Then (i), each previous item with size between $\frac{1-X}{2}$ and $1/2$ has the same size as B_0, (ii) all L items greater than half have the same size, (iii) all $L > 1/2$ is packed into some (L, x_k) FFD-bin where $x_k = x_i$.*

Proof. The proof is similar to the previous one.

3 Case $1/5 < X \leq 1/4$

We put the items into some classes according to their sizes. The classification used here is not the same, only similar to one used in Yue's paper. The classes are **large, big, medium, small, quite small,** and **very small** items, the items being in a class are denoted as L, B, M, S, U, V, respectively. We also add some weights to the items, as follows:

Name	Class	Weight	Or simply
Large	$\frac{1}{2} < L$	$23/36(1-X)$	23
Big	$\frac{1-X}{2} < B \leq \frac{1}{2}$	$18/36(1-X)$	18
Medium	$\frac{1}{3} < M \leq \frac{1-X}{2}$	$15/36(1-X)$	15
Small	$\frac{1-X}{3} < S \leq \frac{1}{3}$	$12/36(1-X)$	12
qUite small	$\frac{1}{4} < U \leq \frac{1-X}{3}$	$9/36(1-X)$	9
Very small	$X \leq V \leq \frac{1}{4}$	$9/36(1-X)$	9

The classification of the items in case $1/5 < X \leq 1/4$.

Note, that the classes are well defined, furthermore $\frac{1-X}{2} < \frac{1+X}{3} < 2X$ holds since $X > 1/5$. Note, that since every optimal bin have at least three items, follows that $L + 2X \leq 1$ holds for any L item, thus in the FFD packing an M item fits into an L-bin, if there is not other item packed yet into the bin. We use the denotation $c_1L + c_2B + c_3M + c_4S + c_5U + c_6V > c_7$, where c_i are integers or 0 for $i = 1, ..., 7$, and the inequality holds substituting the sizes of any large, big, medium, small, quite small and very small items. For example $L + 2U > 1$ holds, since $L > 1/2$ and $U > 1/4$.

We denote the weight of an item Z as $w(Z)$, and the weight of an optimal or FFD bin as $w(B^*)$, or $w(B)$, respectively. We define the **reserve** of an optimal bin as $r(B^*) = 44/36(1-X) - w(B^*)$.. When we define the weights of the classes, we do it in such a way, that no optimal bin will have weight more than $44/36(1-X)$, i.e. the reserve of all optimal bins are nonnegative, and almost all of the optimal bins have positive reserve. (This will not be true in one case, then we will modify the weights of the items.) Define the **surplus** of an FFD bin as $sur(B) = w(B) - (1-X)$, if this value is nonnegative. Otherwise, let $short(B) = (1-X) - w(B)$ be called as **shortage**, (similarly, as in Yue's paper). If the weight of every FFD bin was at least $1-X$, (i.e. in case when there is not shortage, also applying that the reserve of all optimal bin is nonnegative), we could easily get that

$$(1-X)FFD(I) \leq \sum_{k=1}^{FFD(I)} w\left(B_k\right) = w(I) = \sum_{k=1}^{OPT(I)} w\left(B_k^*\right) \leq 11/9(1-X)OPT(I),$$

and our proof would be ready. Unfortunately, such FFD bins that has less weight (i.e. has some shortage) may exist. But we prove that all shortage can be **covered** by the reserve of the optimal bins plus the surplus of the other FFD bins, plus the required additive constant $27/36(1-X)$. In this section the weight of the smallest class will is $w(V) = w(X) = 9/36(1-X)$. Thus, the shortage of the last FFD bin, which contains only the last item X, is just $(1-X) - w(X) = 27/36(1-X)$, thus the additive constant just covers the shortage of the last FFD bin. For the simplicity, we will say that the weight of a V item is 9, (and similarly in case of other items), and the shortage of a bin containing only X is 27, (instead than $9/36(1-X)$, and $27/36(1-X)$, respectively), i.e. we say simple only the numerator of the ratio.

Let $sur(I)$ and $res(I)$ be the total value of the surplus and reserve of all FFD and optimal bins, respectively, let the required additive constant $27/36(1-X)$ be denoted as $rex(I)$, and finally let $short(I)$ be the total value of the shortage given by all FFD bins. Then we have

$$w(I) = \textstyle\sum_{k=1}^{FFD(I)} w\left(B_k\right) = (1-X)FFD(I) + sur(I) - short(I), \tag{10}$$

$$w(I) \quad = \textstyle\sum_{k=1}^{OPT(I)} w\left(B_k^*\right) = 11/9(1-X)OPT(I) - res(I). \tag{11}$$

Suppose that

$$res(I) + sur(I) + rex(I) \geq short(I) \tag{12}$$

holds. Then then applying (10) and (11), we have

$$(1-X)FFD(I) = w(I) - sur(I) + short(I) \tag{13}$$

$$\leq w(I) + res(I) + rex(I) \tag{14}$$

$$= 11/9(1-X)OPT(I) + 27/36(1-X), \tag{15}$$

and dividing by $(1-X)$, and considering that $27/36 < 7/9$ we get our main result. Thus in the remained part of this section we prove (12). First, let us

see what bins are possible, according only to the definition of the classes. First we list all possible optimal bins, then all possible FFD bins. In the last rows r means the reserve of the optimal bins, while s denotes the value of the surplus or shortage of the FFD bins. If s is positive, then it means surplus, if it is negative, for example if it is -3 , it means that the shortage of that FFD bin is three. The value of reserve, surplus of shortage of some bin can be easily computed by the weights of the classes. (In case of (L,U) or (L,V) FFD bins we define the weights of the classes in some other way.)

OPT

L	1	1	1																				
B				1	1	1	1	1	1	1	1	1											
M				1	1	1							2	2	2	1	1	1	1	1	1	1	1
S	1			1			2	1	1				1			2	1	1					
U		1			1			1		2	1			1			1		2	1	1		
V	1	1	2			1			1		1	2			1			1		2	1	3	2
r	0	3	3	-1	2	2	2	5	5	8	8	8	2	5	5	5	8	8	11	2	11	2	11

S	3	2	2	2	2	1	1	1	1	1	1								
U		1	1			2	2	1	1			3	3	2	2	1	1		
V		1		2	1	1		2	1	3	2	1		2	1	3	2	4	3
r	8	2	11	2	11	5	14	5	14	5	14	8	17	8	17	8	17	8	17

FFD

L	1	1	1	1	1			
B	1					2	1	1
M		1					1	
S			1					1
U				1				
V					1			
s	5	2	-1			0	-3	-6

L	1	1	1									
B				1	1	1	1	1	1	1	1	1
M				1	1	1						
S	1			1			2	1	1			
U		1			1			1		2	1	
V	1	1	2			1			1		1	2
s	8	5	5	9	6	6	6	3	3	0	0	0

L											
B											
M	2	2	2	1	1	1	1	1	1	1	1
S	1			2	1	1					
U		1			1		2	1	1		
V			1			1		2	1	3	2
s	6	3	3	3	0	0	-3	6	-3	6	-3

S	3	2	2	2	2	1	1	1	1	1	1					
U		1	1			2	2	1	1			3	2	1		
V		1		2	1	1		2	1	3	2	1	2	3	4	1
s	0	6	-3	6	-3	3	-6	3	-6	3	-6	0	0	0	0	-27

Lemma 9. *There can not be other optimal, nor FFD bins.*

Proof. Each optimal bin contains three or four items, and each FFD bin can contain at least two, and at most four items. Furthermore each bin has items with total size at most one, and each FFD has items with size more than $1 - X$. Using the next inequalities: $L+2U > 1$, $2B+V > 1$, $B+3V > 1$, $M+S+2V > 1$, $3S + V > 1$, $B + U + X \leq 1$, $3U + X \leq 1$, (all of them can be shown easily), we get that exactly the above bins are possible.

We emphasize, that all types of the above FFD bins can not occur at the same time! For example if there exists a (B,2S) FFD bin, then there can not be other FFD bin which contains a big item and no large item, since then these two B items would fit into one bin.

We use the next notation: If L is packed into (L,A) FFD bin, and (L,B,C) optimal bin, where A, B, and C are items and L is a large item, we say that L is packed into the $\{(L,A),(L,B,C)\}$ **bbin** (to denote, that this is not really a bin, but a bin-pair). The first and second part of a bbin are the FFD and the optimal bins of item L, respectively. An arbitrary bin is denoted as (A,.) bin, where A is a class, and the bin contains at least one item from class A, but does not contain items from higher classes. For example, (M,.) denotes a bin, which contain at least one M item, but does not contain L or B items.

Theorem 2. *Suppose that there are not (L,U), (L,V) FFD bins, and there is not* $\{(L,S),(L,S,V)\}$ *bbin. Then statement (12) holds.*

Proof. First suppose that there is not (B,M,S) optimal bin. Since there is not $\{(L,S),(L,S,V)\}$ bbin, follows that by every L item we get 2 reserve at least, and the shortage caused by the (L,S) FFD bins are all covered. Also, since every optimal not L-bin has at least 2 reserve, we have totally at least 12 reserve, since there must be at least six optimal bins by Lemma 6. On the other hand, let us count the possible shortage caused by not (L,S) bins. If there is (B,S) FFD bin, then from Lemma 8 we know that there is not M item. Then it can not be (B,M) FFD bin, and we have at most 12 shortage, since it can not be at the same time two (S,.) FFD bins with shortage, and all shortage is covered. In the opposite case, if there is not (B,S) FFD bin, then the total not covered shortage is at most 3 (by a possible existing (B,M) FFD bin) plus 3 (by an (M,.) FFD bin) plus 6 by an (S,.) FFD bin, this is altogether again 12, and it is again covered.

Now suppose that there is at least one (B,M,S) optimal bin. Then,.follows that $\frac{1-X}{2}+\frac{1}{3}+S<B+M+S\leq 1$ holds for the previous B, M, and S items, from what we get that $S<\frac{1}{6}+\frac{X}{2}$ holds for the smallest S item. Let this smallest S be denoted as S_0. Again, since all S is greater than $\frac{1-X}{3}$, and all M is greater than $\frac{1}{3}$, follows that there is such B item which is less than $\frac{1+X}{3}$. Then we conclude the next things: (i), It can not be $\{(L,M),(L,S,V)\}$ bbin, since then, from Lemma 7 follows that all B items must be greater than the sum of these two S and V items being in the (L,S,V) bin, thus $B>\frac{1-X}{3}+X$ holds for all B items, contradiction. (ii), $M+S+S_0\leq\frac{1-X}{2}+\frac{1}{3}+\frac{1}{6}+\frac{X}{2}=1$ holds. From this second property follows, that if there is (M,S,U) or (M,S,V) FFD bin, then there is not further S item.

Now we are ready to finish the investigation of this case. We redefine the weight of exactly those M items what are packed into (B,M,S) optimal bins as 12, let these M items be called as small M items. Then, as before we have at least 12 reserve, since by all L items we get 2 reserve, and by all other optimal bins we have 2 more reserve, and all shortage caused by the (L,S) or (L,M) FFD bins are covered.

How many shortage can cause the not L FFD bins? Let us realize, that both M items in an (2M,U) or (2M,V) FFD bin can not be small M items, since a small M item, plus an appropriate B item and the smallest S item fit into one bin. Follows that by (2M,S), (2M,U) or (2M,V) FFD bins we get no shortage. By a (B,M) or (B,S) FFD bin (they can not exist at the same time), by an (M,.) FFD bin, and finally by an (S,.) FFD bin we can get at most $6+6+6=18$ shortage. If there is not (B,M) nor (B,S) bin, then all shortage is covered. If there is not (M,.) FFD bin with shortage, again, all shortage is covered, and this is also true for the (S,.) FFD bins. Finally we can consider the next things:

Suppose that there is an (M,.) FFD bin which has no S item, then there is not further S item, thus it can not be (S,.) bin with shortage, and all shortage is covered. Also, we have seen that if there is (M,S,U) or (M,S,V) FFD bin, then again, there is not further S item. Thus, if there is shortage caused by some (M,.) bin, then the total not covered shortage is at most $6+6+0=12$, otherwise, if there is, again it is at most $6+0+6=12$, and the statement is proved.

Theorem 3. *If there is (L,V) or (L,U) FFD bin, or there is* $\{(L,S),(L,S,V)\}$ *bbin, then statement (12) also holds.*

Proof. The proof can be made by case analysis. The details can not fit here, but the author gladly send it to the interested reader.

Thus we proved that in case $1/5 < X \leq 1/4$ our statement holds. It only remained the following case.

4 Case $2/11 < X \leq 1/5$

In this case we redefine the classes of the items and their weights, as follows:

Class	Weight	Or simply
$\frac{1}{2} < L$	$\frac{24}{36}(1-X)$	24
$\frac{1-X}{2} < B1 \leq \frac{1}{2}$	$\frac{18}{36}(1-X)$	18
$\frac{3}{8}-\frac{X}{8} < B2 \leq \frac{1-X}{2}$	$\frac{16}{36}(1-X)$	16
$\frac{1}{3} < M1 \leq \frac{3}{8}-\frac{X}{8}$	$\frac{15}{36}(1-X)$	15
$\frac{1+X}{4} < M2 \leq \frac{1}{3}$	$\frac{12}{36}(1-X)$	12
$\frac{1}{4} < S \leq \frac{1+X}{4}$	$\frac{10}{36}(1-X)$	10
$\frac{1-X}{4} < U \leq \frac{1}{4}$	$\frac{9}{36}(1-X)$	9
$X \leq V \leq \frac{1-X}{4}$	$\frac{8}{36}(1-X)$	8

The classification of the items in case $2/11 < X \leq 1/5$

Then the investigations are similar to that being in the previous section, but there are more than 200 optimal, and also more than 200 possible FFD bins.

Acknowledgement. The author would like to thank Guochuan Zhang, Leah Epstein and David S. Johnson for their valuable comments and help.

References

1. Li, R., Yue, M.: The proof of $FFD(L) \leq 11/9OPT(L) + 7/9$. Chinese Science Bulletin 42(15) (August 1997)
2. Baker, B.S.: A new proof for the first-fit decreasing bin-packing algorithm. J. Algorithms, 49–70 (1985)
3. Coffmann, E.G., Garey Jr., M.R., Johnson, D.S.: Approximation algorithms for bin packing: A survey. In: Hochbaum, D. (ed.) Approximation algorithms for NP-hard problems. PWS Publishing, Boston (1997)
4. Yue, M.: A simple proof of the inequality $FFD(L) \leq 11/9OPT(L) + 1$, $\forall L$, for the FFD bin-packing algorithm. Acta Mathematicae Applicatae Sinica 7(4), 321–331 (1991)
5. Johnson, D.S.: Near-optimal bin-packing algorithms. Doctoral Thesis. MIT Press, Cambridge (1973)
6. Zhong, W., Dósa, Gy., Tan, Z.: On the machine scheduling problem with job delivery coordination. European Journal of Operations Research, online (2006)

Sequential Vector Packing*

Mark Cieliebak[2], Alexander Hall[1], Riko Jacob[1], and Marc Nunkesser[1]

[1] Department of Computer Science, ETH Zurich, Switzerland
{alex.hall,riko.jacob,marc.nunkesser}@inf.ethz.ch
[2] sd&m Schweiz AG, 8050 Zurich, Switzerland
mark.cieliebak@sdm.com

Abstract. We introduce a novel variant of the well known d-dimensional bin (or vector) packing problem. Given a sequence of non-negative d-dimensional vectors, the goal is to pack these into as few bins as possible of smallest possible size. In the classical problem the bin size vector is given and the sequence can be partitioned arbitrarily. We study a variation where the vectors have to be packed in the order in which they arrive and the bin size vector can be chosen once in the beginning. This setting gives rise to two combinatorial problems: One in which we want to minimize the number of used bins for a given total bin size and one in which we want to minimize the total bin size for a given number of bins. We prove that both problems are $\mathcal{NP}$-hard and propose an LP based bicriteria $(\frac{1}{\epsilon}, \frac{1}{1-\epsilon})$-approximation algorithm. We give a 2-approximation algorithm for the version with bounded number of bins. Furthermore, we investigate properties of natural greedy algorithms, and present an easy to implement heuristic, which is fast and performs well in practice.

Suppose you want to spray a long text on a wall using stencils for the letters and spray color. You start from the left and assemble as much of the beginning of the text as you have matching stencils at your disposal. Then you mask the area around the stencils and start spraying. Afterwards, you remove the stencils again, so that you can reuse them in the next steps. You iterate this procedure starting after the last letter that was sprayed until the whole text is finished. The sequential unit vector packing problem can be formulated as the following question: If you have bought enough material to produce at most B stencils before you start, how many stencils b_i of each letter $i \in \{\mathrm{A} \ldots \mathrm{Z}\}$ do you produce in order to minimize the number of steps that you need to spray the whole text?

The problem can be seen as an inverse vector packing problem: The sequence in which the items (characters, interpreted here as unit vectors) occur is fixed and cannot be altered. On the other hand, the bin size vector (here $(b_\mathrm{A}, \ldots, b_\mathrm{Z})$) can be changed as long as its component-wise sum does not exceed a given value B. An equivalent problem was posed to us by an industry partner from the manufacturing industry, where *exactly* this question arises in a production

* Work partially supported by European Commission - Fet Open project DELIS IST-001907 Dynamically Evolving Large Scale Information Systems, for which funding in Switzerland is provided by SBF grant 03.0378-1.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 12–23, 2007.

process. As a small letdown we are not allowed to give the details of this process. In this paper we study both this problem and the slightly generalized version where the vectors in the sequence are not restricted to unit vectors. Formally, the sequential vector packing problem is defined as follows:

Definition 1 (Sequential vector packing, SVP)
Given: *a sequence* $\mathbf{S} = \mathbf{s}_1 \cdots \mathbf{s}_n$ *of* demand vectors $\mathbf{s}_i = (s_{i1}, \ldots, s_{id}) \in \mathbb{Q}_+^d$, $d \in \mathbb{N}$.

Evaluation: *for a* bin vector *(or short:* bin*)* $\mathbf{b} = (b_1, \ldots, b_d) \in \mathbb{Q}_+^d$ *of* total bin size $B = |\mathbf{b}|_1 = \sum_{j=1}^{d} b_j$, *the sequence* $\mathbf{s}_1 \cdots \mathbf{s}_n$ *can be packed in this order into* k *bins, if* breakpoints $0 = \pi_0 < \pi_1 < \cdots < \pi_k = n$ *exist, such that*

$$\sum_{i=\pi_l+1}^{\pi_{l+1}} \mathbf{s}_i \leq \mathbf{b} \qquad \text{for } l \in \{0, \ldots, k-1\} \ ,$$

where the inequalities over vectors are component-wise.
We denote the minimum number of bins for given $\mathbf{S}$ *and* $\mathbf{b}$ *by* $\kappa(\mathbf{b}, \mathbf{S}) = k$. *This number can be computed in linear time. We denote the* j*th component,* $j \in \{1, \ldots, d\}$, *of the demand vectors and the bin vector as* resource j, *i.e.,* s_{ij} *is the demand for resource* j *of the* i*th demand vector. We also refer to* $\mathbf{s}_i$ *as* position i.

Goals: *minimize the total bin size and the number of bins. We formulate this bicriteria objective in the following two versions:*

Bounded Size SVP *for a given bin size* B *find a bin vector* $\mathbf{b}$ *with* $B = |\mathbf{b}|_1$, *such that* $\kappa(\mathbf{b}, \mathbf{S})$ *is minimum.*
Bounded Number SVP *for a given number of bins* k *find a bin vector* $\mathbf{b}$ *with* $\kappa(\mathbf{b}, \mathbf{S}) = k$, *such that the total bin size* $B = |\mathbf{b}|_1$ *is minimum.*

The *sequential unit vector packing (SUVP)* problem considers the restricted variant where $\mathbf{s}_i$, $i \in \{1, \ldots, n\}$, contains exactly one entry equal to 1, all others are zero. Note that any solution for this version can be transformed in such a way that the bin vector is integral, i.e., $\mathbf{b} \in \mathbb{N}^d$, by potentially rounding down resource amounts to the closest integer (therefore one may also restrict the total bin size to $B \in \mathbb{N}$). The same holds if all vectors in the sequence are integral, i.e., $\mathbf{s}_i \in \mathbb{N}^d$, $i \in \{1, \ldots, n\}$.

Given the bicriteria objective function it is natural to consider bicriteria approximation algorithms: We call an algorithm a *bicriteria* (α, β)*-approximation algorithm* for the sequential vector packing problem if it finds for each sequence $\mathbf{S}$ and bin size $\beta \cdot B$ a solution which needs no more than α times the number of bins of an optimal solution for $\mathbf{S}$ and bin size B.

Related Work. There is an enormous wealth of publications both on the classical bin packing problem and on variants of it. The two surveys by Coffman, Garey and Johnson [2, 8] give many pointers to the relevant literature until 1997. In [3]

Coppersmith and Raghavan introduce the multidimensional (on-line) bin packing problem. There are also some variants that take into consideration precedence relations on the items [13, 12] that remotely resemble our setting. Galambos and Woeginger [6] give a comprehensive overview over the on-line bin-packing and vector-packing literature. We like to stress though that our problem is not an on-line problem, since we are given the complete (albeit immutable) sequence in the beginning. We are unaware of any publication that deals with the sequential vector packing problem. In the context of scheduling algorithms, allowing a certain relaxation in bicriteria approximations (here increasing the bin size) is also called resource augmentation, cf. [9, 10].

New Contributions and Outline. In Section 1 we present approximation algorithms for the sequential vector packing problem. These are motivated by the strong $\mathcal{NP}$-hardness results that we give in Section 2. The approximation algorithms are based on an LP relaxation and two different rounding schemes, yielding a bicriteria $(\frac{1}{\varepsilon}, \frac{1}{1-\varepsilon})$-approximation and—as our main result—a 2-approximation for the bounded number version of the problem. Recall that the former algorithm, e.g., for $\varepsilon = \frac{1}{3}$, yields solutions with at most 3 times the optimal number of bins while using at most 1.5 times the given total bin size B, the latter may use at most the optimal number of bins and at most twice the given total bin size B. In Section 3 we present two simple greedy strategies and argue why they perform badly in the worst case. Furthermore, we give an easy to implement randomized heuristic and present two optimizations concerning subroutines. In particular, we show how to compute $\kappa(\mathbf{b}, \mathbf{S})$ in time $O(\kappa(\mathbf{b}, \mathbf{S}) \cdot d)$ after a preprocessing phase which takes $O(n)$ time.Due to space limitations, we omit some of the proofs in this extended abstract. These proofs can be found in the technical report [1], in which we also experimentally evaluate the presented algorithms with promising results on real world data, in particular for the randomized heuristic.

1 Approximation Algorithms

We present an ILP formulation which we subsequently relax to an LP. We continue by describing a simple rounding scheme which yields a bicriteria $(\frac{1}{\varepsilon}, \frac{1}{1-\varepsilon})$-approximation for bounded size and bounded number SVP and then show how to obtain a 2-approximation for bounded number SVP.

1.1 ILP Formulation

For a given sequence $\mathbf{S}$, let $\mathbf{w}_{u,v} := \sum_{i=u+1}^{v} \mathbf{s}_i$, for $u, v \in \{0, \ldots, n\}$ and $u < v$, denote the *total demand* (or *total demand vector*) of the subsequence $\mathbf{S}_{u,v} := \mathbf{s}_{u+1} \cdots \mathbf{s}_v$. If $\mathbf{w}_{u,v} \leq \mathbf{b}$ holds, we can pack the subsequence $\mathbf{S}_{u,v}$ into bin $\mathbf{b}$. The following integer linear programming (ILP) formulation solves both versions of the sequential vector packing problem. Let $X := \{x_i | i \in \{0, \ldots, n\}\}$ and $Y := \{y_{u,v} | u, v \in \{0, \ldots, n\}, u < v\}$ be two sets of 0-1 variables.

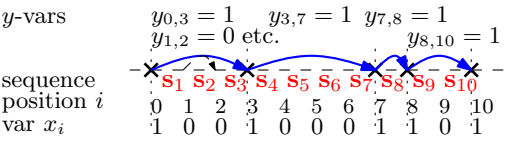


Fig. 1. An exemplary instance, its potential ILP solution, and its flow representation. The solution needs 4 bins.

$$\begin{aligned}
&\min\ k \text{ (or } B) \\
&\text{s.t. } x_0 = 1 && (1)\\
&\sum_{u=0}^{i-1} y_{u,i} = x_i \text{ for } i \in \{1,\dots,n\} && (2)\\
&\sum_{v=i+1}^{n} y_{i,v} = x_i \text{ for } i \in \{0,\dots,n-1\} && (3)\\
&\sum_{\substack{u,v:\\ u<i\le v}} \mathbf{w}_{u,v} \cdot y_{u,v} \le \mathbf{b} \text{ for } i \in \{1,\dots,n\} && (4)\\
&\sum_{j=1}^{d} b_j = B, \qquad \sum_{i=1}^{n} x_i = k && (5)\\
&B \in \mathbb{Q}_+, (k \in \mathbb{N}), \mathbf{b} \in \mathbb{Q}_+^d, x_i, y_{u,v} \in \{0,1\} \text{ for } x_i \in X, y_{u,v} \in Y
\end{aligned}$$

The 0-1 variable x_i indicates whether there is a breakpoint at position $i \geq 1$. The 0-1 variable $y_{u,v}$ can be seen as a flow which is routed on an edge from position $u \in \{0,\dots,n-1\}$ to position $v \in \{1,\dots,n\}$, with $u < v$, see Figure 1.

The Constraints (2),(3) ensure that flow conservation holds for the flow represented by the $y_{u,v}$ variables and that x_i is equal to the inflow (outflow) which enters (leaves) position i. Constraint (1) enforces that only one unit of flow is sent via the Y variables. The path which is taken by this unit of flow directly corresponds to a series of breakpoints.

In Constraints (4) the bin vector $\mathbf{b}$ comes into play: for any two consecutive breakpoints (e.g., $x_u = x_v = 1$) the constraint ensures that the bin vector is large enough for the total demand between the breakpoints (e.g., the total demand $\mathbf{w}_{u,v}$ of the subsequence $\mathbf{S}_{u,v}$). Note that Constraints (4) sum over all edges that span over a position i (in a sense the cut defined by position i), enforcing that the total resource usage is bounded by $\mathbf{b}$. For the two consecutive breakpoints x_u and x_v this amounts to $\mathbf{w}_{u,v} \cdot y_{u,v} \leq \mathbf{b}$. Finally, Constraints (5) ensure the correct total size of the bin vector and the correct number of bins.

1.2 An Easy $(\frac{1}{\varepsilon}, \frac{1}{1-\varepsilon})$-Approximation

As a first step, we relax the ILP formulation to an LP: here this means to have $x_i, y_{u,v} \in [0,1]$ for $x_i \in X, y_{u,v} \in Y$. The following Algorithm Eps Rounding computes a $(\frac{1}{\varepsilon}, \frac{1}{1-\varepsilon})$-approximation:

1. Solve the LP optimally with one of the two objective functions.
   Let $(X^\star, Y^\star, \mathbf{b}^\star)$ be the obtained fractional solution.

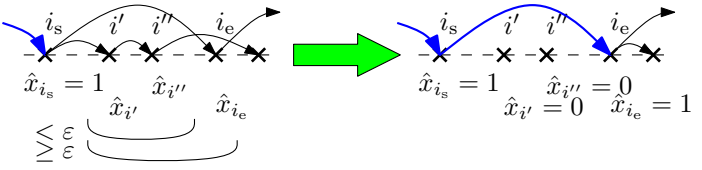


Fig. 2. An example of the rerouting of flow in Lines 4 (a)-(c) of the algorithm

2. Set $(\hat{X}, \hat{Y}, \hat{\mathbf{b}}) = (X^\star, Y^\star, \frac{1}{1-\varepsilon} \cdot \mathbf{b}^\star)$ and $i_s = 0$. Stepwise round $(\hat{X}, \hat{Y})$:
3. Let $i_e > i_s$ be the first position for which $\sum_{i=i_s+1}^{i_e} \hat{x}_i \geq \varepsilon$.
4. Set $\hat{x}_i = 0$ for $i \in \{i_s+1, \ldots, i_e-1\}$, set $\hat{x}_{i_e} = 1$. Reroute the flow accordingly (see also Figure 2): (a) Set $\hat{y}_{i_s,i_e} = 1$. (b) Increase $\hat{y}_{i_e,i}$ by $\sum_{i'=i_s}^{i_e-1} \hat{y}_{i',i}$, for $i > i_e$. (c) Set $\hat{y}_{i_s,i'} = 0$ and $\hat{y}_{i',i} = 0$, for $i' \in \{i_s+1, \ldots, i_e-1\}$, $i > i'$.
5. Set the new i_s to i_e and continue in Line 3, until $i_s = n$.

Theorem 1. *The algorithm* Eps Rounding *is a* $(\frac{1}{\varepsilon}, \frac{1}{1-\varepsilon})$*-approximation algorithm for the sequential vector packing problem.*

Note that one would not actually implement the algorithm Eps Rounding. Instead it suffices to compute the bin vector $\mathbf{b}^\star$ with the LP and then multiply it by $\frac{1}{1-\varepsilon}$ and evaluate the obtained bin vector, see Section 3.

1.3 A 2-Approximation for Bounded Number Sequential Vector Packing

We prove some properties of the LP relaxation and then describe how they can be applied to obtain the rounding scheme yielding the desired bicriteria ratio.

Properties of the Relaxation. Let $(X, Y, \mathbf{b})$ be a fractional LP solution w.r.t. one of the objective functions; recall that the Y variables represent a flow. Let $e_1 = (u, v)$ and $e_2 = (u', v')$ denote two flow carrying edges, i.e., $y_{u,v} > 0$ and $y_{u',v'} > 0$. We say that e_1 *is contained in* e_2 if $u' < u$ and $v' > v$, we also call (e_1, e_2) an *embracing pair*. We say an embracing pair (e_1, e_2) is *smaller* than an embracing pair $(\hat{e}_1, \hat{e}_2)$, if the length of e_1 (for $e_1 = (u, v)$, its length is $v - u$) is less than the length of $\hat{e}_1$ and in case of equal lengths, if $u < \hat{u}$ (order by left endpoint of $e_1, \hat{e}_1$). That is, for two embracing pairs with distinct e_1 and $\hat{e}_1$ we always have that one is smaller than the other. We show that the following structural property holds:

Lemma 1 (no embracing pairs). *Any optimal fractional LP solution* $(X^\star, Y^\star, \mathbf{b}^\star)$ *can be modified in such a way that it contains no embracing pairs, without increasing the number of bins and without modifying the bin vector.*

Proof. We set $Y = Y^\star$ and show how to stepwise treat embracing pairs contained in Y, proving after each step that $(X^\star, Y, \mathbf{b}^\star)$ is still a feasible LP solution. We furthermore show that this procedure terminates and in the end no embracing pairs are left in Y.

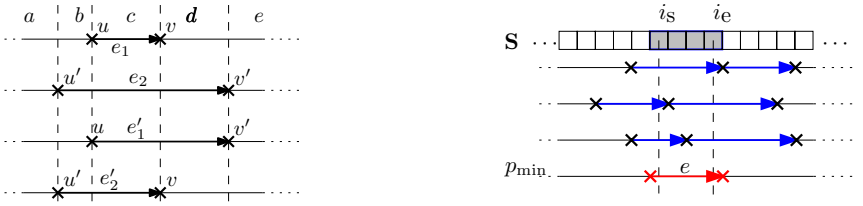


Fig. 3. Left: Replacement of λ units of flow on e_1 and e_2 by λ units of flow on e'_1 and e'_2 in Lemma 1. Right: Extracting the integral solution. Edge e together with other potential edges in $Y^\star$ in Theorem 2.

Let us begin by describing one iteration step, assuming $(X^\star, Y, \mathbf{b}^\star)$ to be a feasible LP solution which still contains embracing pairs. Let (e_1, e_2), with $e_1 = (u, v)$ and $e_2 = (u', v')$, be an embracing pair. We now modify the flow Y to obtain a new flow Y' by rerouting $\lambda = \min\{y_{u,v}, y_{u',v'}\}$ units of flow from e_1, e_2 onto the edges $e'_1 = (u, v')$ and $e'_2 = (u', v)$: $y'_{u,v} = y_{u,v} - \lambda$, $y'_{u',v'} = y_{u',v'} - \lambda$ and $y'_{u',v} = y_{u',v} + \lambda$, $y'_{u,v'} = y_{u,v'} + \lambda$; see also Figure 3 (left). The remaining flow values in Y' are taken directly from Y. It is easy to see that the flow conservation constraints (2),(3) still hold for the values $X^\star, Y'$ (consider a circular flow of λ units sent in the residual network of Y on the cycle u', v, u, v', u'). Since $X^\star$ is unchanged this also implies that the number of bins did not change, as desired. It remains to prove that the Constraints (4) still hold for the values $Y', \mathbf{b}^\star$ and to detail how to consecutively choose embracing pairs (e_1, e_2) in such a way that the iteration terminates.

Feasibility of the Modified Solution. Constraints (4) are parameterized over $i \in \{1, \ldots, n\}$. We argue that they are not violated separately for $i \in \{u'+1, \ldots, u\}$, $i \in \{u+1, \ldots, v\}$, and $i \in \{v+1, \ldots, v'\}$, i.e., the regions b, c, and d in Figure 3 (left). For the remaining regions a and e it is easy to check that the values of the affected variables do not change when replacing Y by Y'. So let us consider the three regions:

Region b (d). The only variables in (4) which change when replacing Y by Y' for this region are: $y'_{u',v'} = y_{u',v'} - \lambda$ and $y'_{u',v} = y_{u',v} + \lambda$. This means that flow is moved to a shorter edge, which can only increase the slack of the constraints: With $\mathbf{w}_{u',v} < \mathbf{w}_{u',v'}$ it is easy to see that (4) still holds in region b. Region d is analogous to b.

Region c. Here the only variables which change in (4) are: $y'_{u,v} = y_{u,v} - \lambda$, $y'_{u',v'} = y_{u',v'} - \lambda$, $y'_{u',v} = y_{u',v} + \lambda$, and $y'_{u,v'} = y_{u,v'} + \lambda$. In other words, λ units of flow were moved from e_1 to e'_1 and from e_2 to e'_2. Let us consider the fraction of demand which is contributed to (4) by these units of flow before and after the modification. Before (on e_1 and e_2) this was $\lambda \cdot (\mathbf{w}_{u,v} + \mathbf{w}_{u',v'})$ and afterwards (on e'_1 and e'_2) it is $\lambda \cdot (\mathbf{w}_{u',v} + \mathbf{w}_{u,v'})$. Since both quantities are equal, the left hand side of (4) remains unchanged in region c.

Choice of (e_1, e_2) and Termination of the Iteration. In each step of the iteration we always choose the smallest embracing pair (e_1, e_2), as defined above. If there

are several smallest embracing pairs (which by definition all contain the same edge e_1), we choose one of these arbitrarily.

First we show that the modification does not introduce an embracing pair which is smaller than (e_1, e_2). We assume the contrary and say w.l.o.g. that the flow added to edge e_1' creates a new embracing pair (e, e_1') which is smaller than the (removed) embracing pair (e_1, e_2). Clearly, e is also contained in e_2. Therefore, before the modification (e, e_2) would have been an embracing pair as well. Since (e, e_2) is smaller than (e_1, e_2) it would have been chosen instead, which gives the contradiction.

It follows that we can divide the iterations into a bounded number of phases: in each phase all considered embracing pairs are with respect to the same e_1-type edge. As soon as a phase is finished (i.e., no embracing pairs with the phase's e_1-type edge remain) this e_1-type edge will never be considered again, since this could only happen by introducing a smaller embracing pair later in the iteration. Consider a single phase during which an edge e_1 is contained in possibly several other edges e_2. By the construction of the modification for an embracing pair (e_1, e_2) it is clear that e_2 could not be chosen twice in the same phase. Therefore, the number of modification steps per phase can also be bounded by $O(n^2)$. □

Choose a Flow Carrying Path. We will use the structural insights from above to prove that bin vector $2 \cdot \mathbf{b}^\star$ yields a 2-approximation for bounded number SVP.

Due to Lemma 1 an optimal fractional LP solution $(X^\star, Y^\star, \mathbf{b}^\star)$ without embracing pairs exists. Let $p_{\min}$ denote the shortest flow carrying path in $(X^\star, Y^\star, \mathbf{b}^\star)$, where shortest is meant with respect to the number of breakpoints. Clearly, the length of $p_{\min}$ is at most the number of bins $\sum_{i=1}^{n} x_i^\star$, since the latter can be seen as a convex combination of the path lengths of an arbitrary path decomposition. Below we show that the integral solution corresponding to $p_{\min}$ is feasible for the bin vector $2 \cdot \mathbf{b}^\star$, and thus $p_{\min}$ and $2 \cdot \mathbf{b}^\star$ give a 2-approximation. Observe that the approximation algorithm does not actually need to transform an optimal LP solution, given, e.g., by an LP solver, into a solution without embracing pairs. The existence of path $p_{\min}$ in such a transformed solution is merely taken as a proof that the bin vector $2 \cdot \mathbf{b}^\star$ yields less than $\sum_{i=1}^{n} x_i^\star$ breakpoints. To obtain such a path, we simply evaluate $2 \cdot \mathbf{b}^\star$ with the algorithm presented in Section 3 ($\mathbf{b}^\star$ given by the LP solver).

Theorem 2. *Given an optimal fractional LP solution $(X^\star, Y^\star, \mathbf{b}^\star)$ without embracing pairs, let $p_{\min}$ denote the shortest flow carrying path. The integral solution corresponding to $p_{\min}$ is feasible for $2 \cdot \mathbf{b}^\star$.*

Proof. We only have to argue for the feasibility of the solution w.r.t. the doubled bin vector. Again we will consider Constraints (4). Figure 3 (right) depicts an edge e on path $p_{\min}$ and other flow carrying edges. We look at the start and end position i_s and i_e in the subsequence defined by e. Denote by $E_{i_s} = \{(u, v) | 0 \leq u < i_s \leq v \leq n\}$ (and E_{i_e}, respectively) the set of all flow carrying edges that cross i_s (i_e) and by $i_{\min}, (i_{\max})$ the earliest tail (latest head) of an arc in $E_{i_s}, (E_{i_e})$. Furthermore, let $E' = E_{i_s} \cup E_{i_e}$. Summing up the two Constraints (4) for i_s and i_e gives $2\mathbf{b}^\star \geq \sum_{(u,v) \in E_{i_s}} y_{u,v}^\star \cdot \mathbf{w}_{u,v} + \sum_{(u,v) \in E_{i_e}} y_{u,v}^\star \cdot \mathbf{w}_{u,v} =: A$ and

thus $2\mathbf{b}^\star \geq A \geq \sum_{i_{\min}<i\leq i_{\max}} \sum_{\substack{(u,v)\in E':\\ u<i\leq v}} y^\star_{u,v} \cdot \mathbf{s}_i \geq \sum_{i_s<i\leq i_e} \sum_{\substack{(u,v)\in E':\\ u<i\leq v}} y^\star_{u,v} \cdot \mathbf{s}_i = \sum_{i_s<i\leq i_e} \mathbf{s}_i = \mathbf{w}_{i_s,i_e}$. The second inequality is in general an inequality because the sets E_{i_s} and E_{i_e} need not be disjoint. For the third inequality we rely on the fact that there are no embracing pairs. For this reason, each position between i_s and i_e is covered by an edge that covers either i_s or i_e. We have shown that the demand between any two breakpoints on $p_{\min}$ can be satisfied by the bin vector $2 \cdot \mathbf{b}^\star$. □

Observe that for integral resources the above proof implies that even $\lfloor 2\mathbf{b}^\star \rfloor$ has no more breakpoints than the optimal solution. Note also that it is easy to adapt both approximation algorithms so that they can handle pre-specified breakpoints. The corresponding x_i values can simply be set to one in the ILP and LP formulations.

2 NP-Completeness

For all considered problem variants it is easy to determine the objective value once a bin vector is chosen. Hence, for all variants of the sequential vector packing problem considered in this article, the corresponding decision problem is in $\mathcal{NP}$. Moreover, the decision problem of both the bounded size and bounded number versions are identical. Therefore, we will not distinguish between the two versions here. We now come to the $\mathcal{NP}$-hardness result. To simplify the exposition, we first consider a variant of the sequential unit vector packing problem where the sequence of vectors has prespecified breakpoints, always after w positions. Then the sequence effectively decomposes into a set of *windows* of length w, and for each position in such a window i it is sufficient to specify the resource that is used at position $j \in \{1, \ldots, w\}$, denoted as $s^i_j \in \{1, \ldots, d\}$. This situation can be understood as a set of sequential unit vector packing problems that have to be solved with the same bin vector. The objective is to minimize the total number of (additional) breakpoints, i.e., the sum of the objective functions of the individual problems. Then, we also show strong $\mathcal{NP}$-hardness for the original problem.

Lemma 2. *Finding the optimal solution for sequential unit vector packing with windows of length 4 (dimension d and bin size B as part of the input) is $\mathcal{NP}$-hard.*

Proof. By reduction from the $\mathcal{NP}$-complete problem Clique [7] or more generally from k-densest subgraph [5]. Let $G = (V, E)$ be an instance of k-densest subgraph, i.e., an undirected graph without isolated vertices in which we search for a subset of vertices of cardinality k that induces a subgraph with the maximal number of edges.

We construct a sequential unit vector packing instance $(\mathbf{S}, B)$ with windows of length 4 and with $d = |V|$ resources, i.e., $V = \{1, .., d\}$. There is precisely one window per edge $e = (u, v) \in E$, the sequence of this window is $s^e = uvuv$. The total bin size is set to $B = d + k$. This transformation can be carried out

in polynomial time and achieves, as shown in the following, that $(\mathbf{S}, B)$ can be solved with at most $|E| - \ell$ (additional) breakpoints if and only if G has a subgraph with k vertices containing at least ℓ edges. Because every window contains at most two vectors of the same resource, having more than two units of one resource does not influence the number of breakpoints. Every resource has to be assigned at least one unit because there are no isolated vertices in G. Hence, a solution to $(\mathbf{S}, B)$ is characterized by the subset R of resources to which two units are assigned (instead of one). By the choice of the total bin size we have $|R| = k$. A window does not induce a breakpoint if and only if both its resources are in R, otherwise it induces one breakpoint.

If G has a node induced subgraph G' of size k containing ℓ edges, we chose R to contain the vertices of G'. Then, every window corresponding to an edge of G' has no breakpoint, whereas all other windows have one. Hence, the number of (additional) breakpoints is $|E| - \ell$.

If $(\mathbf{S}, B)$ can be scheduled with at most $|E| - \ell$ breakpoints, define R as the resources for which there is more than one unit in the bin vector. Now $|R| \leq k$, and we can assume $|R| = k$ since the number of breakpoints only decreases if we change some resource from one to two, or decrease the number of resources to two. The set R defines a subgraph G' with k vertices of G. The number of edges is at least ℓ because only windows with both resources in R do not have a breakpoint. □

It remains to consider the original problem without pre-specified breakpoints.

Lemma 3. *Let $(\mathbf{S}, B)$ be an instance of sequential (unit) vector packing of length n with k pre-specified breakpoints and d resources ($d \leq B$) where every resource is used at least once. Then one can construct in polynomial time an instance $(\mathbf{S}', B')$ of the (unit) vector packing problem with bin size $B' = 3B + 2$ and $d' = d + 2B + 2$ resources that can be solved with at most $\ell + k$ breakpoints if and only if $(\mathbf{S}, B)$ can be solved with at most ℓ breakpoints.*

Proof. The general idea is to use for every prespecified breakpoint some "stopping" sequence F_i with the additional resources in such a way that the bound B' guarantees that there is precisely one breakpoint in F_i. This sequence F_i needs to enforce exactly one breakpoint, no matter whether or not there was a breakpoint within the previous window (i.e., between F_{i-1} and F_i).

We introduce two different stopping sequences F and G which we use alternatingly. This ensures that between two occurrences of F there is at least one breakpoint. The resources $1, \ldots, d$ of $(\mathbf{S}', B')$ are one-to-one the resources of $(\mathbf{S}, B)$. The $2B + 2$ additional resources are divided into two groups $f_1, \ldots, f_{B+1}$ for F and $g_1, \ldots, g_{B+1}$ for G. Every odd pre-specified breakpoint in $\mathbf{S}$ is replaced by the sequence $F := f_1 f_2 \cdots f_{B+1} f_1 f_2 \cdots f_{B+1}$ and all even breakpoints by the sequence $G := g_1 g_2 \cdots g_{B+1} g_1 g_2 \cdots g_{B+1}$.

To see the backward direction of the statement in the lemma, a bin $\mathbf{b}$ for $(\mathbf{S}, B)$ resulting in ℓ breakpoints can be augmented to a bin vector $\mathbf{b}'$ for $(\mathbf{S}', B')$ by adding one unit for each of the new resources. This does not exceed the bound B'. Now, in $(\mathbf{S}', B')$ there will be the original breakpoints and a breakpoint in the

middle of each inserted sequence. This shows that $\mathbf{b}'$ results in $\ell + k$ breakpoints for $(\mathbf{S}', B')$, as claimed.

To consider the forward direction, let $\mathbf{b}'$ be a solution to $(\mathbf{S}', B')$. Because every resource must be available at least once, and $B'-d' = 3B+2-(d+2B+2) = B - d$, at most $B - d < B$ entries of $\mathbf{b}'$ can be more than one. Therefore, at least one of the resources f_i is available only once, and at least one of the resources g_j is available only once. Hence, there must be at least one breakpoint within each of the k inserted stopping sequences. Let $k + \ell$ be the number of breakpoints induced by $\mathbf{b}'$ and $\mathbf{b}$ the projection of $\mathbf{b}'$ to the original resources. Since all resources must have at least one unit and by choice of B' and d' we know that $\mathbf{b}$ sums to less than B. Now, if a subsequence of $\mathbf{S}$ not containing any f or g resources can be packed with the resources $\mathbf{b}'$, this subsequence can also be packed with $\mathbf{b}$. Hence, $\mathbf{b}$ does not induce more than ℓ breakpoints in the instance $(\mathbf{S}, B)$ with pre-specified breakpoints. □

Theorem 3. *The sequential unit vector packing problem is strongly $\mathcal{NP}$-hard.*

Proof. By Lemma 2 and Lemma 3, with the additional observation that all used numbers are polynomial in the size of the original graph. □

For a discussion of polynomially solvable cases and issues related to fixed parameter tractibility, see [1].

3 Practical Algorithms

Both the problem presented in the introduction and the original industry problem are bounded size SUVP problems. For this reason, we focus on this variant when considering practical algorithms.

Greedy Algorithms. We describe two natural greedy heuristics for sequential unit vector packing. Recall that we denote by $\kappa(\mathbf{b}, \mathbf{S})$ the minimal number of breakpoints needed for a fixed bin vector $\mathbf{b}$ and given $(\mathbf{S}, B)$. Observe that it is relatively easy to calculate $\kappa(\mathbf{b}, \mathbf{S})$ in linear time (see end of this section). The two greedy algorithms we discuss here are: Greedy-Grow and Greedy-Shrink. Greedy-Grow grows the bin vector starting with the all one vector. In each step it increases some resource by an amount of 1 until the total bin size B is reached, greedily choosing the resource whose increment improves $\kappa(\mathbf{b}, \mathbf{S})$ the most. Greedy-Shrink shrinks the bin vector starting with a bin vector $\mathbf{b} = \sum_{i=1}^{n} \mathbf{s}_i$, which yields $\kappa(\mathbf{b}, \mathbf{S}) = 0$ but initially ignores the bin size B. In each step it then decreases some resource by an amount of 1 until the total bin size B is reached, greedily choosing the resource whose decrement worsens $\kappa(\mathbf{b}, \mathbf{S})$ the least.

In the light of the following observations (for proofs see [1]) it is important to specify the tie-breaking rule for the case that there is no improvement at all after the addition of a resource. Greedy-Grow can be forced to produce a solution only by this tie-breaking rule, which is an indicator for its bad performance.

Observation 1. *Given any instance* $(\mathbf{S}, B)$ *to bounded size SVP, this instance can be modified to an instance* $(\mathbf{S}', B')$*, with* $n' = n, d' = 2d, B' = 2B$ *such that all of* Greedy-Grow*'s choices of which resource to add depend entirely on the tie-breaking rule.*

It follows that Greedy-Grow with an unspecified tie-breaking rule can be led to produce arbitrarily bad solutions. Also Greedy-Shrink can produce bad solutions depending on the tie-breaking scheme.

Observation 2. *There are instances with d resources on which the solution produced by* Greedy-Shrink *is a factor of* $\lfloor d/2 \rfloor$ *worse than the optimal solution, if the tie-breaking rule can be chosen by the adversary.*

For the experiments in [1] we use for both heuristics a *round-robin* tie-breaking rule that cycles through the resources. Every time a tie occurs it chooses the cyclic successor of the resource that was increased (decreased) in the last tie.

Enumeration Heuristic. We present an enumeration heuristic for integral vectors $\mathbf{s}_i \in \mathbb{N}^d$, $i \in \{1, \ldots, n\}$, that is inspired by a variant of Schöning's 3-SAT algorithm [11] that searches the complete hamming balls of radius $\lfloor n/4 \rfloor$ around randomly chosen assignments, see [4]. The following algorithm uses a similar combination of randomized guessing and complete enumeration of parts of the solution space that are exponentially smaller than the whole solution space. The idea is to guess uniformly at random (u.a.r.) subsequences $\mathbf{S}_{i_1,i_2}$ of the sequence that do not incur a breakpoint in a fixed optimal solution $\mathbf{b}_{\text{opt}}$. For such a subsequence we know that $\mathbf{b}_{\text{opt}} \geq \mathbf{w}_{i_1,i_2}$. In particular, if we know a whole set W of such total demand vectors that all come from subsequences without breakpoints for $\mathbf{b}_{\text{opt}}$, we know that $\mathbf{b}_{\text{opt}} \geq \max_{\mathbf{w} \in W} \mathbf{w}$ must hold for a component-wise maximum. This idea leads to the Randomized Heuristic Enumeration (RHE):

Phase 1: Start with a "lower bound vector" $\mathbf{t} = 0$. For a given subsequence length ssl and a number p of repetitions, in each of p rounds choose $\underline{\sigma}_i =_{\text{u.a.r}} \{0, \ldots, n - \text{ssl}\}$, set $\overline{\sigma}_i = \underline{\sigma}_i + \text{ssl}$, and then set $\mathbf{t} = \max\{\mathbf{t}, \mathbf{w}_{\underline{\sigma}_i, \overline{\sigma}_i}\}$.

Phase 2: Find a bin vector $\mathbf{b}$ of total size B with $\mathbf{b} \geq \mathbf{t}$ that minimizes $\kappa(\mathbf{b}, \mathbf{S})$. Do this by enumerating all $\mathbf{b} \geq \mathbf{t}$ of total size B.

It is straight-forward to analyze the success probability of this algorithm if we relate the subsequence length to an estimate k' of the minimum number of breakpoints k. We give experimental evidence that the algorithm performs well and present an efficient algorithm for the enumeration in Phase 2, see [1].

Evaluation. For demand vectors $\mathbf{s}_i \in \mathbb{Q}_+^d$, $i \in \{1, \ldots, n\}$, the evaluation of a given bin vector $\mathbf{b}$, i.e., computing $\kappa(\mathbf{b}, \mathbf{S})$, can be done in the obvious way in $O(n \cdot d)$ time. With a preprocessing phase and some algorithmic engineering we can show the following theorem (see [1] for a complete discussion).

Theorem 4. *Given a sequence* $\mathbf{S}$ *and a bin vector* $\mathbf{b}$ *we can construct a data structure with* $O(n \cdot d \cdot B)$ *space and preprocessing time such that the evaluation*

of $\kappa(\mathbf{b}, \mathbf{S})$ *for sequential vector packing takes* $O(\kappa(\mathbf{b}, \mathbf{S}) \cdot d)$ *time. For sequential* unit *vector packing only* $O(n)$ *space and preprocessing time is needed.*

4 Conclusion

In this paper, we have introduced the sequential vector packing problem, presented $\mathcal{NP}$-hardness proofs for different variants, approximation algorithms, and several heuristics. The most interesting open challenges are probably to find an approximation algorithm for bounded size SVP and inapproximability results.

References

[1] Cieliebak, M., Hall, A., Jacob, R., Nunkesser, M.: Sequential vector packing. DELIS TR 0335, ETH Zurich (2006)
[2] Coffman Jr., E., Garey, M.R., Johnson, D.S.: Algorithm Design for Computer System Design. In: Approximation Algorithms for Bin Packing: An updated Survey, pp. 49–106. Springer, Heidelberg (1984)
[3] Coppersmith, D., Raghavan, P.: Multidimensional on-line bin packing: Algorithms and worst-case analysis. Operations Research Letters 4, 48–57 (1989)
[4] Dantsin, E., Goerdt, A., Hirsch, E.A., Kannan, R., Kleinberg, J.M., Papadimitriou, C.H., Raghavan, P., Schöning, U.: A deterministic (2-2/(k+1))n algorithm for k-sat based on local search. Theoretical Computer Science 289(1), 69–83 (2002)
[5] Feige, U., Peleg, D., Kortsarz, G.: The dense k-subgraph problem. Algorithmica 29(3), 410–421 (2001)
[6] Galambos, G., Woeginger, G.J.: On-line bin packing—a restricted survey. Mathematical Methods of Operations Research 42(1), 25–45 (1995)
[7] Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman, San Francisco (1979)
[8] Approximation Algorithms. In: Hochbaum, D.S. (ed.) Approximation Algorithms For Bin Packing: A Survey, pp. 46–93. PWS Publishing Company (1997)
[9] Kalyanasundaram, B., Pruhs, K.: Speed is as powerful as clairvoyance. Journal of the ACM 47, 617–643 (2000)
[10] Phillips, C., Stein, C., Torng, E., Wein, J.: Optimal time-critical scheduling via resource augmentation. In: STOC. Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pp. 140–149. ACM Press, New York (1997)
[11] Schöning, U.: A probabilistic algorithm for k-SAT based on limited local search and restart. Algorithmica 32, 615–623 (2002)
[12] Wee, T.S., Magazine, M.J.: Assembly line balancing as generalized bin-packing. Operations Research Letters 1, 56–58 (1982)
[13] Yang, J., Leung, J.Y.-T.: The ordered open-end bin-packing problem. Operations Research 51(5), 759–770 (2003)

A Tighter Analysis of Set Cover Greedy Algorithm for Test Set

Peng Cui

Renmin University of China, Beijing 100872, China
cuipeng@ruc.edu.cn

Abstract. Set cover greedy algorithm is a natural approximation algorithm for test set problem. This paper gives a precise and tighter analysis of approximation ratio of this algorithm. The author improves the approximation ratio $2\ln n$ directly derived from set cover to $1.14\ln n$ by applying potential function technique of derandomization method. In addition, the author gives a nontrivial lower bound $(1+\alpha)\ln n$ of approximation ratio, where α is a positive constant. This lower bound, together with the matching bound of information content heuristic, confirms the fact information content heuristic is slightly better than set cover greedy algorithm in worst case.

1 Introduction

The test set problem is NP-hard. The polynomial time approximation algorithms using in practice includes "greedy" heuristics implemented by set cover criterion or by information criterion[1]. Test set is not approximable within $(1-\varepsilon)\ln n$ for any $\varepsilon > 0$ unless $NP \subset DTIME(n^{\log\log n})$[2,3]. Recently, the authors of [3] design a new information type greedy algorithm, information content heuristic (ICH for short), and prove its approximation ratio $\ln n+1$, which almost matches the inapproximability results.

The set cover greedy algorithm (SGA for short) is a natural approximation algorithm for test set. In practice, its average performance is virtually the same as information type greedy algorithms[1,4]. The approximation ratio $2\ln n$ of SGA is obtained by transforming the test set problem as a set cover problem, and the lower bound $(1-o(1))\ln n$ of approximation ratio is derived by a reverse reduction[2].

Oblivious rounding, a derandomization technique to derive simple greedy algorithm for set cover problems by conditional probabilities is introduced in [5]. Young observes the number of elements uncovered is an "potential function" and the approximation algorithm only need to drive down the potential function at each step, thus gives another proof of the well-known approximation ratio $\ln n + 1$.

In this paper, the author presents a tighter analysis of SGA. We uses the potential function technique of derandomization method in [5] to improve the approximation ratio $2\ln n$ directly derived from set cover to $1.14\ln n$, and construct instances to give nontrivial lower bound $(1+\alpha)\ln n$ of approximation

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 24–35, 2007.

ratio, where α is a positive constant. The latter result confirms the fact ICH is slightly better than SGA in worst case. In the analysis, the author refers to the tight analysis of the greedy algorithm for set cover of [6].

In Section 2, some definitions, notations and basic facts are given. In Section 3, the author analyzes differentiation distribution of test set and gives two propositions. In Section 4, the author uses the potential function method to prove the improved approximation ratio. In Section 5, the author shows the nontrivial lower bound of approximation ratio. Section 6 is some discussions.

2 Preliminaries

The input of test set problem consists of a set of items (called universe) S with $|S| = n$, and $\mathcal{T}$, a collection of subsets (called tests) of S. An item pair is a set of two different items. A test T differentiates item pair a if $|T \cap a| = 1$. $\mathcal{T}$ is a test set of S if tests in $\mathcal{T}$ differentiate any item pair of S. The objective is to find the test set $\mathcal{T}' \subseteq \mathcal{T}$ with minimum cardinality.

Definition 1 (Test Set Problem)
Input: S, $\mathcal{T}$;
Feasible Solution: *test set* $\mathcal{T}'$, $\mathcal{T}' \subseteq \mathcal{T}$;
Measure: $|\mathcal{T}'|$;
Goal: *minimize.*

In an instance of test set problem, there exist $\binom{n}{2}$ different item pairs. Let i, j be two different items, and S_1, S_2 are two disjoint sets. If $i, j \in S_1$, we say $\{i, j\}$ is an item pair inside of S_1, and if $i \in S_1$ and $i \in S_2$, we say $\{i, j\}$ is an item pair between S_1 and S_2.

For any test T, let $T^+ = T$ and $T^- = S - T$. Let $\mathcal{T}^*$ is an optimal test set, denote $m^* = |\mathcal{T}^*|$. We use $a \perp T$ to represent the fact T differentiates a and $a \parallel T$ to represent the fact T does not differentiate a. We use $a \perp \mathcal{T}$ to represent the fact at least one test in $\mathcal{T}$ differentiates a, $a \parallel \mathcal{T}$ to represent the fact any test in $\mathcal{T}$ does not differentiate a, and $\perp (a, \mathcal{T})$ to represent the number of tests in $\mathcal{T}$ that differentiate a.

Fact 1. *For three different items i, j and k, if $\{i, j\} \parallel \mathcal{T}$ and $\{i, k\} \parallel \mathcal{T}$, then $\{j, k\} \parallel \mathcal{T}$.*

Definition 2 (Equivalent Relation Induced by $\bar{\mathcal{T}}$)
Given $\bar{\mathcal{T}}$, we define a binary relation $\backsim_{\bar{\mathcal{T}}}$ on S: for two item i, j, $i \backsim_{\bar{\mathcal{T}}} j$ iff $\{i, j\} \parallel \bar{\mathcal{T}}$. By Fact 1, $\backsim_{\bar{\mathcal{T}}}$ is an equivalent relation. The equivalent classes containing i is denoted as $[i]$.

Fact 2. *If $\mathcal{T}$ is a test set, then $|\mathcal{T}| \geq \log_2 n$.*

Definition 3 (Compact Test Set)
Test set $\mathcal{T}$ with $|\mathcal{T}| = \log_2 n$ is called a compact test set.

Clearly, If $\mathcal{T}$ is a compact test set, then $|S| = 2^q, q \in Z^+$, and for any $T_1, \cdots, T_i \in \mathcal{T}$, $|T_1^{s_1} \cap \cdots \cap T_i^{s_i}| = n/2^i$, where $s_1, \cdots, s_i \in \{+, -\}$.

Fact 3. *If $\mathcal{T}$ is a minimal test set, then $|\mathcal{T}| \leq n - 1$.*

Fact 4. *For any $S_1 \cap S_2 = \varnothing$, $|S_1| \log_2 |S_1| + |S_1| \log_2 |S_1| \leq |S_1 \cup S_2| \log_2 |S_1 \cup S_2|$.*

Test set can be transformed to set cover in a natural way. Let $(S, \mathcal{T})$ be an instance of test set, we construct an instance $(U, \mathcal{C})$ of set cover, where $U = \{\{i, j\} | i, j \in S, i \neq j\}$, and

$$\mathcal{C} = \{c(T) | T \in \mathcal{T}\}, c(T) = \{\{i, j\} | i \in T, j \in T^-\}$$

Clearly, $\mathcal{T}'$ is a test set of S iff $\mathcal{C}' = \{c(T) | T \in \mathcal{T}'\}$ is a set cover of U.

Definition 4 (Differentiation Measure)
The differentiation measure of $\bar{\mathcal{T}}$, $\#(\bar{\mathcal{T}})$, is defined as the number of item pairs not differentiated by $\bar{\mathcal{T}}$.

SGA can be described as:

Algorithm 1. SGA
Input: S,$\mathcal{T}$;
Output: a test set of S;
begin
  $\bar{\mathcal{T}} \leftarrow \varnothing$;
  while $\#(\bar{\mathcal{T}}) > 0$ **do**
    select T in $\mathcal{T} - \bar{\mathcal{T}}$ minimizing $\#(\bar{\mathcal{T}} \cup \{T\})$;
    $\bar{\mathcal{T}} \leftarrow \bar{\mathcal{T}} \cup \{T\}$;
  endwhile
  return $\bar{\mathcal{T}}$
end

In SGA, we call $\bar{\mathcal{T}}$ the partial test set. The greedy algorithm for set cover has approximation ratio $\ln N - \ln \ln N + \Theta(1)$ by [6]. Using the natural transformation, we immediately obtain the approximation ratio $2 \ln n$ of SGA.

We give two facts for proof of Section 4.

Fact 5. *For any $0 < x < 1$, $(1 - x)^{1/x} < 1/e$.*

Denote $\phi_a(x) = \frac{1}{x}(\ln x - a)$.

Fact 6. *For any $x > 1$, $\phi_1(x) \leq 1/e^2 = 0.135 \cdots$.*

3 Differentiation Distribution

In this section, the author analyzes the distribution of times for which item pairs are differentiated in instances of test set, especially the relationship between the differentiation distribution and the size of the optimal test set.

Lemma 1. *Given $T \subseteq S$, suppose $\mathcal{T}'$ is a test set for T and a test set for T^-, then at most $\min(|T|, |T^-|)$ item pairs between T and T^- are not differentiated by any test in $\mathcal{T}'$.*

Proof. W.l.o.g, suppose $|T| \leq |T^-|$. We claim for any item $i \in T$, there exist at most one item j in T^- satisfying $\{i, j\} \parallel \mathcal{T}'$. Otherwise there exist two different items j, k in T^- such that $\{i, j\} \parallel \mathcal{T}'$ and $\{i, k\} \parallel \mathcal{T}'$, then by Fact 1 , $\{j, k\} \parallel \mathcal{T}'$, which contradicts $\mathcal{T}'$ is a test set for T^-. □

Proposition 1. *At most $n \log_2 n$ item pairs are differentiated by exactly one test in $\mathcal{T}^*$.*

Proof. Let B be the set of item pairs that are differentiated by exactly one test in $\mathcal{T}^*$. We prove $|B| \leq \log_2 n$ by induction. When $n = 1$, $|B| = 0 = n \log_2 n$. Suppose the proposition holds for any $n \leq h-1$, we prove the proposition holds for $n = h$.

Select $T \in \mathcal{T}^*$, then $|T| \leq h-1$, $|T^-| \leq h-1$. Clearly, $\mathcal{T}^*$ is a test set of T. By induction hypothesis, at most $|T| \log_2 |T|$ item pairs inside of T are differentiated by exactly one test in $\mathcal{T}^*$. Similarly, at most $|T^-| \log_2 |T^-|$ item pairs inside of T^- are differentiated by exactly one test in $\mathcal{T}^*$.

By Lemma 1, at most $\min(|T|, |T^-|)$ item pairs between T and T^- are not differentiated by any test in $\mathcal{T}^* - \{T\}$. Therefore at most $\min(|T|, |T^-|)$ item pairs between T and T^- are differentiated by exactly one test in $\mathcal{T}^*$.

W.l.o.g, suppose $|T| \leq |T^-|$, then

$$\begin{aligned} |B| &\leq |T| \log_2 |T| + |T^-| \log_2 |T^-| + T \\ &\leq |T| \log_2 (2|T|) + |T^-| \log_2 |T^-| \\ &\leq |T| \log_2 |S| + |T^-| \log_2 |S| \\ &\leq |S| \log_2 |S|. \end{aligned}$$

□

Lemma 2. *Given $T \subseteq S$, suppose $\mathcal{T}'$ is a test set for T and a test set for T^-, then at most $|S| \log_2 |S|$ item pairs between T and T^- are differentiated by exactly one test in $\mathcal{T}'$.*

Proof. Let B be the set of item pairs that are differentiated by exactly one test in $\mathcal{T}'$. We prove that $|B| \leq |S| \log_2 |S|$ by induction. When $|S| = 1$, $|B| = 0 = |S| \log_2 |S|$. Suppose the lemma holds for any $|S| \leq h-1$, we prove the lemma holds for $|S| = h$.

Select $T' \in \mathcal{T}'$, then $|T'| \leq h-1$, $|T'^-| \leq h-1$. Clearly, $\mathcal{T}' - \{T'\}$ is a test set of $T \cap T'$ and a test set of $T^- \cap T'$. By induction hypothesis, at most $|T'| \log_2 |T'|$ item pairs between $T \cap T'$ and $T^- \cap T'$ are differentiated by exactly one test in $\mathcal{T}'$. Similarly, at most $|T'^-| \log_2 |T'^-|$ item pairs between $T \cap T'^-$ and $T^- \cap T'^-$ are differentiated by exactly one test in $\mathcal{T}'$.

By Lemma 1, at most $\min(|T \cap T'|, |T^- \cap T'^-|)$ item pairs between $T \cap T'$ and $T^- \cap T'^-$ are differentiated by exactly one test in $\mathcal{T}'$, and at most

$\min(|T^- \cap T'|, |T \cap T'^-|)$ item pairs between $T^- \cap T'$ and $T \cap T'^-$ are differentiated by exactly one test in $\mathcal{T}'$

W.l.o.g, suppose $|T'| \leq |T'^-|$, clearly

$$|T'| \geq \min(|T \cap T'|, |T^- \cap T'^-|) + \min(|T^- \cap T'|, |T \cap T'^-|).$$

Therefore

$$\begin{aligned} |B| &\leq |T'| \log_2 |T'| + |T'^-| \log_2 |T'^-| + T' \\ &\leq |T'| \log_2 (2|T'|) + |T'^-| \log_2 |T'^-| \\ &\leq |T'| \log_2 |S| + |T'^-| \log_2 |S| \\ &\leq |S| \log_2 |S|. \end{aligned}$$

□

Lemma 3. *At most* $n \log_2 nm^{*t-1}$ *item pairs are differentiated by exactly* t *test in* $\mathcal{T}^*$*, where* $t \geq 2$*.*

Proof. Let B_t^* be the set of item pairs that are differentiated by exactly t test in $\mathcal{T}^*$. For any combination π of $t-1$ tests in $\mathcal{T}^*$, let B_π^* be the subset of B_t^* such that item pair in B_π^* is differentiated by any test in π.

Let $\backsim_\pi$ be the equivalent relation induced by π. For any equivalent class $[i]$, there exists at most one equivalent class $[j]$, such that any item pairs between $[i]$ and $[j]$ are differentiated by any test in π. By Lemma 2, at most $(|[i] \cup [j]|) \log_2 |[i] \cup [j]|$ item pairs between $[i]$ and $[j]$ are differentiated by exactly one test in $\mathcal{T}^* - \pi$, i.e., are differentiated by exactly t tests in $\mathcal{T}^*$. To sum up,

$$|B_\pi^*| = \sum_{[i],[j]} |[i] \cup [j]| \log_2 |[i] \cup [j]| \leq n \log_2 n \quad (Fact\ 4).$$

Therefore,

$$|B_t^*| \leq \sum_\pi |B_\pi| \leq \binom{m^*}{t-1} n \log_2 n \leq n \log_2 nm^{*t-1}.$$

□

Proposition 2. *At most* $2n \log_2 nm^{*t-1}$ *item pairs are differentiated by at most* t *test in* $\mathcal{T}^*$*, where* $t \geq 2$*.*

Proof. Let B_t be the set of item pairs that are differentiated by at most t test in $\mathcal{T}^*$. When $m^* \geq 2$, by Lemma 3 and Fact 2,

$$\begin{aligned} |B_t| &= |B_1^*| + |B_2^*| + \cdots + |B_t^*| \\ &\leq n \log_2 n(1 + m^* + \cdots + m^{*t-1}) \\ &\leq n \log_2 nm^{*t-1}(1 + \frac{1}{m^* - 1}) \\ &\leq 2n \log_2 nm^{*t-1}. \end{aligned}$$

□

4 Improved Approximation Ratio

In this section, the author uses the potential function method to derive improved approximation ratio of SGA for test set. Our proof is based on the idea to "balance" the potential function by appending a negative term to the differentiation measure.

Lemma 4 (Lemma 2 in [6]). *The size of set cover returned by the greedy algorithm is at most* $M^*(\ln N - \ln M^* + 1)$*, where* N *is the size of the universe, and* M^* *is the size of the optimal set cover.*

Theorem 1. *The approximation ratio of SGA for test set can be* $1.14 \ln n$.

Proof. Let I is the integer satisfying

$$2n \log_2 nm^{*I-1} < \binom{n}{2} \leq 2n \log_2 nm^{*I},$$

then $I = \lceil \ln \frac{n-1}{4 \log_2 n} / \ln m^* \rceil$.

Let $\#_0 = 1$, $\#_1 = n \log_2 n$, $\#_t = 2n \log_2 nm^{*t-1}, 2 \leq t \leq I$, and $\#_{I+1} = n(n-1)/2$.

We divide a run of the algorithm into $I+1$ phases, from Phase $I+1$ to Phase 1. In Phase t, $I+1 \geq t \geq 1$, the algorithm runs until $\#(\bar{\mathcal{T}}) < \#_{t-1}$.

Let the set of selected tests in Phase t is $\mathcal{T}_t$, and the partial test set when Phase t stops is $\bar{\mathcal{T}}_t$, $1 \leq t \leq I+1$. Then $\bar{\mathcal{T}}_t = \cup_{t \leq u \leq I+1} \mathcal{T}_u$, $1 \leq t \leq I+1$, and the returned test set is $\mathcal{T}' = \cup_{1 \leq t \leq I+1} \mathcal{T}_t$. Let $\bar{\mathcal{T}}_{I+2} = \varnothing$. If $\mathcal{T}_t \neq \varnothing$, let the last selected test in Phase t is T'_t.

Let $k_t = \frac{1}{t} \ln \frac{t\#_t}{\#_{t-1}} m^*$, $2 \leq t \leq I+1$.

In Phase t, for some t, $I+1 \geq t \geq 2$, define the potential function be

$$f(\bar{\mathcal{T}}) = (\#(\bar{\mathcal{T}}) - \frac{t-1}{t} \#_{t-1})(1 - \frac{t}{m^*})^{k_t - |\bar{\mathcal{T}} - \bar{\mathcal{T}}_{t+1}|}.$$

Given $\bar{\mathcal{T}}$, let p denote the probability distribution on tests in $\mathcal{T}^*$: draw one test uniformly from $\mathcal{T}^*$. For any $T \in \mathcal{T}^*$, the probability of drawing T is $p(T) = \frac{1}{m^*}$.

For any item pair a,

$$\sum_{T: T \in \mathcal{T}^* - \bar{\mathcal{T}}, a \perp T} p(T) = \frac{\perp (a, \mathcal{T}^*)}{m^*}.$$

By Fact 5,

$$\begin{aligned} f(\bar{\mathcal{T}}_{t+1}) &= (\#_t - \frac{t-1}{t} \#_{t-1})(1 - \frac{t}{m^*})^{k_t} \\ &< \#_t / \frac{t\#_t}{\#_{t-1}} = \frac{\#_{t-1}}{t}. \end{aligned}$$

By the definition of $f(\bar{\mathcal{T}})$ and the facts $p(T) \geq 0$ and $\sum_{T\in\mathcal{T}^*} p(T) = 1$,

$$\begin{aligned}
&\min_{T\in\mathcal{T}} f(\bar{\mathcal{T}} \cup \{T\}) \\
&\leq \min_{T\in\mathcal{T}^*} f(\bar{\mathcal{T}} \cup \{T\}) \\
&\leq \sum_{T\in\mathcal{T}^*} (p(T) f(\bar{\mathcal{T}} \cup \{T\})) \\
&= \sum_{T\in\mathcal{T}^*} (p(T)(\#(\bar{\mathcal{T}}) - \#(T, \bar{\mathcal{T}}) - \frac{t-1}{t}\#_{t-1}))(1 - \frac{t}{m^*})^{k_t - |\bar{\mathcal{T}} - \bar{\mathcal{T}}_{t+1}| - 1} \\
&= (\#(\bar{\mathcal{T}}) - \frac{t-1}{t}\#_{t-1} - \sum_{a\| \bar{\mathcal{T}}} \sum_{T:T\in\mathcal{T}^*, a\perp T} p(T))(1 - \frac{t}{m^*})^{k_t - |\bar{\mathcal{T}} - \bar{\mathcal{T}}_{t+1}| - 1}
\end{aligned}$$

and

$$\begin{aligned}
&\sum_{a\| \bar{\mathcal{T}}} \sum_{T:T\in\mathcal{T}^*, a\perp T} p(T) \\
&= \sum_{\perp(a,\mathcal{T}^*)\leq t-1} \sum_{T:T\in\mathcal{T}^*, a\perp T} p(T) + \sum_{\perp(a,\mathcal{T}^*)\geq t} \sum_{T:T\in\mathcal{T}^*, a\perp T} p(T) \\
&\geq \sum_{\perp(a,\mathcal{T}^*)\leq t-1} \frac{1}{m^*} + \sum_{\perp(a,\mathcal{T}^*)\geq t} \frac{t}{m^*} \\
&= \sum_{a\| \bar{\mathcal{T}}} \frac{t}{m^*} - \sum_{\perp(a,\mathcal{T}^*)\leq t-1} \frac{t-1}{m^*} \\
&\geq (\#(\bar{\mathcal{T}}) - \frac{t-1}{t}\#_{t-1})\frac{t}{m^*} \quad (\textit{Proposition 1 and Proposition 2}).
\end{aligned}$$

Therefore,

$$\min_{T\in\mathcal{T}} f(\bar{\mathcal{T}} \cup \{T\}) \leq (\#(\bar{\mathcal{T}}) - \frac{t-1}{t}\#_{t-1})(1 - \frac{t}{m^*})(1 - \frac{t}{m^*})^{k_t - |\bar{\mathcal{T}} - \bar{\mathcal{T}}_{t+1}| - 1} = f(\bar{\mathcal{T}}).$$

During Phase t, the algorithm selects T in $\mathcal{T}$ to minimize $f(\bar{\mathcal{T}} \cup \{T\})$. Therefore, $f(\bar{\mathcal{T}}_t - \{T'_t\}) \leq f(\bar{\mathcal{T}}_{t+1}) < \frac{\#_{t-1}}{t}$.

By definition of Phase t, $\#(\bar{\mathcal{T}}_t - \{T'_t\}) \geq \#_{t-1}$. Hence

$$f(\bar{\mathcal{T}}_t - \{T'_t\}) \geq (\#_{t-1} - \frac{t-1}{t}\#_{t-1})(1 - \frac{t}{m^*})^{k_t - |\mathcal{T}_t - \{T'_t\}|} = \frac{\#_{t-1}}{t}(1 - \frac{t}{m^*})^{k_t - |\mathcal{T}_t - \{T'_t\}|}.$$

Therefore, $(1 - \frac{t}{m^*})^{k_t - |\mathcal{T}_t - \{T'_t\}|} < 1$, thus $|\mathcal{T}_t - \{T'_t\}| < k_t$, and $|\mathcal{T}_t| < k_t + 1$.

To sum up,

$$|\bar{\mathcal{T}}_2| < \sum_{2\leq t\leq I+1} k_t + I.$$

When all Phase t, $I + 1 \geq t \geq 2$, end, consider the instance of set cover $(U, \mathcal{C})$, where $U = \{a | a \parallel \bar{\mathcal{T}}_2\}$ and $\mathcal{C} = \{c(T) | c(T) \cap U \neq \varnothing\}$. Clearly, $|U| < \#_1$.

Let M^* be the size of the optimal set cover of this instance. Then $|M^*| \leq m^*$. Consider the following two cases: (a)$|M^*| \leq m^*/2$; (b)$|M^*| > m^*/2$.

In case (a),

$$|\mathcal{T}_1| \leq M^*(\ln \#_1 + 1) \leq m^*(\frac{1}{2} + o(1)) \ln n,$$

and

$$\begin{aligned}
|\bar{\mathcal{T}}_2| &= m^*(\sum_{2 \leq t \leq I+1} \frac{1}{t} \ln \frac{\#_t}{\#_{t-1}} + \sum_{2 \leq t \leq I+1} \frac{\ln t}{t}) + I \\
&\leq m^*(\sum_{2 \leq t \leq I+1} \frac{1}{2} \ln \frac{\#_t}{\#_{t-1}} + \frac{1}{2} \ln^2(I+2)) + I \\
&= m^*(\frac{1}{2} + o(1)) \ln n.
\end{aligned}$$

Hence

$$|\mathcal{T}'| = |\bar{\mathcal{T}}_2| + |\mathcal{T}_1| = m^*(1 + o(1)) \ln n.$$

In case (b), by Lemma 4,

$$\begin{aligned}
|\mathcal{T}_1| &\leq M^*(\ln \#_1 - \ln M^* + 1) \\
&\leq m^*(\ln \#_1 - \ln m^* + \ln 2 + 1) \\
&\leq m^*((1 + o(1)) \ln n - \ln m^*),
\end{aligned}$$

and

$$\begin{aligned}
|\bar{\mathcal{T}}_2| &\leq m^*(\sum_{2 \leq t \leq I+1} \frac{\ln m^*}{t} + \ln 2 + \sum_{2 \leq t \leq I+1} \frac{\ln t}{t}) + I \\
&\leq m^*(\ln I \ln m^* + \ln 2 + \frac{1}{2} \ln^2(I+2)) + I \\
&\leq m^*(\ln \frac{\ln n}{\ln m^*} \ln m^* + o(1) \ln n).
\end{aligned}$$

Hence

$$|\mathcal{T}'| = |\bar{\mathcal{T}}_2| + |\mathcal{T}_1| \leq m^*(1 + \phi_1(\frac{\ln n}{\ln m^*}) + o(1)) \ln n.$$

By Fact 6,

$$|\mathcal{T}'| \leq m^*(1.135 \cdots + o(1)) \ln n.$$

□

5 Lower Bound

First we consider set cover and show a lemma about the lower bound of approximation ratio as a corollary of [6].

Lemma 5. *Given N, the size of the universe N, and M^*, the size of the optimal set cover, there exists instance of set cover such that the size of set cover returned by the greedy algorithm is at least $(M^* - 1)(\ln N - \ln M^*)$.*

Proof. The "greedy numbers" $N(k,l)$ is the size of smallest set U for which it is possible to have a cover of U with $M^* = l$ and $M' = k$, where M' is the size of the cover returned by the greedy algorithm.

By the recursive formulas $N(k+1,l) = \lceil \frac{l}{l-1} N(k,l) \rceil$, and $N(l,l) = l$, we have

$$N(k,l) \leq (\frac{l}{l-1})^{k-l} N(l,l) + \sum_{1 \leq i \leq k-l} (\frac{l}{l-1})^{i-1} \leq 2e^{\frac{k-l}{l-1}} l$$

which gives

$$k \geq (\ln N(k,l) - \ln l - \ln 2)(l-1) + l \geq (\ln N(k,l) - \ln l)(l-1).$$

By [6], there exists an instance satisfying $N = N(k,l)$, $M^* = l$. □

In this section, we discuss on a variation of test set problem for brevity. Given disjoint sets $S^1, \cdots, S^r$ and $\mathcal{T}$, set of subsets of the universe $S = S^1 \cup \cdots \cup S^r$, we seek $\mathcal{T}' \subseteq \mathcal{T}$ with minimum cardinality which is test set of any S^i for $1 \leq i \leq r$. We denote the instance as $(S^i; \mathcal{T})$.

We could use the copy-split trick in [1,2] to split $S^1, \cdots, S^r$ such that the splitting overhead could be ignored. Hence our preference does not affect the lower bound result.

To prepare for constructing nontrivial instances, we consider the level t "atom" instances, and investigate SGA's behavior on the "atom" instances. Let the instance be $(S^z; \mathcal{T})$. The universe includes integral points in $(t+1)$-dimension Euclid space, $S^z = \{(x_1, \cdots, x_t, z) | x_i, z \in Z^+, 1 \leq x_i \leq 2^q, 1 \leq z \leq r2^{q-2}\}$.

$\mathcal{T} = \mathcal{T}^* \cup \mathcal{T}'$, and $\mathcal{T}^* = \mathcal{T}_1^* \cup \cdots \cup \mathcal{T}_t^*$. For any $1 \leq i \leq t$,$\mathcal{T}_i^*$ contains tests $T_{i,j}^*, 1 \leq j \leq 2^q$ each of which contains all the points in S^z with $x_i = j$.

$\mathcal{T}' = \mathcal{T}_1' \cup \cdots \cup \mathcal{T}_t'$. For any $1 \leq i \leq t$, let $\mathcal{T}_i' = \mathcal{T}_{i,1}' \cup \cdots \cup \mathcal{T}_{i,2^{q-2}}'$. Suppose $\hat{\mathcal{T}}_i$ is a compact test set for $\{x_i | 1 \leq x_i \leq 2^q\}$. For each $1 \leq i \leq t$ and $1 \leq j \leq 2^{q-2}$, $\mathcal{T}_{i,j}'$ contains test $T_{i,j,p}$ for $1 \leq p \leq q$ each of which contains points in S^z with $z = j \ mod \ 2^{q-2}$ and x_i in some test in $\hat{\mathcal{T}}_i$.

Proposition 3. *For the "atom" instances, SGA could select all tests in $\mathcal{T}'$.*

Proof. At the beginning of the algorithm, the differentiation measure of tests in $\mathcal{T}'$ is $r2^{2qt-2}$, and the differentiation measure of tests in $\mathcal{T}^*$ is

$$2^{q(t-1)}(2^{qt} - 2^{q(t-1)})r2^{q-2} = r2^{2qt-2} - r2^{2qt-q-2}.$$

The algorithm could first select tests in $\mathcal{T}_1'$. After that, the differentiation measure of tests in $\mathcal{T}' - \mathcal{T}_1'$ decreases by a factor 2, and the differentiation measure of tests in $\mathcal{T}^*$ decreases by a factor at least 2. Hence, the algorithm could subsequently select tests in $\mathcal{T}_2', \cdots, \mathcal{T}_t'$. □

Theorem 2. *The approximation ratio of SGA for test set has a lower bound $(1+\alpha)\ln n$, where α is a positive constant.*

Proof. We select N and M^* such that $M^* = J!2^q$, $M^{*J+1}|N$, $N \gg M^{*J+1}$ and $N \ll M^{*(1+\gamma)(J+1)}$, where $q \in Z^+$, J is a positive integer and γ is a arbitrarily small positive constant. Let $(U, \mathcal{C})$ be the instance in Lemma 5, $U = \{e_1, \cdots, e_N\}$, $\mathcal{C}^*$ is the optimal set cover, and $\mathcal{C}'$ is the set cover returned by the greedy algorithm. Let $S_0^i = \{e_i, f_i\}$ for $1 \le i \le N$. Construct instance $(S_0^i; \mathcal{T}_0)$ with $\mathcal{T}_0 = \mathcal{T}_0^* \cup \mathcal{T}_0'$, $\mathcal{T}_0^* = \mathcal{C}^*$ and $\mathcal{T}_0' = \mathcal{C}'$.

We construct a series of level t instances $(S_t^{y,z}; \mathcal{T}_t), 1 \le t \le J$, and combine them to an instance with $n = g(J)N$ and $m^* = M^*$. We intend to prove approximation ratio of $(S, \mathcal{T})$ could be $(1 - o(1) + \Omega(\phi_a(J)))\ln n$ for fixed J. When J is sufficiently large, the ratio is at least $(1+\alpha)\ln n$, where n is the size of the universe and α is a positive constant.

Construction of $(S_t^{y,z}; \mathcal{T}_t)$**.** Given t, we define $(S_t^{y,z}; \mathcal{T}_t)$ as assembly of "atom" instances. The universe includes $(\frac{J!}{t})^t t^{t-1} N$ integral points in $(2t+1)$-dimension Euclid space.

$S_t^{y,z} = \{(x_1, \cdots, x_t, y_1, \cdots, y_t, z) | x_i, y_i, z \in Z^+, 1 \le x_i \le 2^q, 1 \le y_i \le \frac{J!}{t}, 1 \le z \le \frac{t^{t-1}N}{2^{qt}}\}$.

$\mathcal{T}_t = \mathcal{T}_t^* \cup \mathcal{T}_t'$, $\mathcal{T}_t^* = \mathcal{T}_{t,1}^* \cup \cdots \cup \mathcal{T}_{t,t}^*$, and $\mathcal{T}_{t,i}^* = \bigcup_{1 \le y_i \le \frac{J!}{t}} \mathcal{T}_{t,i}^{*y_i}$ for $1 \le i \le t$. $\mathcal{T}_{t,i}^{*y_i}$ contains tests $T_{t,i,j}^{*y_i}$ for $1 \le j \le 2^q$ each of which contains all the points in $S_t^{y(y_i),z}$ with $x_i = j$. Clearly, $|\mathcal{T}^*| = M^*$.

$\mathcal{T}_t' = \mathcal{T}_{t,1}' \cup \cdots \cup \mathcal{T}_{t,t}'$, $\mathcal{T}_{t,i}' = \bigcup_{1 \le y_i \le \frac{J!}{t}} \mathcal{T}_{t,i}'^{y_i}$ for $1 \le i \le t$. Let $\mathcal{T}_{t,i}'^{y_i} = \mathcal{T}_{t,i,1}'^{y_i} \cup \cdots \cup \mathcal{T}_{t,i,2^{q-2}}'^{y_i}$. Suppose $\hat{\mathcal{T}}_i$ is a compact test set for $\{x_i | 1 \le x_i \le 2^q\}$. For each $1 \le i \le t$ and $1 \le j \le 2^{q-2}$, $\mathcal{T}_{t,i,j}'^{y_i}$ contains test $T_{t,i,j,p}^{y_i}$ for $1 \le p \le q$ each of which contains points in $S_t^{y(y_i),z}$ with $z = j \ mod\ 2^{q-2}$ and x_i in some test in $\hat{\mathcal{T}}_i$. Clearly $|\mathcal{T}_{t,i}'| = \frac{qM^*}{4t}$ for $1 \le i \le t$.

It is easy to prove for $(S_t^{y,z}; \mathcal{T}_t)$ the algorithm could select all the tests in $\mathcal{T}_{t,t}'$ before selecting any test in $\mathcal{T}_t^*$ (along the line of the proof of Proposition 3), and the differentiation measure of selected tests ranges from $\#_t^{begin} = M^{*t-1}N$ to $\#_t^{end} = 2tM^{*t-2}N$, for $t \ge 1$.

In addition, for $(S_0^i; \mathcal{T}_0)$, the algorithm could select all the tests in $\mathcal{T}_0'$, the the differentiation measure of selected tests ranges from $\#_0^{begin} = N/M^*$ to $\#_0^{end} = 1$ by [6].

Consequently, we combine $(S_t^{y,z}; \mathcal{T}_t), 1 \le t \le J$ and $(S_0^i; \mathcal{T}_0)$ to $(S_t^{y,z}, S_i^0; \mathcal{T})$. The size of universe is $n = N + \sum_1^J (\frac{J!}{t})^t t^{t-1} N \le (1 + J!^J)N$. Let $\mathcal{T} = \mathcal{T}^* \cup \mathcal{T}'$, and $\mathcal{T}' = \mathcal{T}_0' \cup \cdots \cup \mathcal{T}_J'$. We join tests in $\mathcal{T}_t^*$ for $0 \le t \le J$ one-by-one to obtain one test $\mathcal{T}^*$. Suppose $\mathcal{T}_t^* = \{T_{t,1}^*, \cdots, T_{t,M^*}^*\}$, $0 \le t \le J$, then $\mathcal{T}^* = \{T_{0,1}^* \cup \cdots \cup T_{J,1}^*, \cdots, T_{0,M^*}^* \cup \cdots \cup T_{J,M^*}^*\}$.

We modify $(S_t^{y,z}, S_0^i; \mathcal{T})$ by two operations: Expansion and Join. Tests in $\mathcal{T}_{t,t}'$ for any $1 \le t \le J$ are expanded by a factor 2, and tests in $\mathcal{T}_{u,v}'$ for $u > v$ are joined to tests in $\mathcal{T}_{t,t}'$.

Expansion. We modify the definition of $\mathcal{T}_{t,t}'$. $\mathcal{T}_{t,t}' = \bigcup_{y_t} \mathcal{T}_{t,t}'^{y_t}$. Let $\mathcal{T}_{t,t}'^{y_t} = \mathcal{T}_{t,t,1}'^{y_t} \cup \cdots \cup \mathcal{T}_{t,t,2^{q-3}}'^{y_t}$. Suppose $\hat{\mathcal{T}}_t$ is a compact test set for $\{x_t | 1 \le x_t \le 2^q\}$.

For each $1 \leq j \leq 2^{q-3}$, $\mathcal{T}'^{y_t}_{t,t,j}$ contains test $T^{y_t}_{t,t,j,p}$ for $1 \leq p \leq q$ each of which contains points in $S^{y(y_t),z}_t$ with $z = j \ mod \ 2^{q-3}$ and x_t in some test in $\hat{\mathcal{T}}_t$.

Join. For any $1 \leq u \leq J$ and $u \geq v \geq 1$, $\mathcal{T}'_{u,v} = \bigcup_{y_v} \{T^{y_v}_{u,v,j,p} | 1 \leq j \leq 2^{q-2}, 1 \leq p \leq q\}$. For any $2 \leq u \leq J$, join tests in $\mathcal{T}'_{u,v}$ for $1 \leq v \leq u-1$ one-by-one to tests in $\mathcal{T}'_{u-1,u-1}, \mathcal{T}'_{u-2,u-2}, \cdots, \mathcal{T}'_{1,1}$ according to the lexical order of (v, y_v, p, j) (v decreasing and y_v, p, j increasing) until these tests are exhausted.

First, for $J \geq t \geq 2$, we compare the differentiation measure of tests in $\mathcal{T}'_{t,t}$ to that of tests in $\mathcal{T}'_{t-1,t-1}$. Number of item pairs inside of any $S^{y,z}_u$ for $J \geq u > t$ contributing to the former is no less than the latter. Let the number of item pairs inside of $S^{y,z}_t$ contributing to the former is $\#_t$, number of item pairs inside of $S^{y,z}_t$ contributing to the latter is $\#'$, and number of item pairs inside of $S^{y,z}_{t-1}$ contributing to the latter is $\#_{t-1}$.

We have

$$\#_t = 2\#' \geq \#' + \#^{end}_t \geq \#' + 2\#^{begin}_{t-1} \geq \#' + \#_{t-1}.$$

Hence the differentiation measure of tests in $\mathcal{T}'_{t,t}$ is no less than that of tests in $\mathcal{T}'_{t-1,t-1}$ for $t \geq 2$. In addition, since $\#^{end}_1 = 2N/M^* > N/M^* = \#^{begin}_0$, the differentiation measure of tests in $\mathcal{T}'_{1,1}$ is no less than that of tests in $\mathcal{T}'_0$.

Next, for $J \geq t \geq 1$, we compare the differentiation measure of tests in $\mathcal{T}'_{t,t}$ to that of tests in $\mathcal{T}^*$. Number of item pairs inside of any $S^{y,z}_u$ for $J \geq u > t$ contributing to the former is no less than the latter. Let the number of item pairs inside of $S^{y,z}_t$ contributing to the former is $\#_t$, number of item pairs inside of $S^{y,z}_t$ contributing to the latter is $\#^*$, and number of item pairs inside of $S^{y,z}_i$ contributing to the latter is $\#^*_i$, $1 \leq i \leq t-1$.

We have

$$\#_t \geq 2\#^* \geq \#^* + \sum_{i=1}^{t-1} \frac{\#^*}{2^{t-i}} \geq \#^* + \sum_{i=1}^{t-1} \#^*_i.$$

Hence the differentiation measure of tests in $\mathcal{T}'_{t,t}$ is no less than that of any tests in $\mathcal{T}^*$ for $t \geq 1$.

We conclude the algorithm could selects all tests in $\mathcal{T}'_{t,t}$ instead of any tests in $\mathcal{T}'_{t-1,t-1}$ and any tests in $\mathcal{T}^*$. We conclude the algorithm could select all $\frac{qM^*}{8t}$ tests in $\mathcal{T}'_{t,t}$, for $J \geq t \geq 1$, and select all tests in $\mathcal{T}'_0$.

The size of returned test set is

$$\begin{aligned} |\mathcal{T}'| &= M^*((1-o(1))(\ln N - \ln M^*) + \frac{qH_J}{8}) \\ &= M^*((1-o(1))\ln N + (\frac{H_J}{8\ln 2} - 1)\ln M^*) \\ &= m^*(1 - o(1) + \Omega(\phi_a(J)))\ln n, \end{aligned}$$

where $a = 8\ln 2$. □

6 Discussion

We note this is the first time to distinguish precisely the worst case performance guarantees of two type "greedy algorithm" implemented by set cover criterion and by information criterion. In fact, we definitely show the pattern of instances on which ICH performs better than SGA.

In an preceding paper[7], we prove the approximation ratio of SGA can be $(1.5 + o(1)) \ln n$. Unlike this paper, the proof can be extended to weighted case, where each test is assigned a positive weight, and the objective is to find a test set with minimum total weight.

In the minimum cost probe set problem[8] of bioinformatics, tests are replaced with partitions of items. The objective is to find a set of partitions with smallest cardinality to differentiate all item pairs. It is easily observed that the improved approximation ratio is still applicable to this generalized case.

Acknowledgements. The author would like to thank Tao Jiang and Tian Liu for their helpful comments.

References

1. Moret, B.M.E., Shipiro, H.D.: On minimizing a set of tests. SIAM Journal on Scientific and Statistical Computing 6, 983–1003 (1985)
2. De Bontridder, K.M.J., Halldórsson, B.V., Halldórsson, M.M., Hurkens, C.A.J., Lenstra, J.K., Ravi, R., Stougie, L.: Approximation algorithm for the test cover problems. Mathematical Programming-B 98, 477–491 (2003)
3. Berman, P., DasGupta, B., Kao, M.: Tight approximability results for test set problems in bioinformatics. Journal of Computer and System Sciences 71, 145–162 (2005)
4. DasGupta, B., Konwar, K., Mandoiu, I., Shvartsman, A.: Highly scalable algorithms for robust string barcoding. International Journal of Bioinformatics Research and Applications 1, 145–161 (2005)
5. Young, N.E.: Randomized rounding without solving the linear program. In: SODA 95. Sixth ACM-SIAM Symposium on Discrete Algorithms, pp. 170–178. ACM Press, New York (1995)
6. Slavík, P.: A tight analysis of the greedy algorithm for set cover. Journal of Algorithms 25, 237–254 (1997)
7. Cui, P., Liu, H.: Deep Approximation of Set Cover Greedy Algorithm for Test Set (in Chinese). Journal of Software 17, 1494–1500 (2006)
8. Borneman, J., Chrobak, M., Vedova, G.D., Figueora, A., Jiang, T.: Probe selection algorithms with applications in the analysis of microbial communities. Bioinformatics 17(Suppl. 1), S39–S48 (2001)

A More Effective Linear Kernelization for Cluster Editing

Jiong Guo*

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
guo@minet.uni-jena.de

Abstract. In the NP-hard Cluster Editing problem, we have as input an undirected graph G and an integer $k \geq 0$. The question is whether we can transform G, by inserting and deleting at most k edges, into a cluster graph, that is, a union of disjoint cliques. We first confirm a conjecture by Michael Fellows [IWPEC 2006] that there is a polynomial-time kernelization for Cluster Editing that leads to a problem kernel with at most $6k$ vertices. More precisely, we present a cubic-time algorithm that, given a graph G and an integer $k \geq 0$, finds a graph G' and an integer $k' \leq k$ such that G can be transformed into a cluster graph by at most k edge modifications iff G' can be transformed into a cluster graph by at most k' edge modifications, and the problem kernel G' has at most $6k$ vertices. So far, only a problem kernel of $24k$ vertices was known. Second, we show that this bound for the number of vertices of G' can be further improved to $4k$. Finally, we consider the variant of Cluster Editing where the number of cliques that the cluster graph can contain is stipulated to be a constant $d > 0$. We present a simple kernelization for this variant leaving a problem kernel of at most $(d+2)k+d$ vertices.

1 Introduction

Problem kernelization has been recognized as one of the most important contributions of fixed-parameter algorithmics to practical computing [12,16,20]. A *kernelization* is a polynomial-time algorithm that transforms a given instance I with parameter k of a problem P into a new instance I' with parameter $k' \leq k$ of P such that the original instance I is a yes-instance with parameter k iff the new instance I' is a yes-instance with parameter k' and $|I'| \leq g(k)$ for a function g. The instance I' is called the *problem kernel*. For instance, the derivation of a problem kernel of linear size, that is, function g is a linear function, for the Dominating Set problem on planar graphs [2] is one of the breakthroughs in the kernelization area. The problem kernel derived there consists of at most $335k$ vertices, where k denotes the domination number of the given graph, and this was subsequently improved by further refined analysis and some additional reduction rules to a size bound of $67k$ [6]. In this work, we are going to improve a

* Supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 36–47, 2007.

size bound of $24k$ vertices for a problem kernel for Cluster Editing to a size bound of $4k$. Moreover, we present improvements concerning the time complexity of the kernelization algorithm.

The edge modification problem Cluster Editing is defined as follows:

> **Input:** An undirected graph $G = (V, E)$ and an integer $k \geq 0$.
> **Question:** Can we transform G, by deleting and adding at most k edges, into a graph that consists of a disjoint union of cliques?

We call a graph consisting of disjoint cliques a *cluster graph*.

The study of Cluster Editing can be dated back to the 1980's. Křivánek and Morávek [18] showed that the so-called Hierarchical-Tree Clustering problem is NP-complete if the clustering tree has a height of at least 3. Cluster Editing can be easily reformulated as a Hierarchical-Tree Clustering problem where the clustering tree has height exactly 3. After that, motivated by some computational biology questions, Ben-Dor et al. [4] rediscovered this problem. Later, Shamir et al. [22] showed the NP-completeness of Cluster Editing. Bansal et al. [3] also introduced this problem as an important special case of the Correlation Clustering problem which is motivated by applications in machine learning and they also showed the NP-completeness of Cluster Editing. It is also worth to mention the work of Chen et al. [7] in the context of phylogenetic trees; among other things, they also derived that Cluster Editing is NP-complete.

Concerning the polynomial-time approximability of the optimization version of Cluster Editing, Charikar et al. [5] proved that there exists some constant $\epsilon > 0$ such that it is NP-hard to approximate Cluster Editing within a factor of $1 + \epsilon$. Moreover, they also provided a polynomial-time factor-4 approximation algorithm for this problem. A randomized expected factor-3 approximation algorithm has been given by Ailon et al. [1]. The first non-trivial fixed-parameter tractability results were given by Gramm et al. [15]. They presented a kernelization for this problem which runs in $O(n^3)$ time on an n-vertex graph and results in a problem kernel with $O(k^2)$ vertices. Moreover, they also gave an $O(2.27^k + n^3)$-time algorithm [15] for Cluster Editing. A practical implementation and an experimental evaluation of the algorithm given in [15] have been presented by Dehne et al. [8]. Very recently, the kernelization result of Gramm et al. has been improved by two research groups: Protti et al. [21] presented a kernelization running in $O(n+m)$ time on an n-vertex and m-edge graph that leaves also an $O(k^2)$-vertex graph. In his invited talk at IWPEC'06, Fellows [12,13] presented a polynomial-time kernelization algorithm for this problem which achieves a kernel with at most $24k$ vertices. This kernelization algorithm needs to solve an LP-formulation of Cluster Editing. Fellows conjectured that a $6k$-vertex problem kernel should exist.

In this paper, we also study the variant of Cluster Editing, denoted as Cluster Editing$[d]$, where one seeks for a set of at most k edge modifications that transform a given graph into a disjoint union of exactly d cliques for a constant d. For each $d \geq 2$, Shamir et al. [22] showed that Cluster Editing$[d]$ is NP-complete. A simple factor-3 approximation algorithm has been provided by

Bansal et al. [3]. As their main technical contribution, Giotis and Guruswami [14] proved that there exists a PTAS for Cluster Editing$[d]$ for every fixed $d \geq 2$. More precisely, they showed that Cluster Editing$[d]$ can be approximated within a factor of $1+\epsilon$ for arbitrary $\epsilon > 0$ in $n^{O(9^d/\epsilon^2)} \cdot \log n$ time. To our best knowledge, the parameterized complexity of Cluster Editing$[d]$ was unexplored so far.

Here, we confirm Fellows' conjecture by presenting an $O(n^3)$-time combinatorial algorithm which achieves a $6k$-vertex problem kernel for Cluster Editing. This algorithm is inspired by the "crown reduction rule" used in [12,13]. However, by way of contrast, we introduce the *critical clique* concept into the study of Cluster Editing. This concept played a key role in the fixed-parameter algorithms solving the so-called Closest Leaf Power problem [9,10] and it goes back to the work of Lin et al. [19]. It also turns out that with this concept the correctness proof of the algorithm becomes significantly simpler than in [12,13]. Moreover, we present a new $O(nm^2)$-time kernelization algorithm which achieves a problem kernel with at most $4k$ vertices. Finally, based on the critical clique concept, we show that Cluster Editing$[d]$ admits a problem kernel with at most $(d+2) \cdot k + d$ vertices. The corresponding kernelization algorithm runs in $O(m+n)$ time.

2 Preliminaries

In this work, we consider only undirected graphs without self-loops and multiple edges. The open (closed) neighborhood of a vertex v in graph $G = (V, E)$ is denoted by $N_G(v)$ ($N_G[v]$), while with $N_G^2(v)$ we denote the set of vertices in G which have a distance of exactly 2 to v. For a vertex subset $V' \subseteq V$, we use $G[V']$ to denote the subgraph of G induced by V', that is, $G[V'] = (V', \{e = \{u, v\} \mid (e \in E) \wedge (u \in V') \wedge (v \in V')\})$. We use $\triangle$ to denote the symmetric difference between two sets, that is, $A \triangle B = (A \setminus B) \cup (B \setminus A)$. A set C of vertices is called a *clique* if the induced graph $G[C]$ is a complete graph. Throughout this paper, let $n := |V|$ and $m := |E|$.

In the following, we introduce the concepts of *critical clique* and *critical clique graph* which have been used in dealing with leaf powers of graphs [19,10,9].

Definition 1. *A* critical clique *of a graph G is a clique K where the vertices of K all have the same sets of neighbors in $V \setminus K$, and K is maximal under this property.*

Definition 2. *Given a graph $G = (V, E)$, let $\mathcal{K}$ be the collection of its critical cliques. Then the* critical clique graph *$\mathcal{C}$ is a graph $(\mathcal{K}, E_{\mathcal{C}})$ with*

$$\{K_i, K_j\} \in E_{\mathcal{C}} \iff \forall u \in K_i, v \in K_j : \{u, v\} \in E.$$

That is, the critical clique graph has the critical cliques as nodes, and two nodes are connected iff the corresponding critical cliques together form a larger clique.

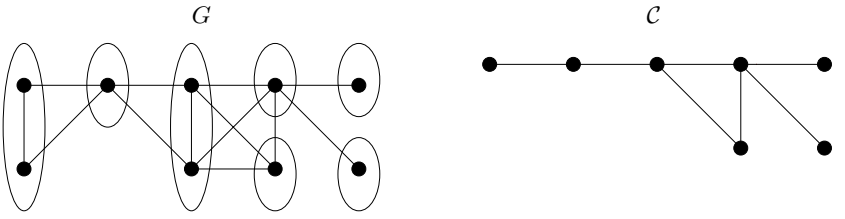


Fig. 1. A graph G and its critical clique graph $\mathcal{C}$. *Ovals* denote the critical cliques of G.

See Figure 1 for an example of a graph G and its critical clique graph. Note that we use the term *nodes* for the vertices in $\mathcal{C}$. Moreover, we use $K(v)$ to denote the critical clique containing vertex v and use $V(K)$ to denote the set of vertices contained in a critical clique $K \in \mathcal{K}$.

Parameterized complexity is a two-dimensional framework for studying the computational complexity of problems [11,20]. One dimension is the input size n (as in classical complexity theory), and the other one is the *parameter* k (usually a positive integer). A problem is called *fixed-parameter tractable* (fpt) if it can be solved in $f(k) \cdot n^{O(1)}$ time, where f is a computable function only depending on k. This means that when solving a combinatorial problem that is fpt, the combinatorial explosion can be confined to the parameter.

A core tool in the development of fixed-parameter algorithms is polynomial-time preprocessing by *data reduction*. Here, the goal is for a given problem instance x with parameter k to transform it into a new instance x' with parameter k' such that the size of x' is upper-bounded by some function only depending on k, the instance (x, k) is a yes-instance iff (x', k') is a yes-instance, and $k' \leq k$. The reduced instance, which must be computable in polynomial time, is called a *problem kernel*, and the whole process is called *reduction to a problem kernel* or simply *kernelization*.

3 Data Reduction Leading to a $6k$-Vertex Kernel

Based on the concept of critical cliques, we present a polynomial-time kernelization algorithm for Cluster Editing which leads to a problem kernel consisting of at most $6k$ vertices. In this way, we confirm the conjecture by Fellows that Cluster Editing admits a $6k$-vertex problem kernel [12,13]. Our data reduction rules are inspired by the "crown reduction rule" introduced in [12,13]. The main innovation from our side is the novel use of the critical clique concept.

The basic idea behind introducing critical cliques is the following: suppose that the input graph $G = (V, E)$ has a solution with at most k edge modifications. Then, at most $2k$ vertices are "affected" by these edge modifications, that is, they are endpoints of edges added or deleted. Thus, in order to give a size bound on V depending only on k, it remains to upper-bound the size of the "unaffected" vertices. The central observation is that, in the cluster graph obtained after making the at most k edge modifications, the unaffected vertices contained in one clique must form a critical clique in the original graph G. By this observation,

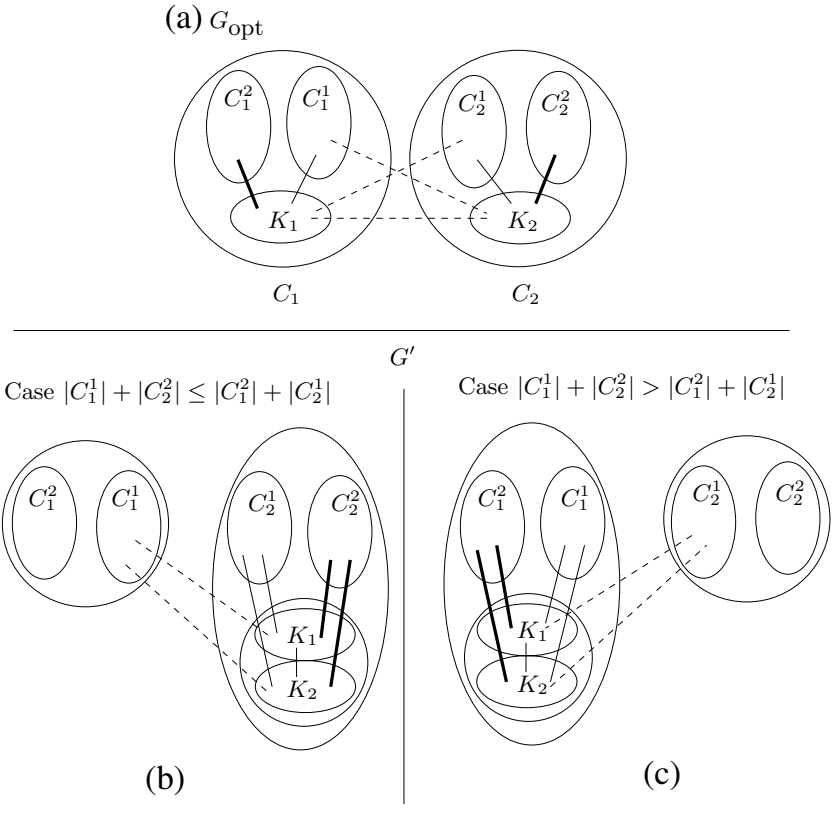


Fig. 2. An illustration of the proof of Lemma 1. The dashed lines indicate edge deletions, the thick lines indicate edge insertions, and the thin lines represent the edges unaffected.

it seems easier to derive data reduction rules working for the critical cliques and the critical clique graph than to derive rules directly working on the input graph.

The following two lemmas show the connection between critical cliques and optimal solution sets for Cluster Editing:

Lemma 1. *There is no optimal solution set E_{opt} for* Cluster Editing *on G that "splits" a critical clique of G. That is, every critical clique is entirely contained in one clique in $G_{\text{opt}} = (V, E \triangle E_{\text{opt}})$ for every optimal solution set E_{opt}.*

Proof. We show this lemma by contradiction. Suppose that we have an optimal solution set E_{opt} for G that splits a critical clique K of G, that is, there are at least two cliques C_1 and C_2 in G_{opt} with $K_1 := C_1 \cap K \neq \emptyset$ and $K_2 := C_2 \cap K \neq \emptyset$. Furthermore, we partition $C_1 \setminus K_1$ (and $C_2 \setminus K_2$) into two subsets, namely, set C_1^1 (and C_2^1) containing the vertices from $C_1 \setminus K_1$ (and $C_2 \setminus K_2$) which are neighbors of the vertices in K in G and $C_1^2 := (C_1 \setminus K_1) \setminus C_1^1$ (and $C_2^2 := (C_2 \setminus K_2) \setminus C_2^1$). See part (a) in Figure 2 for an illustration. Clearly, E_{opt} deletes

the edges E_{K_1,K_2} between K_1 and K_2. In addition, E_{opt} has to delete the edges between K_1 and C_2^1 and the edges between K_2 and C_1^1, and, moreover, E_{opt} has to insert the edges between K_1 and C_1^2 and the edges between K_2 and C_2^2. In summary, E_{opt} needs at least

$$|E_{K_1,K_2}| + |K_1| \cdot |C_2^1| + |K_2| \cdot |C_1^1| + |K_1| \cdot |C_1^2| + |K_2| \cdot |C_2^2|$$

edge modifications.

In what follows, we construct solution sets that are smaller than E_{opt}, giving a contradiction. Consider the following two cases: $|C_1^1| + |C_2^2| \leq |C_1^2| + |C_2^1|$ and $|C_1^1| + |C_2^2| > |C_1^2| + |C_2^1|$. In the first case, we remove K_1 from C_1 and merge it to C_2. Herein, we need the following edge modifications: deleting the edges between $K_1 \cup K_2$ and C_1^1 and inserting the edges between $K_1 \cup K_2$ and C_2^2. Here, we need $|K_1| \cdot |C_1^1| + |K_2| \cdot |C_1^1| + |K_1| \cdot |C_2^2| + |K_2| \cdot |C_2^2|$ edge modifications. See part (b) in Figure 2 for an illustration. In the second case, we remove K_2 from C_2 and merge it to C_1. Herein, we need the following edge modifications: deleting the edges between $K_1 \cup K_2$ and C_2^1 and inserting the edges between $K_1 \cup K_2$ and C_1^2. Here, we need $|K_1| \cdot |C_2^1| + |K_2| \cdot |C_2^1| + |K_1| \cdot |C_1^2| + |K_2| \cdot |C_1^2|$ edge modifications. See part (c) in Figure 2 for an illustration. Comparing the edge modifications needed in these two cases with E_{opt}, we can each time observe that E_{opt} contains some additional edges, namely E_{K_1,K_2}. This means that, in both cases $|C_1^1| + |C_2^2| \leq |C_1^2| + |C_2^1|$ and $|C_1^1| + |C_2^2| > |C_1^2| + |C_2^1|$, we can find a solution set E' that causes less edge modifications than E_{opt}, a contradiction to the optimality of E_{opt}. □

Lemma 2. *Let K be a critical clique with $|V(K)| \geq |\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K')|$. Then, there exists an optimal solution set E_{opt} such that, for the clique C in $G_{\text{opt}} = (V, E \triangle E_{\text{opt}})$ containing K, it holds $C \subseteq \bigcup_{K' \in N_{\mathcal{C}}[K]} V(K')$.*

Proof. By Lemma 1, the critical clique K is contained entirely in a clique C in $G_{\text{opt}} = (V, E \triangle E_{\text{opt}})$ for any optimal solution set E_{opt}. Suppose that, for an optimal solution set E_{opt}, C contains some vertices that are neither from $V(K)$ nor adjacent to a vertex in $V(K)$, that is, $D := C \setminus (\bigcup_{K' \in N_{\mathcal{C}}[K]} V(K')) \neq \emptyset$. Then, E_{opt} has inserted at least $|D| \cdot |V(K)|$ many edges into G to obtain the clique C. Then, we can easily construct a new solution set E' which leaves a cluster graph G' having a clique C' with $C' = C \setminus D$. That is, instead of inserting edges between $V(K)$ and D, the solution set E' deletes the edges between $C \cap (\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K'))$ and D. Since $|V(K)| \geq |\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K')|$, E_{opt} cannot be better than E' and, hence, E' is also an optimal solution. Thus, in the cluster graph that results from performing the modifications corresponding to E', the clique C containing K satisfies $C \subseteq \bigcup_{K' \in N_{\mathcal{C}}[K]} V(K')$. This completes the proof. □

The following data reduction rules work on both the input graph G and its critical clique graph $\mathcal{C}$. Note that the critical clique graph can be easily constructed in $O(m + n)$ time [17].

Rule 1: Remove all isolated critical cliques K from $\mathcal{C}$ and remove $V(K)$ from G.

Lemma 3. *Rule 1 is correct and can be carried out in $O(m+n)$ time.*

Rule 2: If, for a node K in $\mathcal{C}$, it holds $|V(K)| > |\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K')| + |\bigcup_{K' \in N^2_{\mathcal{C}}(K)} V(K')|$, then remove nodes K and $N_{\mathcal{C}}(K)$ from $\mathcal{C}$ and remove the vertices in $\bigcup_{K' \in N_{\mathcal{C}}[K]} V(K')$ from G. Accordingly, decrease parameter k by the sum of the number of edges needed to transform subgraph $G[\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K')]$ into a complete graph and the number of edges in G between the vertices in $\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K')$ and the vertices in $\bigcup_{K' \in N^2_{\mathcal{C}}(K)} V(K')$. If $k < 0$, then the given instance has no solution.

Lemma 4. *Rule 2 is correct and can be carried out in $O(n^3)$ time.*

Proof. Let K denote a critical clique in G that satisfies the precondition of Rule 2. Let $A := \{K' \in N_{\mathcal{C}}(K)\}$ and $B := \{K' \in N^2_{\mathcal{C}}(K)\}$. Let $V(A) := \bigcup_{K' \in A} V(K')$ and $V(B) := \bigcup_{K' \in B} V(K')$. From the precondition of Rule 2, we know that $|V(K)| > |V(A)| + |V(B)|$. We show the correctness of Rule 2 by proving the claim that there exists an optimal solution set leaving a cluster graph where there is a clique having exactly the vertex set $V(K) \cup V(A)$.

From Lemmas 1 and 2, we know that there is an optimal solution set E_{opt} such that K is contained entirely in a clique C in $G_{\text{opt}} = (V, E \triangle E_{\text{opt}})$ and clique C contains only vertices from $V(K) \cup V(A)$, that is, $V(K) \subseteq C \subseteq V(K) \cup V(A)$. We show the claim by contradiction. Suppose that $C \subsetneq V(K) \cup V(A)$. By Lemma 1, there is a non-empty subset A_1 of A whose critical cliques are not in C. Let $A_2 := A \setminus A_1$. Moreover, let $E_{A_2,B}$ denote the edges between $V(A_2)$ and $V(B)$ and E_{A_1,A_2} denote the edges between $V(A_1)$ and $V(A_2)$. Clearly, E_{opt} comprises $E_{A_2,B}$ and E_{A_1,A_2}. Moreover, E_{opt} causes the insertion of a set E_{A_2} of edges to transform $G[V(A_2)]$ into a complete graph and causes the deletion of a set E_{K,A_1} of edges between K and A_1. This means that E_{opt} needs at least

$$|E_{A_1,A_2}|+|E_{A_2,B}|+|E_{A_2}|+|E_{K,A_1}| = |E_{A_1,A_2}|+|E_{A_2,B}|+|E_{A_2}|+|V(K)|\cdot|V(A_1)|$$

edge modifications to obtain clique C.

Now, we construct a solution set that is smaller than E_{opt}, giving a contradiction. Consider the solution set E' that leaves a cluster graph G' where K and all critical cliques in A form a clique C' and the vertices in $V \setminus (V(K) \cup V(A))$ are in the same cliques as in G_{opt}. To obtain clique C', the solution set E' contains also the edges in E_{A_2} and the edges in $E_{A_2,B}$. In addition, E' causes the insertion of all possible edges between the vertices in $V(A_1)$, the insertion of all possible edges between $V(A_1)$ and $V(A_2)$, and the deletion of the edges between $V(A_1)$ and $V(B)$. However, these additional edge modifications together amount to at most $|V(A_1)| \cdot (|V(A)| + |V(B)|)$. To create other cliques which do not contain vertices from $V(K) \cup V(A)$, the set E' causes at most as many edge modifications as E_{opt}. From the precondition of Rule 2 that $|V(K)| > |V(A)| + |V(B)|$,

we know that even if $E_{A_1,A_2} = \emptyset$, E_{opt} needs more edge modifications than E', which contradicts the optimality of E_{opt}. This completes the proof of the correctness of Rule 2.

The running time of Rule 2 is easy to prove: The construction of $\mathcal{C}$ is doable in $O(m+n)$ time [17]. To decide whether Rule 2 is applicable, we need to iterate over all critical cliques and, for each critical clique K, we need to compute the sizes of $\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K')$ and $\bigcup_{K' \in N^2_{\mathcal{C}}(K)} V(K')$. By applying a breadth-first search, these two set sizes for a fixed critical clique can be computed in $O(n)$ time. Thus, we can decide the applicability of Rule 2 in $O(n^2)$ time. Moreover, since every application of Rule 2 removes some vertices from G, it can be applied at most n times. The overall running time follows. □

An instance to which none of the above two reduction rules applies is called *reduced* with respect to these rules. The proof of the following theorem works in analogy to the one of Theorem 3 showing the $24k$-vertex problem kernel in [13].

Theorem 1. *If a reduced graph for* Cluster Editing *has more than* $6k$ *vertices, then it has no solution with at most* k *edge modifications.*

4 Data Reduction Leading to a $4k$-Vertex Kernel

Here, we show that the size bound for the number of vertices of the problem kernel for Cluster Editing can be improved from $6k$ to $4k$. In the proof of Theorem 3 in [13], the size of the set V_2 of the unaffected vertices is bounded by a function of the size of the set V_1 of the affected vertices. Since $|V_1| \leq 2k$ and each affected vertices could be counted twice, we have then the size bound $4k$ for V_2. In the following, we present two new data reduction rules, Rules 3 and 4, which, combined with Rule 1 in Section 3, enable us to show that $|V_2| \leq 2k$. Note that we achieve this smaller number of kernel vertices at the cost of an additional factor of $O(m)$ in the running time.

> **Rule 3:** Let K denote a critical clique in the critical clique graph $\mathcal{C}$ with $|V(K)| \geq |\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K')|$. If, for a critical clique K' in $N_{\mathcal{C}}(K)$, it holds $E_{K',N^2_{\mathcal{C}}(K)} \neq \emptyset$ and $|V(K)| \cdot |V(K')| \geq |E_{K',N_{\mathcal{C}}(K)}| + |E_{K',N^2_{\mathcal{C}}(K)}|$, where $E_{K',N_{\mathcal{C}}(K)}$ denotes the set of edges needed to connect the vertices in $V(K')$ to the vertices in all other critical cliques in $N_{\mathcal{C}}(K)$ and $E_{K',N^2_{\mathcal{C}}(K)}$ denotes the set of edges between $V(K')$ and the vertices in the critical cliques in $N^2_{\mathcal{C}}(K)$, then we remove all edges in $E_{K',N^2_{\mathcal{C}}(K)}$ and decrease the parameter k accordingly. If $k < 0$, then the given instance has no solution.

Lemma 5. *Rule 3 is correct and can be carried out in* $O(nm^2)$ *time.*

Proof. Let K be a critical clique with $|V(K)| \geq |\bigcup_{K' \in N_{\mathcal{C}}(K)} |V(K')|$. Suppose that there is a critical clique K' in $N_{\mathcal{C}}(K)$ for which the precondition of Rule 3 holds. By Lemma 1, an optimal solution splits neither K nor K', that is, every

optimal solution either deletes all edges between $V(K)$ and $V(K')$ or keeps all of them. In the first case, any optimal solution needs to delete $|V(K)| \cdot |V(K')|$ edges to separate K and K'. In the second case, we know by Lemma 2 that there is an optimal solution E_{opt} such that the clique C in $G_{\text{opt}} = (V, E \triangle E_{\text{opt}})$ containing $V(K) \cup V(K')$ has no vertices from $V \setminus (\bigcup_{K' \in N_{\mathcal{C}}[K]} V(K'))$. This means that E_{opt} has to remove the edges in $E_{K', N^2_{\mathcal{C}}(K)}$. In addition, E_{opt} has to insert the edges between $V(K')$ and the vertices in $(C \cap (\bigcup_{K'' \in N_{\mathcal{C}}(K)} V(K''))) \setminus V(K')$. Obviously, these additional edge insertions amount to at most $|E_{K', N_{\mathcal{C}}(K)}|$. By the precondition of Rule 3, that is, $|V(K)| \cdot |V(K')| \geq |E_{K', N_{\mathcal{C}}(K)}| + |E_{K', N^2_{\mathcal{C}}(K)}|$, an optimal solution in the second case will never cause more edge modifications than in first case. Thus, we can safely remove the edges in $E_{K', N^2_{\mathcal{C}}(K)}$ and Rule 3 is correct.

Given a critical clique graph $\mathcal{C}$ and a fixed critical clique K, we can compute, for all critical cliques $K' \in N_{\mathcal{C}}(K)$, the sizes of the two edge sets $E_{K', N_{\mathcal{C}}(K)}$ and $E_{K', N^2_{\mathcal{C}}(K)}$ as defined in Rule 3 in $O(m)$ time. To decide whether Rule 3 can be applied, one iterates over all critical cliques K and computes $E_{K', N_{\mathcal{C}}(K)}$ and $E_{K', N^2_{\mathcal{C}}(K)}$ for all critical cliques $K' \in N_{\mathcal{C}}(K)$. Thus, the applicability of Rule 3 can be decided in $O(nm)$ time. Clearly, Rule 3 can be applied at most m times; this gives us an overall running time of $O(nm^2)$. □

> **Rule 4:** Let K denote a critical clique with $|V(K)| \geq |\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K')|$ and $N^2_{\mathcal{C}}(K) = \emptyset$. Then, we remove the critical cliques in $N_{\mathcal{C}}[K]$ from $\mathcal{C}$ and their corresponding vertices from G. We decrease the parameter k by the number of the missing edges between the vertices in $\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K')$. If $k < 0$, then the given instance has no solution.

Lemma 6. *Rule 4 is correct and can be carried out in $O(n^3)$ time.*

Based on these two data reduction rules, we achieve a problem kernel of $4k$ vertices for Cluster Editing.

Theorem 2. *If a graph G that is reduced with respect to Rules 1, 3, and 4 has more than $4k$ vertices, then there is no solution for* Cluster Editing *with at most k edge modifications.*

Proof. Suppose that there is a solution set E_{opt} of the reduced instance with at most k edge modifications that leads to a cluster graph with ℓ cliques, $C_1, C_2, \ldots, C_\ell$. We partition V into two sets, namely set V_1 of the affected vertices and set V_2 of the unaffected vertices. Obviously, $|V_1| \leq 2k$. We know that in each of the ℓ cliques the unaffected vertices must form exactly one critical clique in G. Let $K_1, K_2, \ldots, K_\ell$ denote the critical cliques formed by these unaffected vertices. These critical cliques can be divided into two sets, $\mathcal{K}_1$ containing the critical cliques K for which $|V(K)| < |\bigcup_{K' \in N_{\mathcal{C}}(K)} V(K')|$ holds, and $\mathcal{K}_2 := \{K_1, K_2, \ldots, K_\ell\} \setminus \mathcal{K}_1$.

First, we consider a critical clique K_i from $\mathcal{K}_1$. Since G is reduced with respect to Rule 1, $\bigcup_{K' \in N_{\mathcal{C}}(K_i)} V(K') \neq \emptyset$ and all vertices in $\bigcup_{K' \in N_{\mathcal{C}}(K_i)} V(K')$ must be

affected vertices. Clearly, the size of $\bigcup_{K' \in N_{\mathcal{C}}(K_i)} V(K')$ can be bounded from above by $2|E_i^+| + |E_i^-|$, where E_i^+ is the set of the edges inserted by E_{opt} with both their endpoints being in C_i, and E_i^- is the set of the edges deleted by E_{opt} with exactly one of their endpoints being in C_i. Hence, $|V(K_i)| < 2|E_i^+| + |E_i^-|$.

Second, we consider a critical clique K_i from $\mathcal{K}_2$. Since G is reduced with respect to Rules 1 and 4, we know that $N_{\mathcal{C}}(K_i) \neq \emptyset$ and $N_{\mathcal{C}}^2(K_i) \neq \emptyset$. Moreover, since G is reduced with respect to Rule 3, there exists a critical cliques K' in $N_{\mathcal{C}}(K_i)$ for which it holds that $E_{K',N_{\mathcal{C}}^2(K_i)} \neq \emptyset$ and $|V(K_i)| \cdot |V(K')| < |E_{K',N_{\mathcal{C}}(K_i)}| + |E_{K',N_{\mathcal{C}}^2(K_i)}|$, where $E_{K',N_{\mathcal{C}}(K_i)}$ denotes the set of edges needed to connect $V(K')$ to the vertices in the critical cliques in $N_{\mathcal{C}}(K_i) \setminus \{K'\}$ and $E_{K',N_{\mathcal{C}}^2(K_i)}$ denotes the set of edges between $V(K')$ and the vertices in the critical cliques in $N_{\mathcal{C}}^2(K_i)$. Then we have

$$|V(K_i)| < (|E_{K',N_{\mathcal{C}}(K_i)}| + |E_{K',N_{\mathcal{C}}^2(K_i)}|)/|V(K')| \leq |E_i^+| + |E_i^-|$$

where E_i^+ and E_i^- are defined as above.

To give an upper bound of $|V_2|$, we use E^+ to denote the set of edges inserted by E_{opt} and E^- to denote the set of edges deleted by E_{opt}. We have

$$\begin{aligned} |V_2| &= \sum_{i=1}^{\ell} |V(K_i)| \overset{(*)}{\leq} \sum_{i=1}^{\ell} (2|E_i^+| + |E_i^-|) \overset{(**)}{=} 2|E^+| + \sum_{i=1}^{\ell} |E_i^-| \\ &\overset{(***)}{=} 2|E^+| + 2|E^-| = 2k. \end{aligned}$$

The inequality $(*)$ follows from the analysis in the above two cases. The fact that E_i^+ and E_j^+ are disjoint for $i \neq j$ gives the equality $(**)$. Since an edge between two cliques C_i and C_j that is deleted by E_{opt} has to be counted twice, once for E_i^- and once for E_j^-, we have the equality $(***)$. Together with $|V_1| \leq 2k$, we thus arrive at the claimed size bound. □

5 Cluster Editing with a Fixed Number of Cliques

In this section, we consider the Cluster Editing$[d]$ problem. The first observation here is that the data reduction rules from Sections 3 and 4 do not work for Cluster Editing$[d]$. The reason is that Lemma 1 is not true if the number of cliques is fixed: in order to get a prescribed number of cliques, one critical clique might be split into several cliques by an optimal solution. However, based on the critical clique concept, we can show that Clique Editing$[d]$ admits a problem kernel with at most $(d+2)k + d$ vertices.

The kernelization is based on a simple data reduction rule.

> **Rule:** If a critical clique K contains at least $k+2$ vertices, then remove the critical cliques in $N_{\mathcal{C}}[K]$ from the critical clique graph $\mathcal{C}$ and remove the vertices in $\bigcup_{K' \in N_{\mathcal{C}}[K]} V(K')$ from the input graph G. Accordingly, decrease the parameter k by the number of the edges needed to transform

the subgraph $G[\bigcup_{K' \in N_{\mathcal{C}}[K]} V(K')]$ into a complete graph. If $k < 0$, then the given instance has no solution.

Lemma 7. *The above data reduction rule is correct and can be executed in $O(m + n)$ time.*

Next, we show a problem kernel for Cluster Editing$[d]$.

Theorem 3. *If a graph G that is reduced with respect to the above data reduction rule has more than $(d + 2) \cdot k + d$ vertices, then it has no solution for* Cluster Editing$[d]$ *with at most k edge modifications allowed.*

Proof. As in the proofs of Theorem 2, we partition the vertices into two sets. The set V_1 of affected vertices has a size bounded from above by $2k$. It remains to upper-bound the size of the set V_2 of unaffected vertices. Since in Cluster Editing$[d]$ the goal graph has exactly d cliques, we can have at most d unaffected critical cliques. Since the graph G is reduced, the maximal size of a critical clique is upper-bounded by $k + 1$. Thus, $|V_2| \leq d \cdot (k + 1)$ and $|V| \leq (d + 2) \cdot k + d$. □

Based on Theorem 3 and the fact that a problem is fixed-parameter tractable iff it admits a problem kernel [11, 20], we get the following corollary.

Corollary 1. *For fixed constant d,* Cluster Editing$[d]$ *is fixed-parameter tractable with the number k of allowed edge modifications as parameter.*

6 Open Problems and Future Research

In this paper, we have presented several polynomial-time kernelization algorithms for Cluster Editing and Cluster Editing$[d]$. We propose the following directions for future research.

- Can the running time of the data reduction rules be improved to $O(n + m)$?
- Can we apply the critical clique concept to derive a problem kernel for the more general Correlation Clustering problem [3]?
- Can the technique from [6] be applied to show a lower bound on the problem kernel size for Cluster Editing?

Acknowledgment. I thank Rolf Niedermeier (Universität Jena) for inspiring discussions and helpful comments improving the presentation.

References

1. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. In: Proc. 37th ACM STOC, pp. 684–693. ACM Press, New York (2005)
2. Alber, J., Fellows, M.R., Niedermeier, R.: Polynomial time data reduction for Dominating Set. Journal of the ACM 51(3), 363–384 (2004)

3. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning 56(1), 89–113 (2004)
4. Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering gene expression patterns. Journal of Computational Biology 6(3/4), 281–297 (1999)
5. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. Journal of Computer and System Sciences 71(3), 360–383 (2005)
6. Chen, J., Fernau, H., Kanj, I.A., Xia, G.: Parametric duality and kernelization: Lower bounds and upper bounds on kernel size. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 269–280. Springer, Heidelberg (2005)
7. Chen, Z.-Z., Jiang, T., Lin, G.: Computing phylogenetic roots with bounded degrees and errors. SIAM Journal on Computing 32(4), 864–879 (2003)
8. Dehne, F., Langston, M.A., Luo, X., Pitre, S., Shaw, P., Zhang, Y.: The Cluster Editing problem: Implementations and experiments. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 13–24. Springer, Heidelberg (2006)
9. Dom, M., Guo, J., Hüffner, F., Niedermeier, R.: Extending the tractability border for closest leaf powers. In: Kratsch, D. (ed.) WG 2005. LNCS, vol. 3787, pp. 397–408. Springer, Heidelberg (2005)
10. Dom, M., Guo, J., Hüffner, F., Niedermeier, R.: Error compensation in leaf power problems. Algorithmica 44(4), 363–381 (2006)
11. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999)
12. Fellows, M.R.: The lost continent of polynomial time: Preprocessing and kernelization. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 276–277. Springer, Heidelberg (2006)
13. Fellows, M.R., Langston, M.A., Rosamond, F., Shaw, P.: Polynomial-time linear kernelization for Cluster Editing. Manuscript (2006)
14. Giotis, I., Guruswami, V.: Correlation clustering with a fixed number of clusters. In: Proc. 17th ACM-SIAM SODA, pp. 1167–1176. ACM Press, New York (2006)
15. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Graph-modeled data clustering: Exact algorithms for clique generation. Theory of Computing Systems 38(4), 373–392 (2005)
16. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. ACM SIGACT News 38(1), 31–45 (2007)
17. Hsu, W., Ma, T.: Substitution decomposition on chordal graphs and applications. In: Hsu, W.-L., Lee, R.C.T. (eds.) ISA 1991. LNCS, vol. 557, pp. 52–60. Springer, Heidelberg (1991)
18. Křivánek, M., Morávek, J.: NP-hard problems in hierarchical-tree clustering. Acta Informatica 23(3), 311–323 (1986)
19. Lin, G., Kearney, P.E., Jiang, T.: Phylogenetic k-root and Steiner k-root. In: Lee, D.T., Teng, S.-H. (eds.) ISAAC 2000. LNCS, vol. 1969, pp. 539–551. Springer, Heidelberg (2000)
20. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, Oxford (2006)
21. Protti, F., da Silva, M.D., Szwarcfiter, J.L.: Applying modular decomposition to parameterized bicluster editing. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 1–12. Springer, Heidelberg (2006)
22. Shamir, R., Sharan, R., Tsur, D.: Cluster graph modification problems. Discrete Applied Mathematics 144, 173–182 (2004)

CR-PRECIS: A Deterministic Summary Structure for Update Data Streams

Sumit Ganguly[1] and Anirban Majumder[2,⋆]

[1] Indian Institute of Technology, Kanpur
[2] Lucent Technologies, Bangalore

Abstract. We present deterministic sub-linear space algorithms for problems over update data streams, including, estimating frequencies of items and ranges, finding approximate frequent items and approximate ϕ-quantiles, estimating inner-products, constructing near-optimal B-bucket histograms and estimating entropy. We also present improved lower bound results for several problems over update data streams.

1 Introduction

The data streaming model [2,26] presents a computational model for a variety of monitoring applications, for example, network monitoring, sensor networks, etc., where data arrives rapidly and continuously and has to be processed in an online fashion using sub-linear space. Some examples of popular data stream queries include, estimating the frequency of items (point queries) and ranges (range-sum queries), finding approximate frequent items, approximate quantiles and approximate hierarchical heavy hitters, estimating inner-product, constructing approximately optimal B-bucket histograms, estimating entropy, etc.. A data stream in the *general update model* is viewed as a sequence of records of the form (pos, i, δ), where, *pos* is the current sequence position, i is an item index from the domain $[n] = \{1, 2, \ldots, n\}$ and $\delta \in \mathbb{Z}$. δ denotes the increment or decrement in the *frequency* of i, that is, $\delta > 0$ signifies that i has been inserted δ times and $\delta < 0$ signifies $|\delta|$ deletions of i. The frequency f_i of $i \in [n]$ is defined as the sum of the δ's over all the records in the stream that contain i, that is, $f_i = \sum_{(\ pos, i, \delta) \in stream} \delta$. A special case of this model is the *strict update model*, where, deletions always correspond to prior insertions, that is, item frequencies defined by every prefix of the input stream is non-negative. The *insert-only* model refers to data streams that have no deletions.

Randomized algorithms dominate the landscape of sub-linear space algorithms for problems over update streams. There are no deterministic sub-linear space algorithms known for a variety of basic problems over update streams, including, estimating the frequency of items and ranges, finding approximate frequent items and approximate ϕ-quantiles, finding approximate hierarchical heavy hitters, constructing approximately optimal B-bucket histograms, estimating inner-products, estimating entropy, etc.. Deterministic algorithms are often indispensable, for example, in a marketing scenario where frequent items correspond to subsidized

⋆ Work done while at IIT Kanpur.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 48–59, 2007.

customers, a false negative would correspond to a missed frequent customer, and conversely, in a scenario where frequent items correspond to punishable misuse [21], a false positive results in an innocent victim.

We now review a data structure introduced by Gasieniec and Muthukrishnan [26] (page 31) that we use later. We refer to this structure as the CR-PRECIS structure, since the Chinese Remainder theorem plays a crucial role in the analysis. The structure is parameterized by a height k and width t. Choose t consecutive prime numbers $k \leq p_1 < p_2 < \ldots < p_t$ and keep a collection of t tables T_j, for $j = 1, \ldots, t$, where, T_j has p_j integer counters, numbered from $0, 1, \ldots, p_j - 1$. Each stream update of the form (pos, i, δ) is processed as follows.

$$\textbf{for } j := 1 \textbf{ to } t \textbf{ do } \{\ T_j[i \mod p_j] := T_j[i \mod p_j] + \delta\ \}$$

Lemma 1 presents the space requirement of a CR-PRECIS structure and is implicit in [26] (pp. 31) and is proved by a direct application of the prime number theorem [27]. Let $L_1 = \sum_i |f_i|$.

Lemma 1. *The space requirement of a* CR-PRECIS *structure with height parameter* $k \geq 12$ *and width parameter* $t \geq 1$ *is* $O(t(t + \frac{k}{\ln k}) \log(t + \frac{k}{\ln k})(\log L_1))$ *bits. The time required to process a stream update is* $O(t)$ *arithmetic operations.* □

[26] (pp. 31) uses the CR-PRECIS structure to present a k-set structure [15] using space $O(k^2(\log k)(\log L_1))$ bits. k-set structures using space $O(k(\log N)(\log N + \log L_1))$ bits are presented in [15].

Contributions. We present deterministic, sub-linear space algorithms for each of the problems mentioned above in the model of update streams using the CR-PRECIS structure. We also present improved space lower bounds for some problems over update streams, namely, (a) the problem of estimating frequencies with accuracy ϕ over strict update streams is shown to require $\Omega(\phi^{-1}(\log m) \log(\phi n))$ space (previous bound was $\Omega(\phi^{-1} \log(\phi n))$ [4]), and, (b) all the above problems except for the problems of estimating frequencies of items and range-sums, are shown to require $\Omega(n)$ space in the general update streaming model.

2 Review

In this section, we review basic problems over data streams. For strict update streams, let $m = \sum_i f_i$ and for general update streams, $L_1 = \sum_i |f_i|$.

The *point query* problem with parameter ϕ is the following: given $i \in \mathcal{D}$, obtain an estimate $\hat{f}_i$ such that $|\hat{f}_i - f_i| \leq \phi L_1$. For *insert-only streams*, the Misra-Gries algorithm [25], rediscovered and refined in [12,4,22], uses $\lceil \phi^{-1} \rceil \log m$ bits and satisfies $f_i \leq \hat{f}_i \leq f_i + \phi m$. The *Lossy Counting* algorithm [23] for insert-only streams returns $\hat{f}_i$ satisfying $f_i \leq \hat{f}_i \leq f_i + \phi m$ using $\lceil \phi^{-1} \rceil \log(\phi m) \log m$ bits. The *Sticky Sampling* algorithm [23] extends the *Counting Samples* algorithm [16] and returns $\hat{f}_i$ satisfying $f_i - \phi m \leq \hat{f}_i \leq f_i$ with probability $1 - \delta$ using space $O(\lceil \phi^{-1} \rceil \log(\delta^{-1}) \log m)$ bits. The COUNT-MIN sketch algorithm returns $\hat{f}_i$ that satisfies (a) $f_i \leq \hat{f}_i \leq f_i + \phi m$ with probability $1 - \delta$ for strict update streams, and, (b) $|\hat{f}_i - f_i| \leq \phi L_1$ using $O(\lceil \phi^{-1} \rceil (\log \delta^{-1}) \log L_1)$ bits. The

COUNTSKETCH algorithm [6] satisfies $|\hat{f}_i - f_i| \leq (\lceil \phi^{-1} \rceil F_2^{res}(\lceil \phi^{-1} \rceil))^{1/2} \leq \phi L_1$ with probability $1 - \delta$ using space $O(\lceil \phi^{-1} \rceil (\log \delta^{-1} \log L_1)$ bits, where, $F_2^{res}(s)$ is the sum of the squares of all but the top-s frequencies in the stream. [4] shows that any algorithm satisfying $|\hat{f}_i - f_i| \leq \phi m$ must use $\Omega(\lceil \phi^{-1} \rceil \log \phi n)$ bits.

Given a parameter $0 < \phi < 1$, an item i is said to be *ϕ-frequent* if $|f_i| \geq \phi L_1$. [11,22] show that finding all and only frequent items requires $\Omega(n)$ space. Therefore low-space algorithms find ϵ-approximate frequent items, where, $0 < \epsilon < 1$ is another parameter: return i such that $|f_i| \geq \phi L_1$ and do not return any i such that $|f_i| < (1 - \epsilon)\phi L_1$ [6,11,9,12,16,22,25,23,28]. Algorithms for finding frequent items with parameters ϕ and ϵ typically use point query estimators with parameter $\frac{\epsilon\phi}{2}$ and return all items i such that $\hat{f}_i > (1 - \frac{\epsilon}{2})\phi L_1$. A superset of ϵ-approximate frequent items is typically found using the technique of dyadic intervals [10,9], reviewed below.

A *dyadic interval* at level l is an interval of size 2^l from the family of intervals $\{[i2^l, (i+1)2^l - 1], 0 \leq i \leq \frac{n}{2^l} - 1\}$, for $0 \leq l \leq \log n$, assuming that n is a power of 2. The set of dyadic intervals at levels 0 through $\log n$ form a complete binary tree, whose root is the level $\log n$ dyadic interval $[0, n-1]$ and whose leaves are the singleton intervals. Each dyadic interval I_l with level $1 \leq l \leq \log n$ has two children that are dyadic intervals at levels $l-1$. If $I_l = [i2^l, (i+1)2^l - 1]$, for $0 \leq i \leq \frac{n}{2^l}$, then, the left child of I_l is $[2i\frac{n}{2^{l+1}}, (2i+1)\frac{n}{2^{l+1}} - 1]$ and the right child is $[(2i+1)\frac{n}{2^{l+1}}, (2i+2)\frac{n}{2^{l+1}} - 1]$. Given a stream, one can naturally extend the notion of item frequencies to dyadic interval frequencies. The frequency of a dyadic interval I_l at level l is the aggregate of the frequencies of the level 0 items that lie in that interval , that is, $f_{I_l} = \sum_{x \in I_l} f_x$. The efficient solution to a number of problems over *strict update* streams, including the problem of finding approximate frequent items, is facilitated by using summary structures for each dyadic level $l = 0, 1, \ldots, \lceil \log \phi n \rceil$ [10]. For the problem of ϵ-approximate ϕ-frequent items, we keep a point query estimator structure corresponding to accuracy parameter $\epsilon\phi$ for each dyadic level $l = 0, \ldots, \lceil \log(\phi n) \rceil$. An arrival over the stream of the form (pos, i, δ) is processed as follows: for each $l = 0, 1, \ldots, \lfloor \log \phi n \rfloor$, propagate the update $(\text{pos}, (i \ \% \ 2^l), \delta)$ to the structure at level l. Since, each item i belongs to a unique dyadic interval at each level l, the sum of the interval frequencies at level l is m. If an item i is frequent (i.e., $f_i \geq \phi m$), then for each $1 \leq l \leq \log n$, the unique dyadic interval I_l that contains i at level l has frequency at least f_i and is therefore also frequent at level l. To find ϵ-approximate ϕ-frequent items, we start by enumerating $O(\lceil \phi^{-1} \rceil)$ dyadic intervals at level $\lfloor \log \phi n \rfloor$. Only those candidate intervals are considered whose estimated frequency is at least $(1 - \frac{\epsilon}{2})\phi n$. We then consider the left and the right child of these candidate intervals, and repeat the procedure. In general, at level l, there are $O(\lceil \phi^{-1} \rceil)$ candidate intervals, and thus, the total number of intervals considered in the iterations is $O(\lceil \phi^{-1} \rceil \log(\phi n))$.

The *hierarchical heavy hitters* problem [8,13] is a useful generalization of the frequent items problem for domains that have a natural hierarchy (e.g., domain of IP addresses). Given a hierarchy, the frequency of a node X is defined as the sum of the frequencies of the leaf nodes (i.e., items) in the sub-tree rooted at

X. The definition of hierarchical heavy hitter node (*HHH*) is inductive: a leaf node x is an *HHH* node provided $f_x > \phi m$. An internal node is an *HHH* node provided that its frequency, after discounting the frequency of all its descendant *HHH* nodes, is at least ϕm. The problem is, (a) to find all nodes that are *HHH* nodes, and, (b) to not output any node whose frequency, after discounting the frequencies of descendant *HHH* nodes, is below $(1-\epsilon)\phi m$. This problem has been studied in [8,10,13,21]. As shown in [8], this problem can be solved by using a simple bottom-up traversal of the hierarchy, identifying the frequent items at each level, and then subtracting the estimates of the frequent items at a level from the estimated frequency of its parent [8]. Using COUNT-MIN sketch, the space complexity is $O((\epsilon\phi^2)^{-1}(\log((\delta\epsilon\phi^2)^{-1}\log n))(\log n)(\log m))$ bits. [21] presents an $\Omega(\phi^{-2})$ space lower bound for this problem, for fixed $\epsilon \leq 0.01$.

Given a *range* $[l, r]$ from the domain $\mathcal{D}$, the range frequency is defined as $f_{[l,r]} = \sum_{x=l}^{r} f_x$. The *range-sum query* problem with parameter ϕ is: return an estimate $\hat{f}_{[l,r]}$ such that $|\hat{f}_{[l,r]} - f_{[l,r]}| \leq \phi m$. The range-sum query problem can be solved by using the technique of dyadic intervals [19]. Any range can be uniquely decomposed into the disjoint union of at most $2\log n$ dyadic intervals of maximum size (for e.g., over the domain $\{0,\ldots,15\}$, the interval $[3,12] = [3,3]+[4,7]+[8,11]+[12,12]$). The technique is to keep a point query estimator corresponding to each dyadic level $l = 0, 1, \ldots, \log n - 1$. The range-sum query is estimated as the sum of the estimates of the frequencies of each of the constituent maximal dyadic intervals of the given range. Using COUNT-MIN sketch at each level, this can be accomplished using space $O(\phi^{-1}\log(\log(\delta^{-1}n))(\log n)(\log m))$ bits and with probability $1-\delta$ [10].

Given $0 \leq \phi \leq 1$ and $j = 1, 2, \ldots, \lceil\phi^{-1}\rceil$, an *$\epsilon$-approximate j^{th} ϕ-quantile* is an item a_j such that $(j\phi - \epsilon)m \leq \sum_{i=a_j}^{n-1} f_i \leq (j\phi+\epsilon)m$. The problem has been studied in [10,20,18,24]. For insert-only streams, [20] presents an algorithm requiring space $O((\log(\epsilon\phi)^{-1})\log(\epsilon\phi m))$. For strict update streams, the problem of finding approximate quantiles can be reduced to that of estimating range sums [18] as follows. For each $k = 1, 2, \ldots, \phi^{-1}$, a binary search is performed over the domain to find an item a_k such that the estimated range sum $\hat{f}_{[a_k,n-1]}$ lies between $(k\phi-\epsilon)m$ and $(k\phi+\epsilon)m$. [10] uses COUNT-MIN sketches and the above technique to find ϵ-approximate ϕ-quantiles with confidence $1-\delta$ using space $O(\epsilon\phi^{-1}\log^2 n((\log(\epsilon\phi\delta)^{-1}) + \log\log n))$.

A B-bucket histogram h divides the domain $\mathcal{D} = \{0, 1, \ldots, n-1\}$ into B non-overlapping intervals, say, $I_1, I_2, \ldots, I_B$ and for each interval I_j, chooses a value v_j. Then $h[0\ldots n-1]$ is the vector defined as $h_i = v_j$, where, I_j is the unique interval containing i. The cost of a B-bucket histogram h with respect to the frequency vector f is defined as $||f-h||$. Let h^{opt} denote an optimal B-bucket histogram satisfying $||f-h^{opt}|| = \min_{B\text{-bucket histogram } h} ||f-h||$. The problem is to find a B-bucket histogram $\hat{h}$ such that $||f-\hat{h}|| \leq (1+\epsilon)||f-h^{opt}||$. An algorithm for this problem is presented in a seminal paper [17] using space and time poly $(B, \frac{1}{\epsilon}, \log m, \log n)$ (w.r.t. L_2 distance $\|f-h\|_2$).

Given two streams R and S with item frequency vectors f and g respectively, the *inner product* $f \cdot g$ is defined as $\sum_{i\in\mathcal{D}} f_i \cdot g_i$. The problem is to return an

estimate $\hat{P}$ satisfying $|\hat{P} - f \cdot g| \leq \phi m_R m_S$. The problem finds applications in database query processing. The work in [1] presents a space lower bound of $s = \Omega(\phi^{-1})$ for this problem. Randomized algorithms [1,7,14] match the space lower bound, up to poly-logarithmic factors. The *entropy* of a data stream is defined as $H = \sum_{i \in \mathcal{D}} |f_i| \log \frac{L_1}{|f_i|}$. The problem is to return an ϵ-approximate estimate $\hat{H}$ satisfying $|\hat{H} - H| \leq \epsilon H$. For insert-only streams, [5] presents a randomized estimator that uses space $O(\epsilon^{-2}(\log \delta^{-1}) \log^3 m)$ bits and also shows an $\Omega(\epsilon^{-2}(\log(\epsilon^{-1}))^{-1})$ space lower bound for estimating entropy. [3] presents a randomized estimator for update streams using space $O(\epsilon^{-3} \log^5 m (\log \epsilon^{-1})(\log \delta^{-1}))$.

We note that sub-linear space deterministic algorithms over update streams are not known for any of the above-mentioned problems.

3 CR-PRECIS Structure for Update Streams

In this section, we use the CR-PRECIS structure to present algorithms for a family of problems over update streams.

An application of the Chinese Remainder Theorem. Consider a CR-PRECIS structure with height k and width t. Fix $x, y \in \{0, \ldots, n-1\}$ where $x \neq y$. Suppose x and y collide in the tables indexed by J, where, $J \subset \{1, 2, \ldots, t\}$. Then, $x \equiv y \mod p_j$, for each $j \in J$. By the Chinese Remainder theorem, $x \equiv y \mod \left(\prod_{j \in J} p_j\right)$. Therefore, $|J| < \log_k n$, otherwise, $\prod_{j \in J} p_j \geq k^{\log_k n} = n$, which is a contradiction, since, $x, y \in \{0, 1, \ldots, n-1\}$ and are distinct. Therefore, for any given $x, y \in \{0, 1, \ldots, n-1\}$ such that $x \neq y$,

$$|\{j \mid y \equiv x \bmod p_j \text{ and } 1 \leq j \leq t\}| \leq \log_k n - 1 \ . \tag{1}$$

3.1 Algorithms for Strict Update Streams

In this section, we use the CR-PRECIS structure to design algorithms over strict update streams.

Point Queries. Consider a CR-PRECIS structure with height k and width t. The frequency of $x \in \mathcal{D}$ is estimated as: $\hat{f}_x = \min_{j=1}^{t} T_j[x \bmod p_j]$. The accuracy guarantees are given by Lemma 2.

Lemma 2. *For* $0 \leq x \leq n-1$, $0 \leq \hat{f}_x - f_x \leq \frac{(\log_k n - 1)}{t}(m - f_x)$.

Proof. Clearly, $T_j[x \bmod p_j] \geq f_x$. Therefore, $\hat{f}_x \geq f_x$. Further,

$$t\hat{f}_x \leq \sum_{j=1}^{t} T_j[x \mod p_j] \quad = \quad tf_x \ + \ \sum_{j=1}^{t} \sum_{\substack{y \neq x \\ y \equiv x \bmod p_j}} f_y \ .$$

$$\text{Thus,}\quad t(\hat{f}_x - f_x) = \sum_{j=1}^{t} \sum_{\substack{y \neq x \\ y \equiv x \bmod p_j}} f_y = \sum_{y \neq x} \sum_{j: y \equiv x \bmod p_j} f_y$$

$$= \sum_{y \neq x} f_y |\{j : y \equiv x \bmod p_j\}| \leq (\log_k n - 1)(m - f_x), \quad \text{by (1)} \quad . \quad \square$$

If we let $k = \lceil \phi^{-1} \rceil$ and $t = \lceil \phi^{-1} \rceil \log_{\lceil \phi^{-1} \rceil} n$, then, the space requirement of the point query estimator is $O(\phi^{-2}(\log_{\lceil \phi^{-1} \rceil} n)^2(\log m))$ bits. A slightly improved guarantee that is often useful for the point query estimator is given by Lemma 3, where, $m^{res}(s)$ is the sum of all but the top-s frequencies [3,6].

Lemma 3. *Consider a* CR-PRECIS *structure with height s and width $2s \log_s n$. Then, for any $0 \leq x \leq n-1$, $0 \leq \hat{f}_x \leq \frac{m^{res}(s)}{s}$.*

Proof. Let $y_1, y_2, \ldots, y_s$ denote the items with the top-s frequencies in the stream (with ties broken arbitrarily). By (1), x conflicts with each $y_j \neq x$ in at most $\log_s n$ buckets. Hence, the total number of buckets at which x conflicts with any of the top-s frequent items is at most $s \log_s n$. Thus there are at least $t - s \log_s n$ tables where, x does not conflict with any of the top-s frequencies. Applying the proof of Lemma 2 to only these set of $t - s \log_s n \geq s \log_s n$ tables, the role of m is replaced by $m^{res}(s)$. $\square$

We obtain deterministic algorithms for estimating range-sums, finding approximate frequent items, finding approximate hierarchical heavy hitters and ϵ-approximate quantiles over strict update streams, by using the corresponding well-known reductions to point query estimators. The only change is that the use of randomized summary structures is replaced by a CR-PRECIS structure. Theorem 4 summarizes the space versus accuracy guarantees for these problems.

Theorem 4. *There exist deterministic algorithms over the strict update streaming model for the problems mentioned in Figure 1 using the space and per-update processing time depicted there.* $\square$

Estimating inner product. Let $m_R = \sum_{i \in \mathcal{D}} f_i$ and let $m_S = \sum_{i \in \mathcal{D}} g_i$. We maintain a CR-PRECIS structure for each of the streams R and S, that have the same height k, same width t and use the same prime numbers as the table sizes. For $j = 1, 2, \ldots, t$, let T_j and U_j respectively denote the tables maintained for streams R and S corresponding to the prime p_j respectively. The estimate $\hat{P}$ for the inner product is calculated as $\hat{P} = \min_{j=1}^{t} \sum_{b=1}^{p_j} T_j[b] U_j[b]$.

Lemma 5. $f \cdot g \leq \hat{P} \leq f \cdot g + \left(\frac{\log_k n - 1}{t}\right) m_R m_S$.

Proof. For $j = 1, \ldots, t$, $\sum_{b=0}^{p_j - 1} T_j[b] U_j[b] \geq \sum_{b=0}^{p_j - 1} \sum_{x \equiv b \bmod p_j} f_x g_x = f \cdot g$. Thus, $\hat{P} \geq f \cdot g$. Further,

$$t\hat{P} \leq \sum_{j=1}^{t}\sum_{b=1}^{p_j} T_j[b]U_j[b] = t(f\cdot g) + \sum_{j=1}^{t} \sum_{x\equiv y \bmod p_j, x\neq y} f_x g_y$$

$$= t(f\cdot g) + \sum_{x,y:x\neq y} f_x g_y \sum_{j:x\equiv y \bmod p_j} 1$$

$$\leq t(f\cdot g) + (\log_k n - 1)(m_R m_S - f\cdot g), \text{ by (1)}. \qquad \square$$

Problem	Space	Time
1. ϵ-approx. ϕ-frequent items ($\lambda = \lceil(\epsilon\phi)^{-1}\rceil$)	$O(\lambda^2(\log_\lambda n)(\log\lambda)$ $\log(\lambda^{-1}n)\ (\log m))$	$O(\lambda \log_\lambda n \log n)$
2. Range-sum: parameter ϕ, ($\rho = \lceil\phi^{-1}\rceil$)	$O(\rho^2(\log_\rho n)\ (\log\rho$ $+\log\log_\rho n))(\log m)\log n)$	$O(\rho(\log_\rho n)\ (\log n))$
3. ϵ-approx. ϕ-quantile ($\lambda = \lceil(\epsilon\phi)^{-1}\rceil$)	$O(\lambda^2(\log^5 n)(\log m)$ $(\log\log n + \log\lambda)^{-1})$	$O\big(\lambda(\log^2 n)$ $(\log\log n + \log\lambda)^{-1})\big)$
4. ϵ-approx. ϕ-hierarchical heavy hitters. h = height, ($\tau = (\epsilon\phi^2)^{-1}h$)	$O(\tau^2(\log_\tau n)^2$ $(\log\tau + \log\log n)\log m)$	$O(\tau(\log n)(\log_\tau n))$
5. ϵ-approx. B-bucket histogram	$O\big((\epsilon^{-2}B^2)(\log^3 n)$ $(\log^{-1}(\epsilon^{-1}B))(\log m)$	$O((\epsilon^{-1}B)(\log^2 n)$ $(\log^{-1}(\epsilon^{-1}B))$

Fig. 1. Space and time requirement for problems using CR-PRECIS technique

Estimating entropy. We use the CR-PRECIS structure to estimate the entropy H over a strict update stream. For parameters k and t to be fixed later, a CR-PRECIS structure of height $k \geq 2$ and width t is maintained. Also, let $0 < \epsilon < 1$ be a parameter. First, we use the point query estimator to find all items x such that $\hat{f}_x \geq \frac{m}{\epsilon t}$. The contribution of these items, called *dense* items, to the estimated entropy is given by $\hat{H}_d = \sum_{x:\hat{f}_x > \frac{m}{\epsilon t}} \hat{f}_x \log\frac{m}{\hat{f}_x}$. Next, we remove the estimated contribution of the dense items from the tables. To ensure that the residues of dense frequencies remain non-negative, the estimated frequency is altered. Since, $0 \leq \hat{f}_x - f_x \leq \frac{(m-f_x)}{t}$, $f_x \geq \hat{f}_x - \frac{m-\hat{f}_x}{t-1} = f'_x$ (say), the tables are modified as follows: $T_j[x \mod p_j] := T_j[x \mod p_j] - f'_x$, for each x s.t. $\hat{f}_x \geq \frac{m}{\epsilon t}$ and $j = 1,\ldots,t$. $\hat{H}_s$ estimates the contribution to H by the non-dense or *sparse* items: $\hat{H}_s = \text{avg}_{j=1}^{t} \sum_{1\leq b\leq p_j \text{ and } T_j[b]\leq \frac{m}{\epsilon^2 t}} T_j[b]\log\frac{m}{T_j[b]}$. The final estimate is returned as $\hat{H} = \hat{H}_d + \hat{H}_s$.

Analysis. The main part of the analysis concerns the accuracy of the estimation of H_s. Let H_d and H_s denote the (true) contributions to entropy due to dense and sparse items respectively, that is, $H_d = \sum_{x \text{ dense}} f_x \log\frac{m}{f_x}$ and $H_s = \sum_{x \text{ sparse}} f_x \log\frac{m}{f_x}$. A standard case analysis [3,5] (Case 1: $f_x \leq m/e$ and Case 2: $f_x > m/e$, where, $e = 2.71828..$) is used to show that $|\hat{H}_d - H_d| \leq \frac{2}{t}H_d$. Define $g_x = f_x$, if x is a sparse item and $g_x = f_x - f'_x$, if x is a dense item. Let

$m' = \sum_x g_x$ and let $H(g) = \sum_{x:f_x>0} g_x \log \frac{m}{g_x}$. Suppose x maps to a bucket b in table T_j. Then, $f_x \leq T_j[b]$ and

$$\sum_{b=1}^{p_j} T_j[b] \log \frac{m}{T_j[b]} \leq \sum_{b=1}^{p_j} \sum_{x \bmod p_j = b} g_x \log \frac{m}{T_j[b]} = H(g) \ .$$

Thus, $\hat{H}_s \leq H(g)$. Since, $H(g)$ may contain the contribution of the residues of the dense items, $H_s \leq H(g)$. The contribution of the residues of dense items to $H(g)$ is bounded as follows. Let x be a dense item. Define $h(a) = y \log \frac{m}{y}$. For $t > \frac{1}{\epsilon^2}$, and since, $f_x \geq \frac{m}{\epsilon t}$, $\quad h(\frac{g_x}{m}) \leq h\left(\frac{m-f_x}{mt}\right) \leq \epsilon h(f_x)$. Since, Therefore,

$$H(g) = \sum_x g(x) \log \frac{m}{g_x} = H_s + \sum_{x \text{ dense}} mh(g_x) \leq H_s + \sum_{x \text{ dense}} m\epsilon h(f_x) = H_s + \epsilon H_d.$$

Further, for $0 \leq x \leq n-1$, let $S_x = \sum_{j=1}^{t} (T_j[x \mod p_j] - g_x)$. Then

$$S_x = \sum_{j=1}^{t} \sum_{\substack{y \neq x \\ y \equiv x \bmod p_j}} g_y = \sum_{y \neq x} f_y |\{j : y \equiv x \bmod p_j\}| \leq (m' - g_x)(\log n - 1) \quad (2)$$

Define a bucket b in a table T_j to be dense if $T_j[b] > \frac{m}{\epsilon^2 t}$ and $c = \epsilon^2 t$. Then,

$$\begin{aligned}
t\hat{H}_s &= \sum_{j=1}^{t} \sum_{\substack{1 \leq b \leq p_j \\ b \text{ not dense}}} T_j[b] \log \frac{m}{T_j[b]} \\
&\geq \sum_{j=1}^{t} \sum_{\substack{1 \leq b \leq p_j \\ b \text{ not dense}}} \sum_{x : x \equiv b \bmod p_j \text{ and } g_x \geq 1} g_x \log c \\
&= tm' \log c - \sum_x g_x \log c \ |\{j : T_j[x \mod p_j] \text{ is dense } \}| \\
&= tm' \log c - \sum_x g_x (\log c) \ \lfloor S_x / (c^{-1} m - g_x) \rfloor \\
&\geq tm' \log c - \sum_x g_x c (1-\epsilon)^{-1} (\log c)(\log_k N) \text{ by (2) and since, } g_x \leq \epsilon c^{-1} m \\
&\geq tm' \log c - m' c (1-\epsilon)^{-1} (\log c)(\log_k N) \\
&\geq tm' \log c \left(1 - \epsilon^2 (1-\epsilon)^{-1} \log_k N\right) \qquad (3)
\end{aligned}$$

Lemma 6. *For each $0 < \varepsilon < 1$ and $\alpha > 1$, there exists a deterministic algorithm over strict update streams that returns $\hat{H}$ satisfying $\frac{H(1-\varepsilon)}{\alpha} \leq H \leq (1 + \frac{\varepsilon}{\sqrt{\log N}})H$ using space $O(\frac{\log^2 N}{\varepsilon^4} m^{\frac{2}{\alpha}} (\log m + \log \varepsilon^{-1})(\log m))$ bits.*

Proof. By earlier argument,

$$\begin{aligned}
\hat{H}_d + \hat{H}_s &\leq (1 + 2t^{-1}) H_d + H(g) \leq (1 + 2t^{-1}) H_d + \epsilon H_d + H_s \\
&\leq (1 + 2t^{-1} + \epsilon)(H_d + H_s) \ .
\end{aligned}$$

Further, since, $H(g) \leq m' \log m$, using (3) we have,

$$\hat{H}_d + \hat{H}_s \geq (1 - 2t^{-1})H_d + H_s \frac{\log c}{\log m}(1 - \epsilon^2(1-\epsilon)^{-1} \log_k N) \ .$$

The lemma follows by letting $\epsilon = \frac{\varepsilon}{2\sqrt{\log N}}$ and $t = \frac{m^{1/\alpha}}{\epsilon^2}$. □

Lower Bounds for Computation Over Strict Update Streams. In this section, we improve on the existing space lower bound of $\Omega(\phi^{-1} \log(\phi n))$ for point query estimation with accuracy parameter ϕ [4].

Lemma 7. *For $\phi > \frac{8}{\sqrt{n}}$, a deterministic point query estimator with parameter ϕ over strict update streams requires $\Omega(\phi^{-1}(\log m) \log(\phi n))$ space.*

Proof. Let $s = \lceil \phi^{-1} \rceil$. Consider a stream consisting of s^2 distinct items, organized into s levels $1, \ldots, s$ with s items per level. The frequency of an item at level l is set to $t_l = 2^{l-1}$. Let $\phi' = \frac{\phi}{16}$ and let A be a deterministic point query estimator with accuracy parameter ϕ'. We will apply A to obtain the identities of the items, level by level. Initially, the stream is inserted into the data structure of A. At iteration $r = 1, \ldots, s$ in succession, we maintain the invariant that items in levels higher than $s - r + 1$ have been discovered and their exact frequencies are deleted from A. Let $l = s - r + 1$. At the beginning of iteration r, the total frequency is $m = m_l = \sum_{u=1}^{l}(st_u) \leq s \sum_{u=1}^{l} 2^{l-1} < s2^l$. At iteration r, we use A to return the set of items x such that $\hat{f}_x \geq U_l = 2^{l-1} - 2^{l-4}$. Therefore, (a) estimated frequencies of items in level l cross the threshold U_l, since, $\hat{f}_x \geq f_x - \phi' m \geq 2^{l-1} - \frac{\phi 2^l s}{16} \geq U_l$, and, (b) estimated frequencies of items in level $l-1$ or lower do not cross U_l, since, $\hat{f}_y \leq f_y + \phi' m \leq 2^{l-2} + 2^{l-3} < U_l$. After s iterations, the level by level arrangement of the items can be reconstructed. The number of such arrangements is $\binom{n}{s\ s\ldots\ s}$ and therefore $\mathcal{A}$ requires space $\log \binom{n}{s\ s\ldots\ s} = \Omega(s^2 \log \frac{n}{s}) = \Omega(s(\log m)(\log \frac{n}{s}))$, since, $n > 64s^2$ and $m = m_s = s2^{s+1}$. □

Lemma 8. *For $\phi > 8n^{-1/2}$, any point query estimator for strict update streams with parameter ϕ and confidence 0.66 requires $\Omega(\phi^{-1}(\log m)(\log(\phi n)))$ bits.*

Proof. We reduce the bit-vector indexing problem to a randomized point query estimator. In the bit-vector indexing problem, bit vector v of size $|v|$ is presented followed by an index i between 1 and $|v|$. The problem is to decide whether $v[i] = 1$. It is known to require space $\Omega(|v|)$ by a randomized algorithm that gives the correct answer with probability $\frac{2}{3}$. Let $s = \lceil \phi^{-1} \rceil, |v| = s^2 \lceil \log(\lceil \frac{n}{s} \rceil) \rceil$ and $\rho = \lceil \log \lceil \frac{n}{s} \rceil \rceil$. We can isomorphically view the vector $v[1 \ldots |v|]$ as a set of contiguous segments τ of size ρ each, starting at positions 1 modulo ρ. The s^2 possible starting positions can be represented as (a_τ, b_τ), where, $a_\tau, b_\tau \in \{0, 1, \ldots, s-1\}$. Map each such segment to an item from the domain $s^2 2^\rho$ with $2\lceil \log s \rceil + \rho$ bit binary address $a_\tau \circ b_\tau \circ \tau$, where, $\circ$ represents bit concatenation. The frequency of the item is set to 2^{a_τ}. The bit vector is isomorphically mapped to a set of s^2 items of frequencies between 1 and 2^{s-1}, such that there are exactly s items with frequency 2^l, for $l = 0, 1, \ldots, s-1$. If the error probability of the

point estimator is at most $1-\frac{1}{3s^2}$, then, using the argument of Lemma 7, the set of all the s^2 items and their frequencies are correctly retrieved with error probability bounded by $\frac{s^2}{3s^2}=\frac{1}{3}$. That is, the bit vector is effectively reconstructed and the original bit vector index query can be answered. The above argument holds for every choice of the 1-1 onto mappings of the s^2 starting positions of ρ-size segments to (a_τ, b_τ). In particular, it holds for the specific map when the query index i belongs to a segment τ_0 whose starting position is mapped to the highest level, i.e., $b_{\tau_0}=s-1$. In this case, a single invocation of the point query suffices. Thus the space required is $\Omega(s^2\log\frac{n}{s})=\Omega(\phi^{-1}(\log m)(\log \phi n))$. □

3.2 General Update Streaming Model

In this section, we consider the general update streaming model. Lemma 9 presents the property of point query estimator for general update streams.

Lemma 9. *Consider a* CR-PRECIS *structure with height k and width t. For $x\in\mathcal{D}$, let $\hat{f}_x=\frac{1}{t}\sum_{j=1}^{t}T_j[x \mod p_j]$. Then, $|\hat{f}_x-f_x|\le\frac{(\log_k n-1)}{t}(L_1-|f_x|)$.*

Proof. $t\hat{f}_x=\sum_{j=1}^{t}T_j[x \mod p_j]=tf_x+\sum_{j=1}^{t}\sum\{f_y \mid y\neq x \text{ and } y\equiv x \bmod p_j\}$.

$$\text{Thus, } t|\hat{f}_x-f_x|=|\sum_{j=1}^{t}\sum_{y\neq x \atop y\equiv x \bmod p_j} f_y|=|\sum_{y\neq x}\sum_{j:y\equiv x \bmod p_j} f_y|$$

$$\le \sum_{y\neq x}\sum_{j:y\equiv x \bmod p_j}|f_y| \le (\log_k n-1)(F_1-|f_x|), \text{ by (11)}$$ □

Similarly, we can obtain an estimator for the inner-product of streams R and S. Let $L_1(R)$ and $L_1(S)$ be the L_1 norms of streams R and S respectively.

Lemma 10. *Consider a* CR-PRECIS *structure of height k and width t. Let $\hat{P}=\frac{1}{t}\sum_{j=1}^{t}\sum_{b=1}^{p_j}T_j[b]U_j[b]$. Then, $|\hat{P}-f\cdot g|\le\frac{(\log_k n-1)}{t}L_1(R)L_1(S)$.* □

Lower Bounds for Computations Over General Update Streams. We now present space lower bounds for problems over general update streams.

Lemma 11. *Deterministic algorithms for the following problems in the general update streaming model requires $\Omega(n)$ bits: (1) finding ϵ-approximate frequent items with parameter s for any $\epsilon<\frac{1}{2}$, (2) finding ϵ-approximate ϕ-quantiles for any $\epsilon<\phi/2$, (3) estimating the k^{th} norm $L_k=(\sum_{i=0}^{n-1}|f_i|^k)^{1/k}$, for any real value of k, to within any multiplicative approximation factor, and (4) estimating entropy to within any multiplicative approximation factor.*

Proof. Consider a family $\mathcal{F}$ of sets of size $\frac{n}{2}$ elements each such that the intersection between any two sets of the family does not exceed $\frac{n}{8}$. It can be shown[1] that

[1] Number of sets that are within a distance of $\frac{n}{8}$ from a given set of size $\frac{n}{2}$ is $\sum_{r=0}^{\frac{n}{8}}\binom{n/2}{r}^2\le 2\binom{n/2}{n/8}^2$. Therefore, $|\mathcal{F}|\ge\frac{\binom{n}{n/2}}{2\binom{n/2}{n/8}^2}\ge\frac{2^{n/2}}{2(3e)^{n/8}}=\frac{1}{2}\left(\frac{16}{3e}\right)^{n/8}$.

there exist such families of size $2^{\Omega(n)}$. Corresponding to each set S in the family, we construct a stream $str(S)$ such that $f_i = 1$ if $i \in S$ and $f_i = 0$, otherwise. Denote by $str_1 \circ str_2$ the stream where the updates of stream str_2 follow the updates of stream str_1 in sequence. Let $\mathcal{A}$ be a deterministic frequent items algorithm. Suppose that after processing two distinct sets S and T from $\mathcal{F}$, the same memory pattern of $\mathcal{A}$'s store results. Let Δ be a stream of deletions that deletes all but $\frac{s}{2}$ items from $str(S)$. Since, $L_1(str(S) \circ \Delta) = \frac{s}{2}$, all remaining $\frac{s}{2}$ items are found as frequent items. Further, $L_1(str(T) \circ \Delta) \geq \frac{n}{2} - \frac{s}{2}$, since, $|S \cap T| \leq \frac{n}{8}$. If $s < \frac{n}{3}$, then, $\frac{F_1}{s} > 1$, and therefore, none of the items qualify as frequent. Since, $str(S)$ and $str(T)$ are mapped to the same bit pattern, so are $str(S) \circ \Delta$ and $str(T) \circ \Delta$. Thus $\mathcal{A}$ makes an error in reporting frequent items in at least one of the two latter streams. Therefore, $\mathcal{A}$ must assign distinct bit patterns to each $str(S)$, for $S \in \mathcal{F}$. Since, $|\mathcal{F}| = 2^{\Omega(n)}$, $\mathcal{A}$ requires $\Omega(\log(|\mathcal{F}|)) = \Omega(n)$ bits, proving part (1) of the lemma.

Let S and T be sets from $\mathcal{F}$ such that $str(S)$ and $str(T)$ result in the same memory pattern of a quantile algorithm $\mathcal{Q}$. Let Δ be a stream that deletes all items from S and then adds item 0 with frequency $f_0 = 1$ to the stream. now all quantiles of $str(S) \circ \Delta = 0$. $str(T) \circ \Delta$ has at least $\frac{7n}{8}$ distinct items, each with frequency 1. Thus, for every $\phi < \frac{1}{2}$ and $\epsilon \leq \frac{\phi}{2}$ the kth ϕ quantile of the two streams are different by at least $k\phi n$. Part (3) is proved by letting Δ be an update stream that deletes all elements from $str(S)$. Then, $L_k(str(S) \circ \Delta) = 0$ and $L_k(str(T) \circ \Delta) = \Omega(n^{1/k})$.

Proceeding as above, suppose Δ is an update stream that deletes all but one element from $str(S)$. Then, $H(str(S) \circ \Delta) = 0$. $str(T) \circ \Delta$ has $\Omega(n)$ elements and therefore $H(str(T) \circ \Delta) = \log n + \Theta(1)$. The multiplicative gap $\log n : 0$ is arbitrarily large—this proves part (4) of the lemma. □

References

1. Alon, N., Gibbons, P.B., Matias, Y., Szegedy, M.: Tracking Join and Self-Join Sizes in Limited Storage. In: Proc. ACM PODS, ACM Press, New York (1999)
2. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems. In: Proc. ACM PODS, ACM Press, New York (2002)
3. Bhuvanagiri, L., Ganguly, S.: Estimating Entropy over Data Streams. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 148–159. Springer, Heidelberg (2006)
4. Bose, P., Kranakis, E., Morin, P., Tang, Y.: Bounds for Frequency Estimation of Packet Streams. In: SIROCCO, pp. 33–42 (2003)
5. Chakrabarti, A., Cormode, G., McGregor, A.: A Near-Optimal Algorithm for Computing the Entropy of a Stream. In: Proc. ACM SODA, ACM Press, New York (2007)
6. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 693–703. Springer, Heidelberg (2002)
7. Cormode, G., Garofalakis, M.: Sketching Streams Through the Net: Distributed Approximate Query Tracking. In: Proc. VLDB (September 2005)
8. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Finding Hierarchical Heavy Hitters in Data Streams. In: Proc. VLDB (2003)

9. Cormode, G., Muthukrishnan, S.: What's New: Finding Significant Differences in Network Data Streams. In: IEEE INFOCOM, IEEE Computer Society Press, Los Alamitos (2004)
10. Cormode, G., Muthukrishnan, S.: An Improved Data Stream Summary: The Count-Min Sketch and its Applications. J. Algorithms 55(1), 58–75 (2005)
11. Cormode, G., Muthukrishnan, S.: What's hot and what's not: tracking most frequent items dynamically. ACM Trans. Database Syst. 30(1), 249–278 (2005)
12. Demaine, E.D., López-Ortiz, A., Munro, J.I: Frequency estimation of internet packet streams with limited space. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, Springer, Heidelberg (2002)
13. Estan, C., Savage, S., Varghese, G.: Automatically inferring patterns of resource consumption in network traffic. In: Proc. ACM SIGCOMM, pp. 137–148. ACM Press, New York (2003)
14. Ganguly, S., Kesh, D., Saha, C.: Practical Algorithms for Tracking Database Join Sizes. In: Ramanujam, R., Sen, S. (eds.) FSTTCS 2005. LNCS, vol. 3821, Springer, Heidelberg (2005)
15. Ganguly, S., Majumder, A.: Deterministic K-set Structure. In: Proc. ACM PODS, ACM Press, New York (2006)
16. Gibbons, P.B., Matias, Y.: New Sampling-Based Summary Statistics for Improving Approximate Query Answers. In: Proc. ACM SIGMOD, ACM Press, New York (1998)
17. Gilbert, A., Guha, S., Indyk, P., Kotidis, Y., Muthukrishnan, S., Strauss, M.: Fast Small-space Algorithms for Approximate Histogram Maintenance. In: Proc. ACM STOC, ACM Press, New York (2002)
18. Gilbert, A., Kotidis, Y., Muthukrishnan, S., Strauss, M.: How to Summarize the Universe: Dynamic Maintenance of Quantiles. In: Bressan, S., Chaudhri, A.B., Lee, M.L., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, Springer, Heidelberg (2003)
19. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.J.: Surfing Wavelets on Streams: One-pass Summaries for Approximate Aggregate Queries. In: Jonker, W. (ed.) VLDB-WS 2001 and DBTel 2001. LNCS, vol. 2209, Springer, Heidelberg (2001)
20. Greenwald, M., Khanna, S.: Space-efficient online computation of quantile summaries. In: SIGMOD (2001)
21. Hershberger, J., Shrivastava, N., Suri, S., Toth, C.D.: Space Complexity of Hierarchical Heavy Hitters in Multi-Dimensional Data Streams. In: Proc. ACM PODS, ACM Press, New York (2005)
22. Karp, R.M., Shenker, S., Papadimitriou, C.H.: A Simple Algorithm for Finding Frequent Elements in Streams and Bags. ACM TODS 28(1), 51–55 (2003)
23. Manku, G., Motwani, R.: Approximate Frequency Counts over Data Streams. In: Bressan, S., Chaudhri, A.B., Lee, M.L., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, pp. 346–357. Springer, Heidelberg (2003)
24. Manku, G., Rajagopalan, S., Lindsay, B.: Random sampling techniques for space efficient online computation of order statistics of large datasets. In: Proc. ACM SIGMOD, ACM Press, New York (1999)
25. Misra,J.,Gries,D.:Findingrepeatedelements.Sci.Comput.Programm.2,143–152(1982)
26. Muthukrishnan, S.: Data Streams: Algorithms and Applications. Foundations and Trends in Theoretical Computer Science 1(2) (2005)
27. Rosser, J.B.: Explicit bounds on some functions on prime numbers. Amer. J. Math. 63 (1941)
28. Schweller, R., Li, Z., Chen, Y., Gao, Y., Gupta, A., Zhang, Y., Dinda, P., Kao, M-Y., Memik, G.: Monitoring Flow-level High-speed Data Streams with Reversible Sketches. In: IEEE INFOCOM, IEEE Computer Society Press, Los Alamitos (2006)

An Effective Refinement Algorithm Based on Swarm Intelligence for Graph Bipartitioning

Lingyu Sun[1] and Ming Leng[2]

[1] Department of Computer Science,
Jinggangshan College, Ji'an, PR China 343009
[2] School of Computer Engineering and Science,
Shanghai University, Shanghai, PR China 200072
sunlingyu@jgsu.edu.cn, lengming@shu.edu.cn

Abstract. Partitioning is a fundamental problem in diverse fields of study such as VLSI design, parallel processing, data mining and task scheduling. The min-cut bipartitioning problem is a fundamental graph partitioning problem and is NP-Complete. In this paper, we present an effective multi-level refinement algorithm based on swarm intelligence for bisecting graph. The success of our algorithm relies on exploiting both the swarm intelligence theory with a boundary refinement policy. Our experimental evaluations on 18 different benchmark graphs show that our algorithm produces high quality solutions compared with those produced by MeTiS that is a state-of-the-art partitioner in the literature.

1 Introduction

Partitioning is a fundamental problem with extensive applications to many areas, including VLSI design [1], parallel processing [2], data mining [3] and task scheduling [4]. The problem is to partition the vertices of a graph into k roughly equal-size sub-domains, such that the number of the edges connecting vertices in different sub-domains is minimized. The *min-cut bipartitioning problem* is a fundamental partitioning problem and is NP-Complete. It is also NP-Hard to find good approximate solutions for this problem [5]. The survey by Alpert and Kahng [1] provides a detailed description and comparison of various such schemes which can be classified as *move-based* approaches, *geometric representations*, *combinatorial* formulations, and *clustering* approaches.

Because of its importance, the problem has attracted a considerable amount of research interest and a variety of algorithms have been developed over the last thirty years [6],[7]. Most existing partitioning algorithms are heuristics in nature and they seek to obtain reasonably good solutions in a reasonable amount of time. For example, Kernighan and Lin (KL) [6] proposed an iterative improvement algorithm for partitioning graphs that consists of making several improvement passes. Fiduccia and Mattheyses (FM) [7] proposed a fast heuristic algorithm for bisecting a weighted graph by introducing the concept of cell *gain* into the KL algorithm. These algorithms belong to the class of *move-based* approaches in which the solution is built iteratively from an initial solution by applying a move or transformation to the current solution. Move-based approaches

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 60–69, 2007.

are the most frequently combined with stochastic hill-descending algorithms such as those based on Simulated Annealing [8], Tabu Search [8], Genetic Algorithms [9], Neural Networks [10], which allow movements towards solutions worse than the current one in order to escape from local minima. For example, Leng and Yu [11],[12] proposed a boundary Tabu Search refinement algorithm that combines an effective Tabu Search strategy with a boundary refinement policy.

As the problem sizes reach new levels of complexity recently, a new class of graph partitioning algorithms have been developed that are based on the multi-level paradigm. The multi-level graph partitioning schemes consist of three phases [13],[14],[15]. During the *coarsening phase*, a sequence of successively coarser graph is constructed by collapsing vertex and edge until its size is smaller than a given threshold. The goal of the *initial partitioning phase* is to compute initial partition of the coarsest graph such that the balancing constraint is satisfied and the partitioning objective is optimized. During the *uncoarsening and refinement phase*, the partitioning of the coarser graph is successively projected back to the next level finer graph and an iterative refinement algorithm is used to optimize the objective function without violating the balancing constraint.

In this paper, we present a multi-level refinement algorithm which combines the *swarm intelligence* theory with a boundary refinement policy. Our work is motivated by the multi-level *ant* colony algorithm(MACA) of Korošec who runs basic *ant* colony algorithm on every level graph in [16] and Karypis who proposes the boundary KL (BKL) refinement algorithm in [14] and supplies **MeTiS** [13], distributed as open source software package for partitioning unstructured graphs. We test our algorithm on 18 graphs that are converted from the hypergraphs of the ISPD98 benchmark suite [17]. Our algorithm has shown encouraging performance in the comparative experiments which produce excellent partitions that are better than those produced by **MeTiS** in a reasonable time.

The rest of the paper is organized as follows. Section 2 provides some definitions and describes the notation that is used throughout the paper. Section 3 briefly describes the motivation behind our algorithm. Section 4 presents an effective multi-level *swarm intelligence* refinement algorithm. Section 5 experimentally evaluates our algorithm and compares it with **MeTiS**. Finally, Section 6 provides some concluding remarks and indicates the directions for further research.

2 Mathematical Description

A graph $G=(V,E)$ consists of a set of vertices V and a set of edges E such that each edge is a subset of two vertices in V. Throughout this paper, n and m denote the number of vertices and edges respectively. The vertices are numbered from *1* to n and each vertex $v \in V$ has an integer weight $S(v)$. The edges are numbered from *1* to m and each edge $e \in E$ has an integer weight $W(e)$. A decomposition of a graph V into two disjoint subsets V^1 and V^2, such that $V^1 \cup V^2 = V$ and $V^1 \cap V^2 = \varnothing$, is called a *bipartitioning* of V. Let $S(A) = \sum_{v \in A} S(v)$ denotes the size of a subset $A \subseteq V$. Let ID_v be denoted as v's *internal degree* and is equal to

the sum of the edge-weights of the adjacent vertices of v that are in the same side of the partition as v, and v's *external degree* denoted by ED_v is equal to the sum of edge-weights of the adjacent vertices of v that are in different sides. The *cut* of a *bipartitioning* $P=\{V^1,V^2\}$ is the sum of weights of edges which contain two vertices in V^1 and V^2 respectively. Naturally, vertex v belongs at the boundary if and only if $ED_v > 0$ and the *cut* of P is also equal to $0.5 \sum_{v \in V} ED_v$. Given a balance constraint r, the *min-cut bipartitioning problem* seeks a solution $P=\{V^1,V^2\}$ that minimizes *cut(P)* subject to $(1\text{-}r)S(V)/2 \leq S(V^1), S(V^2) \leq (1+r)S(V)/2$. A *bipartitioning* is *bisection* if r is as small as possible. The task of minimizing *cut(P)* can be considered as the *objective* and the requirement that solution P will be of the same size can be considered as the *constraint*.

3 Motivation

Over the last decades, researchers have been looking for new paradigms in optimization. *Swarm intelligence* arose as one of the paradigms based on natural systems and its five basic principles are *proximity*, *quality*, *diverse response*, *stability* and *adaptability* [18]. Formally, the term *swarm* can be defined as a group of agents which communicate with each other, by acting on their local environment. Although the individuals of the *swarm* are relatively simple in structure, their collective behavior is usually very complex. The complex behavior of the *swarm*, as distributive collective problem-solving strategies, is a result of the pattern of interactions between the individuals of the *swarm* over time. *Swarm intelligence* is the property of a system whereby the collective behaviors of unsophisticated agents interacting locally with their environment cause coherent functional global patterns to emerge.

The *ant* colony optimization (ACO) and the particle swarm optimization (PSO) are two attractive topics for researchers of *swarm intelligence*. ACO is a population-based meta-heuristic framework for solving discrete optimization problems [19]. It is based on the indirect communication among the individuals of a colony of agents, called *ants*, mediated by trails of a chemical substance, called *pheromone*, which real *ants* use for communication. It is inspired by the behavior of real *ant* colonies, in particular, by their foraging behavior and their communication through *pheromone* trails. PSO is also a population-based optimization method first proposed by Kennedy and Eberhart [20]. It has been shown to be effective in optimizing difficult multidimensional discontinuous problems. PSO is initialized with a random population of candidate solutions that are flown in the multidimensional search space in search for the optimum solution. Each individual of the population, called *particle*, has an *adaptable velocity* according to which it moves in the search space. Moreover, each *particle* has a *memory*, remembering the best position of the search space it has ever visited. Thus, its movement is an aggregated acceleration towards its best previously visited position and towards the best *particle* of a topological neighborhood.

In [21], Langham and Grant proposed the Ant Foraging Strategy (AFS) for k-way partitioning. The basic idea of the AFS algorithm is very simple: We have

k colonies of *ants* that are competing for food, which in this case represents the vertices of the graph. At the end the *ants* gather food to their nests, i.e. they partition the graph into k subgraphs. In [16], Korošec presents the MACA approach that is enhancement of the AFS algorithm with the multi-level paradigm. However, since Korošec simply runs the AFS algorithm on every level ℓ graph $G_l(V_l,E_l)$, most of computation on the coarser graphs is wasted. Furthermore, MACA comes into collision with the key idea behind the multi-level approach. The multi-level graph partitioning schemes needn't the direct partitioning algorithm on $G_l(V_l,E_l)$ in the *uncoarsening and refinement phase*, but the refinement algorithm that improves the quality of the finer graph $G_l(V_l,E_l)$ partitioning $P_{G_l}=\{V_l^1,V_l^2\}$ which is projected from the partitioning $P_{G_{l+1}}=\{V_{l+1}^1,V_{l+1}^2\}$of the coarser graph $G_{l+1}(V_{l+1},E_{l+1})$.

In this paper, we present a new multi-level *swarm intelligence* refinement algorithm(MSIR) that combines the *swarm intelligence* theory with a boundary refinement policy. It employs *swarm intelligence* in order to select two subsets of vertices $V_l^{1'} \subset V_l^1$ and $V_l^{2'} \subset V_l^2$ such that $\{(V_l^1 - V_l^{1'}) \cup V_l^{2'}, (V_l^2 - V_l^{2'}) \cup V_l^{1'}\}$ is a bisection with a smaller edge-cut. It has distinguishing features which are different from the MACA algorithm. First, MACA exploits two or more colonies of *ants* to compete for the vertices of the graph, while MSIR employs one *swarm* to find $V_l^{1'}$ and $V_l^{2'}$ such that moving them to the other side improves the quality of partitioning. Second, MACA is a partitioning algorithm while MSIR is a refinement algorithm. Finally, MSIR is a boundary refinement algorithm whose runtime is significantly smaller than that of a non-boundary refinement algorithm, since the vertices moved by MSIR are boundary vertices that straddle two sides of the partition and only the gains of boundary vertices are computed.

4 MSIR: The Framework

Informally, the MSIR algorithm works as follows: At time zero, an initialization phase takes place during which initial values for *pheromone* trail are set on the vertices of graph G and a population of agents is initialized, using the initial partitioning as the individual's best partition. In the main loop of MSIR, each *agent*'s tabu list is emptied and each *agent* chooses $(V^{1'}, V^{2'})$ by repeatedly selecting boundary vertices of each part according to a *state transition rule* given by Equation(1) (2), moving them into the other part, updating the gains of the remaining vertices and etc. After constructing its solution, each *agent* also modifies the amount of *pheromone* on the moved vertices by applying the *local updating rule* of Equation(3). Once all agents have terminated their solutions, the amount of *pheromone* on vertices is modified again by applying the *global updating rule* of Equation(4). The process is iterated until the cycles counter reaches the maximum number of cycles NC_{max}, or the MSIR algorithm stagnates.

The pseudocode of the MSIR algorithm is shown in Algorithm 1. The cycles counter is denoted by t. *Best* represents the best partitioning seen by the *swarm* so far and $Best^k$ represents the best partitioning visited by *agent* k. The initial values for *pheromone* trail is denoted by $\tau_0=1/\varepsilon$, where ε is total number of

agents. At *cycle* t, let $\tau_v(t)$ be the *pheromone* trail on the vertex v and $tabu^k(t)$ be the tabu list of *agent* k, $Best^k(t)$ represents the best partitioning found by *agent* k and the current partitioning of *agent* k is denoted by $P^k(t)$, the *agent* k also stores the internal and external degrees of all vertices and boundary vertices independently which be denoted as $ID^k(t)$, $ED^k(t)$ and $boundary^k(t)$ respectively. Let $allowed^k(t)$ be denoted as the *candidate* list which is a list of preferred vertices to be moved by *agent* k at *cycle* t and is equal to $\{V - tabu^k(t)\} \bigcap boundary^k(t)$.

Algorithm 1 (MSIR)

MSIR(initial bipartitioning P, maximum number of cycles NC_{max},
balance constraint r, similarity tolerance φ, maximum steps s_{max})
/***Initialization***/
$t = 0$
$Best = P$
For every vertex v in $G = (V, E)$ do
$\tau_v(t) = \tau_0$
$ID_v = \sum_{(v,u)\in E \wedge P[v]=P[u]} W(v,u)$
$ED_v = \sum_{(v,u)\in E \wedge P[v]\neq P[u]} W(v,u)$
Store v as *boundary vertex* if and only if $ED_v > 0$;
End For
For $k = 1$ to ε do
Constrcut *agent* k and Store $Best^k = P$ independently;
End For
/***Main loop***/
For $t = 1$ to NC_{max} do
For $k = 1$ to ε do
$tabu^k(t) = \varnothing$
Store $P^k(t) = P$ and $Best^k(t) = P$ independently;
Store $ID^k(t)$, $ED^k(t)$, $boundary^k(t)$ of $G = (V, E)$ independently;
For $s = 1$ to s_{max} do
Decide the *move direction* of the current step s;
If exists at least one vertex $v \in allowed^k(t)$ then
Choose the vertex v to move as follows

$$v = \begin{cases} \arg \max\limits_{v \in allowed^k(t)} \left[\psi_v^k(t)\right]^\alpha \cdot \left[\eta_v^k(t)\right]^\beta & \text{if } q \le q_0 \\ w & \text{if } q > q_0 \end{cases} \tag{1}$$

Where the vertex w is chosen according to the probablity

$$p_w^k(t) = \begin{cases} \frac{\left[\psi_w^k(t)\right]^\alpha \cdot \left[\eta_w^k(t)\right]^\beta}{\sum_{u \in allowed^k(t)} \left[\psi_u^k(t)\right]^\alpha \cdot \left[\eta_u^k(t)\right]^\beta} & \text{if } w \in allowed^k(t) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Else

Break;
End If
Update $P^k(t)$ by moving the vertex v to the other side;
Lock the vertex v by adding to $tabu^k(t)$;
original cut Minus its original *gain* as the *cut* of $P^k(t)$;
Update $ID_u^k(t)$, $ED_u^k(t)$, *gain* of its neighboring vertices u and $boundary^k(t)$;
If ($cut(P^k(t)) < cut(Best^k(t))$ and $P^k(t)$ satisfies constraints r) then
$Best^k(t) = P^k(t)$
End If
End For /*$s \leq s_{max}$*/
Apply the *local update rule* for the vertices v moved by *agent* k

$$\tau_v(t) \leftarrow (1-\rho) \cdot \tau_v(t) + \rho \cdot \triangle\tau_v^k(t) \tag{3}$$

Update $Best^k$ and $cut(Best^k)$ if $cut(Best^k(t)) < cut(Best^k)$;
End For /*$k \leq \varepsilon$*/
Apply the *global update rule* for the vertices v moved by *global-best agent*

$$\tau_v(t) \leftarrow (1-\xi) \cdot \tau_v(t) + \xi \cdot \triangle\tau_v^{gb} \tag{4}$$

Update $Best$ and $cut(Best)$ if $\min\limits_{1 \leq k \leq \varepsilon} cut(Best^k) < cut(Best)$;
For every vertex v in $G = (V, E)$ do
$\tau_v(t+1) = \tau_v(t)$
End For
End For /*$t \leq NC_{max}$*/
Return $Best$ and $cut(Best)$

In the MSIR algorithm, a *state transition rule* given by Equation(1)(2) is called *pseudo-random-proportional rule*, where q is a random number uniformly distributed in $[0 \ldots 1]$ and q_0 is parameter $(0 \leq q_0 \leq 1)$ which determines the relative importance of exploitation versus exploration. To avoid trapping into *stagnation behavior*, MSIR adjusts dynamically the parameter q_0 based on the solutions *similarity* between $(V^{1'}, V^{2'})^k$ and $(V^{1'}, V^{2'})^{(k-1)}$ found by *agent* k and *k-1*. In Equation(1)(2), α and β denote the relative importance of the *revised pheromone trail* and *visibility* respectively. $\eta_v^k(t)$ represents the *visibility* of *agent* k on the vertex v at *cycle* t and is given by:

$$\eta_v^k(t) = \begin{cases} \sqrt{1.0 + ED_v^k(t) - ID_v^k(t)} & \text{if } (ED_v^k(t) - ID_v^k(t)) \geq 0 \\ \sqrt{1.0/(ID_v^k(t) - ED_v^k(t))} & \text{otherwise} \end{cases} \tag{5}$$

$\psi_v^k(t)$ represents the *revised pheromone trail* of *agent* k on the vertex v at *cycle* t and is given by:

$$\psi_v^k(t) = \omega \cdot \tau_v(t) + (\lambda_1 \cdot rand() \cdot \delta_1 + \lambda_2 \cdot rand() \cdot \delta_2) \cdot \eta_v^k(t)^2 / cut(P) \tag{6}$$

$$\delta_1 = \begin{cases} 1 & \text{if } Best^k[v] \neq P^k[v] \\ -1 & \text{if } Best^k[v] = P^k[v] \end{cases} \qquad \delta_2 = \begin{cases} 1 & \text{if } Best[v] \neq P^k[v] \\ -1 & \text{if } Best[v] = P^k[v] \end{cases} \tag{7}$$

where ω is called *inertia weight* and regulates the trade-off between the global and local exploration abilities of the *swarm*; λ_1, called *cognitive* parameter, is a factor determining how much the *agent* is influenced by $Best^k$; λ_2, called *social* parameter, is a factor determining how much the *agent* is influenced by *Best*; The difference between the *agent*'s previous best and current partitioning on vertex v is denoted by δ_1; δ_2 denotes the difference between the global best partitioning and current partitioning on vertex v.

In Equation(3), ρ is a coefficient and represents the local evaporation of *pheromone* trail between *cycle* t and $t+1$ and the term $\triangle\tau_v^k(t)$ is given by:

$$\triangle\tau_v^k(t) = \begin{cases} \frac{cut(Best^k(t)) - cut(P)}{cut(P)\cdot\varepsilon} & \text{if } v \text{ was moved by } agent\ k \text{ at } cycle\ t \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

In Equation(4), ξ is a parameter and represents the global evaporation of *pheromone* trail between *cycle* t and $t+1$ and the term $\triangle\tau_v^{gb}$ is given by:

$$\triangle\tau_v^{gb} = \begin{cases} \frac{cut(Best) - cut(P)}{cut(P)} & \text{if } v \text{ was moved by } global\text{-}best\ agent \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

5 Experimental Results

We use the 18 graphs in our experiments that are converted from the hypergraphs of the ISPD98 benchmark suite [17] and range from 12,752 to 210,613 vertices. Each hyperedge is a subset of two or more vertices in hypergraph. We convert hyperedges into edges by the rule that every subset of two vertices in hyperedge can be seemed as edge [11],[12] and store 18 edge-weighted and vertex-weighted graphs in format of **MeTiS** [13]. The characteristics of these graphs are shown in Table 1.

We implement the MSIR algorithm in ANSI C and integrate it with the leading edge partitioner **MeTiS**. In the evaluation of our algorithm, we must make sure that the results produced by our algorithm can be easily compared against those produced by **MeTiS**. First, we use the same balance constraint r and random seed in every comparison. Second, we select the sorted heavy-edge matching (SHEM) algorithm during the *coarsening phase* because of its consistently good behavior in **MeTiS**. Third, we adopt the greedy graph growing partition algorithm during the *initial partitioning phase* that consistently finds smaller edge-cuts than other algorithms. Finally, we select the BKL algorithm to compare with MSIR during *uncoarsening and refinement phase* because BKL can produce smaller edge-cuts when coupled with the SHEM algorithm. These measures are sufficient to guarantee that our experimental evaluations are not biased in any way.

The quality of partitions produced by our algorithm and those produced by **MeTiS** are evaluated by looking at two different quality measures, which are the

Table 1. The characteristics of 18 graphs to evaluate our algorithm

benchmark	vertices	hyperedges	edges
ibm01	12752	14111	109183
ibm02	19601	19584	343409
ibm03	23136	27401	206069
ibm04	27507	31970	220423
ibm05	29347	28446	349676
ibm06	32498	34826	321308
ibm07	45926	48117	373328
ibm08	51309	50513	732550
ibm09	53395	60902	478777
ibm10	69429	75196	707969
ibm11	70558	81454	508442
ibm12	71076	77240	748371
ibm13	84199	99666	744500
ibm14	147605	152772	1125147
ibm15	161570	186608	1751474
ibm16	183484	190048	1923995
ibm17	185495	189581	2235716
ibm18	210613	201920	2221860

Table 2. Min-cut bipartitioning results with up to 2% deviation from exact bisection

benchmark	Metis(α)		MSIR(β)		ratio(β:α)	
	MinCut	AveCut	MinCut	AveCut	MinCut	AveCut
ibm01	517	1091	505	758	0.977	0.695
ibm02	4268	11076	2952	7682	0.692	0.694
ibm03	10190	12353	4452	6381	0.437	0.517
ibm04	2273	5716	2219	3464	0.976	0.606
ibm05	12093	15058	12161	14561	**1.006**	0.967
ibm06	7408	13586	2724	7715	**0.368**	0.568
ibm07	3219	4140	2910	3604	0.904	0.871
ibm08	11980	38180	11038	15953	0.921	**0.418**
ibm09	2888	4772	2857	3524	0.989	0.738
ibm10	10066	17747	5915	9966	0.588	0.562
ibm11	2452	5095	2421	4218	0.987	0.828
ibm12	12911	27691	10303	16609	0.798	0.600
ibm13	6395	13469	5083	10178	0.795	0.756
ibm14	8142	12903	8066	12959	0.991	**1.004**
ibm15	22525	46187	12105	31399	0.537	0.680
ibm16	11534	22156	10235	14643	0.887	0.661
ibm17	16146	26202	15534	20941	0.962	0.799
ibm18	15470	20018	15536	17521	1.004	0.875
average					**0.823**	**0.713**

minimum *cut* (MinCut) and the average *cut* (AveCut). To ensure the statistical significance of our experimental results, two measures are obtained in twenty runs whose random seed is different to each other. For all experiments, we use a

49-51 *bipartitioning* balance constraint by setting r to 0.02 and set the number of vertices of the current level graph as the value of s_{max}. Furthermore, we adopt the experimentally determined optimal set of parameters values for MSIR, α=2.0, β=1.0, ρ=0.1, ξ=0.1, q_0=0.9, φ=0.8, ε=10, ω=2, λ_1=1, λ_2=1, NC_{max}=80.

Table 2 presents *min-cut bipartitioning* results allowing up to 2% deviation from exact bisection and gives the MinCut and AveCut comparisons of two algorithms on 18 graphs. As expected, our algorithm reduces the AveCut by -0.4% to 58.2% and reaches 28.7% average AveCut improvement. In the MinCut evaluation, we obtain 17.7% average improvement and between -0.6% and 63.2% improvement. All evaluations that twenty runs of two algorithms on 18 graphs are run on an 1800MHz AMD Athlon2200 with 512M memory and can be done in two hours.

6 Conclusions

In this paper, we have presented an effective multi-level algorithm based on *swarm intelligence*. The success of our algorithm relies on exploiting both the *swarm intelligence* theory with a boundary refinement policy. We obtain excellent *bipartitioning* results compared with those produced by **MeTiS**. Although it has the ability to find cuts that are lower than the result of **MeTiS** in a reasonable time, there are several ways in which this algorithm can be improved. In MSIR, we have a set of parameters (α, β, ρ, ξ, q_0, φ, ε, ω, λ_1, λ_2, NC_{max}) to balance the relative importance of exploitation versus exploration, to regulate the trade-off between the global and local exploration abilities and etc. However, in the MinCut evaluation of benchmark ibm05 and ibm18, our algorithm is 0.6% worse than **MeTiS**. Therefore, the question is to guarantee find good approximate solutions by setting optimal set of parameters values for MSIR. Ultimately, we may wonder if it is possible to let the algorithm determine the right value of parameters in a preprocessing step.

Acknowledgments

This work was supported by the international cooperation project of Ministry of Science and Technology of PR China, grant No. CB 7-2-01, and by "SEC E-Institute: Shanghai High Institutions Grid" project. Meanwhile, the authors would like to thank professor Karypis of university of Minnesota for supplying source code of **MeTiS**. The authors also would like to thank Alpert of IBM Austin Research Laboratory for supplying the ISPD98 benchmark suite.

References

1. Alpert, C.J., Kahng, A.B.: Recent directions in netlist partitioning. Integration, the VLSI Journal 19, 1–81 (1995)
2. Hendrickson, B., Leland, R.: An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM Journal on Scientific Computing 16, 452–469 (1995)

3. Ding, C., He, X., Zha, H., Gu, M., Simon, H.: A Min-Max cut algorithm for graph partitioning and data clustering. In: Proc. IEEE Conf Data Mining, pp. 107–114. IEEE Computer Society Press, Los Alamitos (2001)
4. Khannat, G., Vydyanathant, N.: A hypergraph partitioning based approach for scheduling of tasks with batch-shared I/O. In: IEEE International Symposium on Cluster Computing and the Grid, pp. 792–799 (2005)
5. Bui, T., Leland, C.: Finding good approximate vertex and edge partitions is NP-hard. Information Processing Letters 42, 153–159 (1992)
6. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal 49, 291–307 (1970)
7. Fiduccia, C., Mattheyses, R.: A linear-time heuristics for improving network partitions. In: Proc. 19th Design Automation Conf., pp. 175–181 (1982)
8. Tao, L., Zhao, Y.C., Thulasiraman, K., Swamy, M.N.S.: Simulated annealing and tabu search algorithms for multiway graph partition. Journal of Circuits, Systems and Computers, 159–185 (1992)
9. Żola, J., Wyrzykowski, R.: Application of genetic algorithm for mesh partitioning. In: Proc. Workshop on Parallel Numerics, pp. 209–217 (2000)
10. Bahreininejad, A., Topping, B.H.V., Khan, A.I.: Finite element mesh partitioning using neural networks. Advances in Engineering Software, pp. 103–115 (1996)
11. Leng, M., Yu, S., Chen, Y.: An effective refinement algorithm based on multi-level paradigm for graph bipartitioning. In: The IFIP TC5 International Conference on Knowledge Enterprise. IFIP Series, pp. 294–303. Springer, Heidelberg (2006)
12. Leng, M., Yu, S.: An effective multi-level algorithm for bisecting graph. In: Li, X., Zaïane, O.R., Li, Z. (eds.) ADMA 2006. LNCS (LNAI), vol. 4093, pp. 493–500. Springer, Heidelberg (2006)
13. Karypis, G., Kumar, V.: MeTiS 4.0: Unstructured graphs partitioning and sparse matrix ordering system. Technical Report, Department of Computer Science, University of Minnesota (1998)
14. Karypis, G., Kumar, V.: A fast and highly quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing 20, 359–392 (1998)
15. Selvakkumaran, N., Karypis, G.: Multi-objective hypergraph partitioning algorithms for cut and maximum subdomain degree minimization. IEEE Trans. Computer Aided Design 25, 504–517 (2006)
16. Korošec, P., Šilc, J., Robič, B.: Solving the mesh-partitioning problem with an ant-colony algorithm. Parallel Computing, 785–801 (2004)
17. Alpert, C.J.: The ISPD98 circuit benchmark suite. In: Proc. Intel Symposium of Physical Design, pp. 80–85 (1998)
18. Millonas, M.: Swarms, phase transitions, and collective intelligence. In: Langton, C. (ed.) Artificial Life III, Addison-Wesley, Reading (1994)
19. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: Optimization by a colony of cooperating agents. IEEE Trans on SMC, 29–41 (1996)
20. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. IEEE Conf Neural Networks IV, pp. 1942–1948. IEEE Computer Society Press, Los Alamitos (1995)
21. Langham, A.E., Grant, P.W.: Using competing ant colonies to solve k-way partitioning problems with foraging and raiding strategies. In: Floreano, D., Mondada, F. (eds.) ECAL 1999. LNCS, vol. 1674, pp. 621–625. Springer, Heidelberg (1999)

On the Complexity and Approximation of the Min-Sum and Min-Max Disjoint Paths Problems

Peng Zhang* and Wenbo Zhao**

State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, P.O. Box 8718, Beijing, 100080, China
Graduate University of Chinese Academy of Sciences, Beijing, China
zhangpeng04@iscas.cn, zwenbo@gcl.iscas.ac.cn

Abstract. Given a graph $G = (V, E)$ and k source-sink pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ with each $s_i, t_i \in V$, the Min-Sum Disjoint Paths problem asks k disjoint paths to connect all the source-sink pairs with minimized total length, while the Min-Max Disjoint Paths problem asks also k disjoint paths to connect all source-sink pairs but with minimized length of the longest path. In this paper we show that the weighted Min-Sum Disjoint Paths problem is $\mathbf{FP}^{\mathbf{NP}}$-complete in general graph, and the uniform Min-Sum Disjoint Paths and uniform Min-Max Disjoint Paths problems can not be approximated within $\Omega(m^{1-\epsilon})$ for any constant $\epsilon > 0$ even in planar graph if $\mathbf{P} \neq \mathbf{NP}$, where m is the number of edges in G. Then we give at the first time a simple bicriteria approximation algorithm for the uniform Min-Max Edge-Disjoint Paths and the weighted Min-Sum Edge-Disjoint Paths problems, with guaranteed performance ratio $O(\log k / \log \log k)$ and $O(1)$ respectively. Our algorithm is based on randomized rounding.

1 Introduction

1.1 The Problems

Disjoint paths problem is a classical problem in combinatorial optimization and graph theory. This problem finds its practical applications in network routing, VLSI-design and other areas. Nowadays disjoint paths problem has being paid more and more attention due to the rapid development of the network technology. See Kleinberg's remarkable thesis for background and motivation [4]. The classical disjoint paths problem is to find disjoint paths to connect as many as possible source-sink pairs in a graph, see [4,2,5]. We consider the Min-Sum Disjoint Paths problem and the Min-Max Disjoint Paths problem in this paper.

Two paths are said to be *vertex-disjoint* if they do not share any vertex, and are said to be *edge-disjoint* if they do not share any edge. In the Min-Sum

* Supported by NSFC grants No. 60325206 and No. 60310213. This work is part of the author's Ph.D. thesis prepared at Institute of Software, CAS under the supervision of Prof. Angsheng Li.

** Supported by NSFC grants No. 60325206 and No. 60310213.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 70–81, 2007.

Disjoint Paths problem (Min-Sum DP), given a graph $G = (V, E)$, k source-sink pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ with each $s_i, t_i \in V$, and a weight $w_e \geq 0$ for every $e \in E$, the goal of the problem is to find k disjoint paths to connect every source-sink pair, while the total length of these paths is minimized. The instance of the Min-Max Disjoint Paths problem (Min-Max DP) is the same as that of Min-Sum DP, but the goal is to find k disjoint paths to connect every source-sink pair with minimized length of the longest path. Throughout this paper, we always use n to denote $|V|$, and use m to denote $|E|$. If the weights are identical for all edges (without loss of generality, we may assume that $w_e = 1$ for every edge e), then we call the problem *uniform*. We also use EDP to denote the term "Edge-Disjoint Paths", and use VDP to denote the term "Vertex-Disjoint Paths". When we talk about DP, we mean both EDP and VDP. Source and sink are also called *terminal*, meanwhile a source-sink pair is also called a *demand*. We call an instances of the problem *feasible* if all source-sink pairs can be connected by disjoint paths. As usual, let I denote an instance of some problem, and let $OPT(I)$ denote the value of an optimal solution to I.

1.2 Related Work

Suurballe and Tarjan gave a polynomial-time algorithm for the Min-Sum problem that finds the shortest pair of edge-disjoint and vertex-disjoint paths from one source to one sink both in undirected and directed graph [10]. Yang and Zheng extended the algorithm in [10] to solve the problem that finds all shortest disjoint path pairs from one source s to every sink pairs (t_1, t_2) in a graph [12]. For the problem that finds two normalized Min-Sum s–t disjoint paths, Yang, Zheng and Lu proved that the problem is NP-hard and gave a $(1+\alpha)/2\alpha$-approximation algorithm [11], where α is the normalization parameter given by the problem. Li, Thomas and Simchi-Levi considered the Min-Max problem of finding two disjoint paths from one source to one sink such that the length of the longer path is minimized [6]. They showed that both of the four versions of the problem, i.e., the graph may be directed or undirected, and the paths may be edge-disjoint or vertex-disjoint, are strongly **NP**-hard. They also gave a pseudo-polynomial time algorithm for the problem in directed acyclic graph. Brandes, Neyer and Wagner proved that the uniform Min-Sum EDP problem and the uniform Min-Max EDP problem are **NP**-hard in planar graph [1], even if the graph fullfils the Eulerian condition and the maximum degree is four.

1.3 Our Results

We systematically study the Min-Sum DP and Min-Max DP problems. The main contribution of the paper is the new and strong complexity and hardness results and better approximate solutions to the problems, although the problems are very hard in terms of approximation hardness as we prove.

Specifically, we show that the weighted Min-Sum DP problem is $\mathbf{FP}^{\mathbf{NP}}$-complete both in undirected and directed graphs, and the uniform Min-Sum DP and uniform Min-Max DP problems can not be approximated within $\Omega(m^{1-\epsilon})$

for any constant $\epsilon > 0$ even in planar graph, unless $\mathbf{P} = \mathbf{NP}$. It is well known that to decide whether the input instance is feasible (i.e., all the source-sink pairs can be connected by disjoint paths) is **NP**-hard. Our results are strong since we prove all of the above complexity and hardness results even if the given instances are known to be feasible. Recall that $\mathbf{FP}^{\mathbf{NP}}$ is the class of all functions from strings to strings that can be computed by a polynomial-time Turing machine with a SAT oracle. If an optimization problem is $\mathbf{FP}^{\mathbf{NP}}$-complete, then it is **NP**-hard. But the opposite direction is not true. So, $\mathbf{FP}^{\mathbf{NP}}$-complete is a stronger conception than **NP**-hard. Then we give at the first time a simple bicriteria approximation algorithm for the uniform Min-Max EDP problem and the weighted Min-Sum EDP problem, with approximation factor $(O(\log k/\log\log k), O(\log n/\log\log n))$ and $(O(1), O(\log n/\log\log n))$ respectively, where the first parameter is the performance ratio, and the second parameter is the number of *congestion*, that is, the number of paths allowed per edge. Our algorithm is based on randomized rounding and runs in polynomial time. In fact, our algorithm shows that whenever the LP relaxation of the instance has a fractional solution, the instance has an approximate integer solution with congestion of at most $O(\log n/\log\log n)$. Since solving linear program tells whether the fractional feasible solution exists, our algorithm avoids to decide whether the instance is feasible, which is **NP**-hard.

2 Complexity of the Weighted Min-Sum DP Problem

In this section we prove that the weighted Min-Sum DP problem is $\mathbf{FP}^{\mathbf{NP}}$-compete in general graph. It is known that the problem Max Weight SAT is $\mathbf{FP}^{\mathbf{NP}}$-complete. In Max Weight SAT, we are given a CNF formula ϕ with positive integer weights for all clauses, and the problem is to find a truth assignment that satisfies a set of clauses with the most total weight. We shall reduce Max Weight SAT to Min-Sum DP by the reduction between function problems (refer to [8] for the definition).

Theorem 1. *Both in undirected and directed graph, the weighted Min-Sum VDP and weighted Min-Sum EDP problems are* $\mathbf{FP}^{\mathbf{NP}}$*-complete.*

Proof. First, we show that Min-Sum DP is in class $\mathbf{FP}^{\mathbf{NP}}$. It is easy to see that the decision version of this problem (that is, given a graph G, k source-sink pairs and length bound B, is there k vertex-disjoint (or edge-disjoint) paths with total length less than B which connect each source-sink pair?) is in **NP**. Therefore, by binary search, using a SAT oracle, we can find the minimum total length of such k vertex-disjoint (or edge-disjoint) paths. Then we can find these paths by removing every edge and making a query to the resulted decision problem one by one.

Next we reduce Max Weight SAT to Min-Sum EDP. Suppose that the instance of Max Weight SAT is I_1. We assign a gadget to each variable x_i. For example, if variable x_1 appears positively in clauses c_{i_1} and c_{i_2}, and appears negatively in clauses c_{j_1}, c_{j_2} and c_{j_3}, then we assign to it the gadget in Figure 1. For each

clause c_j, there is a source-sink pair (s_j, t_j). And for each source-sink pair there is a thick edge connecting s_j and t_j with weight w_j, called *survival edge*. All other edges have weight zero. In all these variable gadgets, the sources and sinks corresponding to the same clause are identical. This gives the graph G' (not including the survival edges).

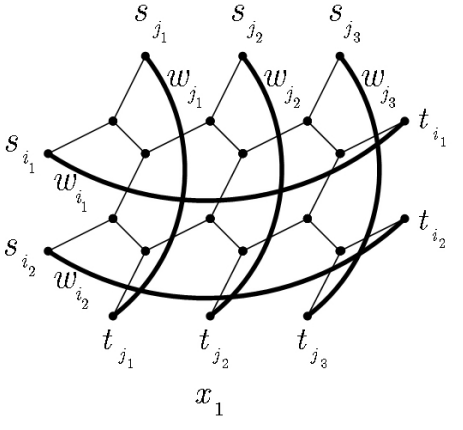


Fig. 1. Gadget for variable

Notice that in G' there may be some terminals with degree strictly great than 2, leading to that the path for some demand may traverse between different variable gadgets. Say that one such a terminal is v. By recursively applying on v the transformation in Figure 2, which is proposed in [7], we may decrease the degree of v to 2. The dashed edge (not belonging to the graph) in Figure 2, called *demand edge*, indicates that the two endpoints form a source-sink pair. Any path through v before the transformation must still pass through v after the transformation, if all the demands introduced in the transformation are satisfied. Since the degree of v is 2 in the final transformation and v is a terminal, this implies that the path through v that serves other demand is impossible. So for every source-sink pair corresponding to some clause, the path connects them is restricted within just one variable gadget. Denote by $\widetilde{G}$ the resulted graph after the transformation. Then $\widetilde{G}$ together with all survival edges form the final graph G. This gives the reduced instance I_2 of weighted Min-Sum EDP.

Then we conduct the second step of the reduction, that is, recovering an optimal solution to I_1 from an optimal solution to I_2. Denote by W_{tot} the total weight of clauses in formula ϕ. Suppose that O is an optimal solution to I_2 with value $OPT(I_2)$. For variable x_i, if O uses the horizontal zigzag paths in its corresponding gadget, then $x_i \triangleq 1$. If O uses the vertical zigzag paths, then $x_i \triangleq 0$. Otherwise x_i is assigned an arbitrary truth value. This gives a truth assignment τ for formula ϕ. Consider the source-sink pair (s_j, t_j) corresponding to clause c_j. If (s_j, t_j) is connected by zigzag path in some gadget of variable x_i contained in c_j, then (s_j, t_j) consumes zero weight in the optimal solution O.

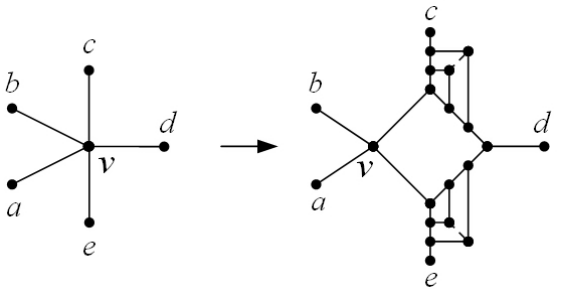


Fig. 2. Decreasing the degree of vertex v

Without loss of generality, suppose that (s_j, t_j) is connected by the horizontal zigzag path. By the construction of the gadget, variable x_i appears positively in clause c_j. Since x_i is assigned true, c_j is satisfied under τ. Conversely, if (s_j, t_j) is not connected by zigzag path in any gadget corresponding to the variables in c_j, then (s_j, t_j) is connected by the path weighted w_j. So, in each gadget corresponding to variables in c_j, there must be other source-sink pairs that use the zigzag paths, which are opposite to that of c_j. Then we know that all the literals in c_j are false, and hence c_j is unsatisfied under τ. So, the total weight of satisfied clauses of formula ϕ under truth assignment τ is $W_{tot} - OPT(I_2)$.

Furthermore, τ must be the optimal solution to I_1, otherwise O is not optimal. Notice that since every vertex has degree at most 3 in graph G, the edge-disjoint paths are identical to the vertex-disjoint paths in G. So far we have completed the proof in the case of undirected graph.

For the directed case, it is easy to assign a direction for every edge in the variable gadget. Since each source has in-degree 0 and each sink has out-degree 0, we do not need the transformation in Figure 2. Again, in the resulted graph the vertex-disjoint paths are identical to the edge-disjoint paths, and we also have $OPT(I_1) = W_{tot} - OPT(I_2)$. The theorem follows. □

3 Approximation Hardness of the Min-Sum DP and Min-Max DP Problems

We first prove the approximation hardness for the uniform Min-Sum DP problem in planar graph, and then extend the result to the uniform Min-Max DP problem. Our proof is based on the work in [7] about the **NP**-completeness of the EDP and VDP problems in undirected planar graph.

Theorem 2. *Even in planar graph, uniform Min-Sum DP can not be approximated within* $\Omega(m^{1-\epsilon})$ *for any constant* $\epsilon > 0$*, unless* $\mathbf{P} = \mathbf{NP}$.

Proof. Let us first focus on the vertex-disjoint case. We reduce the **NP**-complete problem planar E3SAT(3) to the uniform Min-Sum VDP problem by gap-introducing reduction. Suppose that the instance of planar E3SAT(3) is I_1. In the CNF formula ϕ of I_1, each clause contains exactly 3 literals, and each

variable appears at most 3 times. Suppose that there are M clauses and N variables in ϕ. Without loss of generality, assume that each variable has at most two positive occurrences and at most one negative occurrence (otherwise we flip all the occurrences of the variable).

We assign the variable gadget G_v in Figure 3 to each variable x_i, and assign the clause gadget G_c in Figure 3 to each clause c_j. Fix any constant $c > 1$. There is a path from source a_j to sink b_j in the clause gadget, called *survival path*, which is depicted by thick line in the gadget. The length of the survival path is $\lceil N^c \rceil$.

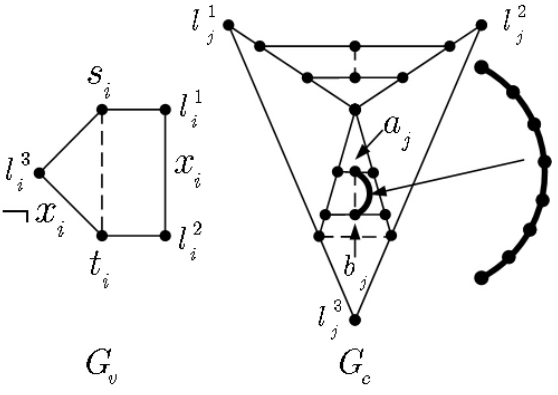


Fig. 3. Gadgets for variables and clauses

The dashed line in variable and clause gadgets means that its two endpoints is a source-sink pair. The variable gadget G_v has three literal vertices. The left vertex corresponds to the negative occurrence $\neg x_i$, and the right two vertices correspond to the two positive occurrences x_i. The clause gadget has three literal vertices, corresponding to the three literals in the clause respectively. Note that the literal vertices in G_v and G_c for the same literal are identical. This gives the whole graph $\widetilde{G}$ (not including all the survival paths and demand edges). Graph $\widetilde{G}$ together with all the survival paths form the graph G. Since ϕ is planar, it is clear that G is also planar.

Denote by $\widetilde{G}_c$ the gadget resulted by removing the survival path from G_c. One can verify the following properties hold: 1) $\widetilde{G}_c$ remains feasible if any two literal vertices are removed from $\widetilde{G}_c$, but turns infeasible if all the three literal vertices are removed; 2) If other source-sink pair not in $\widetilde{G}_c$ uses one of the paths between any two literal vertices of $\widetilde{G}_c$, $\widetilde{G}_c$ turns infeasible.

Suppose that there is a truth assignment satisfying ϕ. Then each source-sink pair (s_i, t_i) in G_v can be connected by the path corresponding to false literal. Since ϕ is satisfiable, each clause gadget has at least one literal vertex not traversed. By property 1, every $\widetilde{G}_c$ is feasible and no survival path is used.

Conversely, Suppose that $\widetilde{G}$ is feasible. Property 2 guarantees that the path connecting source-sink pair (s_i, t_i) in G_v has to be entirely in G_v. This gives a truth assignment for every variable. The fact that $\widetilde{G}_c$ is feasible shows that at least one literal vertex in $\widetilde{G}_c$ is not used by the paths belonging to variable gadgets. Since the literal corresponding to this vertex is true, we know that the clause corresponding to $\widetilde{G}_c$ is satisfiable. Thus ϕ is satisfiable. On the other hand, no matter whether ϕ is satisfiable, G is always feasible due to the survival paths. So in general the following claim holds.

Claim 1. The reduced graph G is feasible. Moreover, ϕ is satisfiable if and only if no any survival path is used.

It is easy to see that the maximum degree in G is 4. By applying the transformation in Figure 2, we can decrease the degree of non-terminal vertex to 3 and decrease the degree of terminal vertex to 2. Denote by I_2 the reduced instance of Min-Sum VDP.

Since ϕ is a CNF expression with each clause contains exactly 3 literals and each variable appears at most 3 times, we know that $N/3 \leq M \leq N$. By Claim 1, if ϕ is satisfiable, then

$$OPT(I_2) \leq c_1 M + c_2 N \leq c_0 N,$$

where $c_0 = c_1 + c_2$, $c_1 M$ is the total length of vertex-disjoint paths to connect the source-sink pairs in all clause gadgets, and $c_2 N$ is the total length of vertex-disjoint paths to connect the source-sink pairs in all vertex gadgets. We charge the length of paths used to connect source-sink pairs introduced in the transformation on the literal vertices into $c_2 N$ (by carefully counting, one can see that $c_1 \leq 55$ and $c_2 \leq 15$). If ϕ is unsatisfiable, then

$$OPT(I_2) > \lceil N^c \rceil \geq (\frac{1}{c_0} N^{c-1}) c_0 N.$$

Since $m = \Theta(N^{c+1})$, where m is the number of edges in G, we know that $N = \Theta(m^{1/(c+1)})$. This gives the gap $\frac{1}{c_0} N^{c-1} = \Theta(m^{1-2/(c+1)})$. Then, for any constant $\epsilon > 0$, we can set $c = 2/\epsilon - 1$. So we have approximation hardness $\Omega(m^{1-\epsilon})$ for uniform Min-Sum VDP in undirected planar graph.

Since in graph with maximum degree 2 for terminals and maximum degree 3 for other vertices the vertex-disjoint paths are identical to the edge-disjoint paths, we also obtain approximation hardness of $\Omega(m^{1-\epsilon})$ for uniform Min-Sum EDP in undirected planar graph. And by the well known transformation from undirected graph to directed graph for disjoin paths, the hardness result can be extended to directed case. The theorem follows. □

Theorem 3. *Even in planar graph, uniform Min-Max DP can not be approximated within $\Omega(m^{1-\epsilon})$ for any constant $\epsilon > 0$, unless* $\mathbf{P} = \mathbf{NP}$.

Proof. Similar as that in Theorem 2 and omitted. □

In addition, it is easy to see that since assuming $m = \Omega(n)$ does not lose generality, the uniform Min-Sum DP and uniform Min-Max DP problems also have approximation hardness $\Omega(n^{1-\epsilon})$ for any constant $\epsilon > 0$ even in planar graph.

4 The Randomized Approximation Algorithms

The complexity and approximation hardness of the Min-Sum DP and Min-Max DP problems show that efficient heuristic for the problems may be useful in practice. Several heuristics and experimental results for Min-Sum EDP are given in [1]. Other than experimental results, we give at the first time efficient approximation algorithm with guaranteed performance ratio for the problems, allowing that some constraints are slightly violated. Such approximation algorithm is often called *bicriteria approximation algorithm*. Although the problems are very hard in terms of complexity and approximation hardness, we can get $(O(\log k/\log\log k), O(\log n/\log\log n))$-factor approximation and $(O(1), O(\log n/\log\log n))$-factor approximation to the uniform Min-Max EDP problem and the weighted Min-Sum EDP problem respectively. Note that the congestion is only $O(\log n/\log\log n)$, a super-constant. Our analysis about the congestion of the algorithm is almost tight. Our algorithm is based on randomized rounding, which was introduced at the first time in [9].

4.1 LP Formulation and Algorithm for Uniform Min-Max EDP

The LP relaxation for uniform Min-Max EDP can be written as (1) to (6), denoted by $LP1$. The underlying graph can be either undirected or directed.

$$(LP1) \qquad \min L \tag{1}$$

s. t.

$$\sum_{p:\, e\in p} f(p) \le 1, \qquad \forall e \in E \tag{2}$$

$$\sum_{p\in P_i} f(p) = 1, \qquad \forall i \in [k] \tag{3}$$

$$\sum_{p\in P_i} |p| f(p) \le L, \qquad \forall i \in [k] \tag{4}$$

$$f(p) \ge 0, \qquad \forall p \in \mathcal{P} \tag{5}$$

$$L \ge 0 \tag{6}$$

In the linear program $LP1$, P_i denotes the set of all the possible paths from s_i to t_i. $\mathcal{P}$ is the union of all P_i. For each path $p \in P_i$, we define a variable $f(p) \in [0,1]$ which denotes the flow value on p for demand i. Consider the integral program version (that is, $f(p) \in \{0,1\}$ and $L \in \{1,2,\ldots,m\}$) of $LP1$. Then (2) specifies capacity constraint for every edge, (3) specifies the requirement for every demand, and (4) specifies the maximum length of flow path for every demand should not exceed L. The symbol $|p|$ in constraint (4) means the length of the path p.

Notice that although $LP1$ has exponential size, we can obtain a solution to $LP1$ in polynomial time, since there is a polynomial-size linear program which is equivalent to $LP1$. The polynomial-size linear program for uniform Min-Max EDP can be obtained by the method introduced in [3], and is omitted here due to the space restriction. We can first solve the equivalent polynomial-size LP, then get an optimal solution to $LP1$ using the flow decomposition method. Now we give the randomized algorithm for the uniform Min-Max EDP problem.

Algorithm $\mathcal{A}$

1. Find an optimal solution f to $LP1$. If $LP1$ has no solution, then output "Unfeasible".
2. **Repeat** the following for $r = \lceil 2 \log n \rceil$ times. In the jth time, **do**
3. $\quad S_j \leftarrow \emptyset$.
4. $\quad$ **For** each $i \in [k]$ **do**
5. $\quad\quad$ Choose $p \in P_i$ exclusively with probability $f(p)$.
6. $\quad\quad S_j \leftarrow S_j \cup \{p\}$.
7. $\quad$ **End**
8. **End**
9. **Return** the best S among $S_1, S_2, \ldots, S_r$.

Algorithm $\mathcal{A}$ first solves $LP1$ to get an optimal solution f, then rounds f to an integer solution S by picking a path $p \in P_i$ as casting a $|P_i|$-side die. The best solution in step 9 means the solution with first the minimum length of the longest path and then the least number of congestion. About algorithm $\mathcal{A}$ we have Theorem 4.

Theorem 4. *Algorithm $\mathcal{A}$ is a randomized bicriteria approximation algorithm for the uniform Min-Max EDP problem in general graph G. In polynomial time $\mathcal{A}$ outputs a solution S that connects all the source-sink pairs, and with high probability satisfies that (1) the value of S is at most $O(\log k / \log \log k)OPT(I)$; and (2) the congestion of S is at most $O(\log n / \log \log n)$.*

Proof. Since we pick a path $p \in P_i$ as casting a $|P_i|$-side die for each demand i, obviously S connects all the source-sink pairs. Then we turn to the property 1.

Fix any demand i. For every edge $e \in q$ for some path $q \in P_i$, define random variable Z_i^e to be 1 if there exist some chosen path $p \in P_i$ such that $e \in p$, and to be 0 otherwise. Then define random variable

$$Z_i = \sum_{\substack{e:\ e \in q \\ q \in P_i}} Z_i^e$$

to be the length of the path in S connecting the source-sink pair (s_i, t_i). Then we know that

$$\mathrm{E}[Z_i^e] = \sum_{\substack{p:\ e \in p \\ p \in P_i}} f(p)$$

and

$$\mathrm{E}[Z_i] = \sum_{\substack{e:\ e\in q\\ q\in P_i}} \mathrm{E}[Z_i^e] = \sum_{\substack{e:\ e\in q\\ q\in P_i}} \sum_{\substack{p:\ e\in p\\ p\in P_i}} f(p) = \sum_{p\in P_i} f(p)|p| \le L.$$

Set $\delta = 2\ln k \cdot L/(\mathrm{E}[Z_i]\ln\ln k) - 1$. Since $\mathrm{E}[Z_i] \le L$, we have that $\delta > 0$. By the Chernoff bound, we have

$$\begin{aligned}\Pr[Z_i > \frac{2\ln k}{\ln\ln k}L] &= \Pr[Z_i > (1+\delta)\mathrm{E}[Z_i]] < \Big(\frac{e}{1+\delta}\Big)^{(1+\delta)\mathrm{E}[Z_i]}\\ &= \Big(\frac{e\cdot \mathrm{E}[Z_i]\ln\ln k}{2\ln k\cdot L}\Big)^{2\ln k\cdot L/\ln\ln k} \le \Big(\frac{1}{4k}\Big)^{L} \le \frac{1}{4k}\end{aligned}$$

when k is sufficiently large, where the last inequality is due to

$$L \ge \sum_{p\in P_i} f(p)|p| \ge \min_{p\in P_i}\{|p|\}\sum_{p\in P_i} f(p) = \min_{p\in P_i}\{|p|\} \ge 1.$$

Define random variable Z to be the value of S, and denote by B_1 the event that $Z > (2\ln k/\ln\ln k)OPT(I)$. Then we know that

$$\begin{aligned}\Pr[B_1] &\le \Pr[Z > \frac{2\ln k}{\ln\ln k}L] = \Pr[\exists i, Z_i > \frac{2\ln k}{\ln\ln k}L]\\ &\le \sum_{i=1}^{k}\Pr[Z_i > \frac{2\ln k}{\ln\ln k}L] \le \frac{1}{4}.\end{aligned} \tag{7}$$

There may be some edges used for several times under the solution S. For the analysis of the property 2, fix any edge e. For each $p \in \mathcal{P}$ such that $e \in p$, define random variable X_e^p to be 1 if p is chosen and to be 0 otherwise. Then define random variable

$$X_e = \sum_{p:\ e\in p} X_e^p$$

to be the number of times that e is used in the procedure of randomized rounding for all demands. Then we know that $\mathrm{E}[X_e^p] = f(p)$ and

$$\mathrm{E}[X_e] = \sum_{p:\ e\in p}\mathrm{E}[X_e^p] = \sum_{p:\ e\in p} f(p) \le 1.$$

Set $\delta = 3\ln n/(\mathrm{E}[X_e]\ln\ln n) - 1$. Since $\mathrm{E}[X_e] \le 1$, we have that $\delta > 0$. Again by the Chernoff bound, we have

$$\begin{aligned}\Pr[X_e > \frac{3\ln n}{\ln\ln n}] &= \Pr[X_e > (1+\delta)\mathrm{E}[X_e]] < \Big(\frac{e}{1+\delta}\Big)^{(1+\delta)\mathrm{E}[X_e]}\\ &= \Big(\frac{e\cdot \mathrm{E}[X_e]\ln\ln n}{3\ln n}\Big)^{3\ln n/\ln\ln n} \le \Big(\frac{e\cdot\ln\ln n}{3\ln n}\Big)^{3\ln n/\ln\ln n} < \frac{1}{4n^2}\end{aligned}$$

when n is sufficiently large.

Denote by B_2 the event that there is an edge $e \in E$ such that $X_e > 3 \ln n / \ln \ln n$. Then we know

$$\Pr[B_2] \leq \sum_{e \in E} \Pr[X_e > \frac{3 \ln n}{\ln \ln n}] < \frac{1}{4}. \tag{8}$$

From (7) and (8), we have $\Pr[B_1 + B_2] < 1/2$. By repeating the procedure of randomized rounding for $\lceil 2 \log n \rceil$ times, we get a solution to the problem satisfying all the properties in the theorem with probability at least $1 - 1/n^2$. This completes the proof. □

4.2 LP Formulation and Algorithm for Weighted Min-Sum EDP

The LP relaxation for the weighted Min-Sum EDP problem, given by (9) to (12) and denoted by $LP2$, is similar to that of the uniform Min-Max EDP problem.

$$(LP2) \qquad \min \sum_{p \in \mathcal{P}} |p| f(p) \tag{9}$$

s. t.

$$\sum_{p:\ e \in p} f(p) \leq 1, \qquad \forall e \in E \tag{10}$$

$$\sum_{p \in P_i} f(p) = 1, \qquad \forall i \in [k] \tag{11}$$

$$f(p) \geq 0, \qquad \forall p \in \mathcal{P} \tag{12}$$

The symbol $|p|$ in objective function (9) means the total weight of edges in path p. Algorithm $\mathcal{A}$ also applies to the weighted Min-Sum EDP problem. For weighted Min-Sum EDP we have Theorem 5.

Theorem 5. *Algorithm $\mathcal{A}$ is a randomized bicriteria approximation algorithm for the weighted Min-Sum EDP problem in general graph G. In polynomial time $\mathcal{A}$ outputs a solution S that connects all the source-sink pairs, and with high probability satisfies that (1) the value of S is at most $O(1)OPT(I)$; and (2) the congestion of S is at most $O(\log n / \log \log n)$.*

Proof. We only prove the property 1. Define random variable Y to be the value of S. Then,

$$\mathrm{E}[Y] = \sum_{p \in \mathcal{P}} |p| \Pr[p \in S] = \sum_{p \in \mathcal{P}} |p| f(p) = OPT_f(I) \leq OPT(I),$$

where $OPT_f(I)$ denotes the value of the optimal fractional solution to $LP2$.

Notice that the expected value of S is always lower than $OPT(I)$. In order to show that algorithm $\mathcal{A}$ outputs a solution satisfying all the properties with high probability, we introduce a constant factor, say 4. Denote by B_1 the event that $Y > 4OPT(I)$. By Markov's inequality, we get

$$\Pr[B_1] = \Pr[Y > 4OPT(I)] \leq \frac{1}{4}.$$

Combining with the analysis for property 2 in Theorem 4 completes the proof. □

5 Discussion

We systematically study the complexity and approximation hardness of the Min-Sum DP and Min-Max DP problems. We also give randomized bicriteria approximation algorithm for the weighted Min-Sum DP and uniform Min-Max DP problems in general graph. We suspect that the weighted Min-Sum DP problem is $\mathbf{FP}^{\mathbf{NP}}$-complete even in planar graph. Or can one get an approximation algorithm for Min-Sum DP, or Min-Max DP, with slightly relaxed performance ratio but constant bound on the congestion? Both the problems are still interesting.

References

1. Brandes, U., Neyer, G., Wagner, D.: Edge-disjoint paths in planar graphs with short total length. Technical Report, in Konstanzer Schriften in Mathematik und Informatik, No. 19, Germany (1996)
2. Chekuri, C., Khanna, S.: Edge disjoint pahts revisited. In: Proc. of the 14th ACM-SIAM Symposium on Discrete Algorithms, pp. 628–637. ACM Press, New York (2003)
3. Garg, N., Vazirani, V., Yannakakis, M.: Approximate max-flow min-(multi)cut theorems and their applications. SIAM Journal on Computing 25, 235–251 (1996)
4. Kleinberg, J.: Approximation algorithms for disjoint paths problems. PhD Thesis, Department of EECS, MIT, Cambridge, MA (1996)
5. Kleinberg, J.: An Approximation Algorithm for the Disjoint Paths Problem in Even-Degree Planar Graphs. In: Proc. of the 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 627–636. IEEE Computer Society Press, Los Alamitos (2005)
6. Li, C., McCormick, T., Simchi-Levi, D.: The complexity of finding two disjoint paths with min-max objective function. Discrete Applied Mathematics 26, 105–115 (1990)
7. Middendorf, M., Pfeiffer, F.: On the complexity of the disjoint paths problem. Combinatorica 13(1), 97–107 (1993)
8. Papadimitriou, C.: Computational complexity. Addison-Wesley Publishing Company, Reading (1994)
9. Raghavan, P., Thompson, C.: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. Combinatorica 7(4), 365–374 (1987)
10. Suurballe, J., Tarjan, R.: A quick method for finding shortest pairs of disjoint paths. Networks 14, 325–336 (1984)
11. Yang, B., Zheng, S.Q., Lu, E.Y.: Finding two disjoint paths in a network with normalized α^+-min-sum objective function. In: Deng, X., Du, D.-Z. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 954–963. Springer, Heidelberg (2005)
12. Yang, B., Zheng, S.Q.: Finding min-sum disjoint shortest paths from a single source to all pairs of destinations. In: Cai, J.-Y., Cooper, S.B., Li, A. (eds.) TAMC 2006. LNCS, vol. 3959, pp. 206–216. Springer, Heidelberg (2006)

A Digital Watermarking Scheme Based on Singular Value Decomposition

Chin-Chen Chang[1,2], Yih-Shin Hu[2], and Chia-Chen Lin[3]

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung, 40724, Taiwan
[2] Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, 621, Taiwan
{ccc, hyhs91}@cs.ccu.edu.tw
[3] Department of Computer Science and Information Management, Providence University, Taichung 43301, Taiwan
mhlin3@pu.edu.tw

Abstract. Recently, digital watermarking techniques have been extensively applied on protecting rightful ownership of digital images. Some novel watermarking schemes which are based on SVD (Singular Value Decomposition) are being developed in some literatures [1, 15] nowadays. All of these SVD-based watermarking schemes have good embedding quality and high robustness, but they must depend on an original image or store some extra tables to extract watermarks. In this paper, we will propose an SVD-based watermarking scheme in order to reduce the load of the SVD-based watermarking system. Our proposed scheme can directly extract watermarks in the watermarked images without extra information. Furthermore, in our proposed scheme, we use a secure pseudo random number generator to decide embedded position in order to promote its security. According to the experimental results, our system can also maintain good embedding quality and high robustness.

Keywords: digital watermark, rightful ownership, singular value decomposition.

1 Introduction

Because of the huge rapid growth of the Internet, the traditional business has been expanded to deal in on the Internet nowadays. It is quite convenient for businessmen and consumers to sell or buy some commodities in this way. However, dealing in on the Internet also brings about some problems on information security, such as corruption, stealing, and assuming another's name to exchange. These problems usually can be solved by encryption. Besides, on the Internet, the transmission of digital multimedia, such as image, audio, and video can also settle the problem of protecting rightful ownership. Digital watermarking techniques are used to solve this problem. The digital watermark of rightful owner is hidden into the protected multimedia to avoid others' detection. Because the little distortional multimedia data is acceptable, most digital watermarking techniques exploited this property of multimedia data to hide watermark.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 82–93, 2007.

Generally, digital watermarking techniques must conform to some following requirements. (1) Invisibility: the difference between watermarked and original multimedia must not be noticed by naked eyes, namely, the quality of watermarked multimedia must be good. (2) Security: everyone except rightful one cannot detect watermark which is hidden in multimedia. Furthermore, watermarking algorithm must be public, namely, the security of the watermarking system should not build on attackers who do not know how the system works. (3) Efficiency: in order to be implemented efficiently, the watermarking algorithm must have good executing efficiency, and it does not need original multimedia to extract watermark. (4) Robustness: after the embedded multimedia is processed by digital signal processing (such as filtering, compressing, cropping, sharpening, blurring, etc.), the watermark still can be extracted when the quality of the multimedia is acceptable.

Digital watermarking schemes are usually classified into two categories: one is in spatial domain [9, 12, 13, 14, 17]. It directly changes digital data to hide watermark. The advantage of this kind is low computational complexity. But, it is weak to be against digital signal processing. Another is in frequency domain [2, 4, 7, 8, 10]. It must first transform digital data to be in frequency domain with transformation (such as Fast Fourier Transformation or Discrete Cosine Transformation or Discrete Wavelet Transformation, etc.). Then, it changes the coefficients which are obtained by transformation to hide watermarks. Finally, it inversely transforms these changed coefficients to be in spatial domain. Compared with the fist one, it needs more computation, but it can provide better robustness.

Besides, the SVD- (Singular Value Decomposition) based watermarking schemes are novel techniques [1, 15]. It is similar to frequency-domain-based scheme, and the SVD can be considered as a transformation. For example, Liu and Tan proposed an SVD-based watermarking scheme [15]. Although this scheme owns good embedding quality and high robustness, it needs to store three matrices whose sizes are equal to these of the original image to extract the watermark. In addition, Chandra also proposed two SVD-based watermarking schemes [1]. One is a global-based scheme, and another is a blocked-based scheme. Their robustness and embedding quality are good. However, Chandra's global-based scheme also needs to store three matrices to extract watermarks, while Chandra's block-based scheme needs the original images to extract the embedded watermarks. These schemes will add the load for the watermarking system. In next section, we will show the details of these related schemes.

The purpose of this paper is to propose a new SVD-based watermarking scheme. Our proposed method is block-based, and it does not need original image or storing additional matrices to extract the watermarks, that is, our proposed method can directly extract the watermarks from the watermarked images. Furthermore, it also maintains high robustness and good embedding quality.

The remainder of this paper is organized as follows. In Section 2, we will review the related works which we have just mentioned above, namely Liu and Tan's and Chandra's schemes. Besides, we will also shortly describe the SVD at the beginning of this section. In Section 3, we will present our proposed scheme. Then, in Section 4, the experimental results and discussions will be shown. Finally, the conclusions of this paper will be given in Section 5.

2 Related Works

In this section, we will briefly describe the SVD and then review these related schemes whose efficiency will be improved by our proposed method.

2.1 SVD

SVD is a linear algebra scheme, which is developed for a variety of applications, particularly in least-squares problems [5]. Recently, it has been also used in image processing applications including image compressing [18], image hiding [3], noise reducing [6, 11], and image watermarking [1, 15], because the singular values of an image do not change greatly when a small interference is added to an image.

Assume that the rank of an image matrix A whose size is $N \times N$ is r, and $r \leq N$. The SVD of A is defined as

$$A = U\ S\ V^T = [u_1, u_2, ..., u_N] \times \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & \sigma_N \end{bmatrix} \times [v_1, v_2, ..., v_N]^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T, \quad (1)$$

where U and V are $N \times N$ orthogonal matrices, S is an $N \times N$ diagonal matrix, u_i and v_i are U's and V's column vectors, respectively, and σ_i's are singular values satisfying $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r = \sigma_{r+1} = \cdots = \sigma_N = 0$. Furthermore, the columns of U are called left singular vectors, and the columns of V are called right singular vectors.

2.2 Liu and Tan's Watermarking Scheme

Liu and Tan proposed an SVD-based watermarking scheme [15]. Suppose that A represents the original image whose size is $N \times N$ and W is the watermark image whose size is $P \times P$. To start work, W must be resized to $N \times N$. In the embedding process, the SVD of A is computed to obtain U, S, and V. Then, a watermark, W, is added into S and compute the SVD of new matrix $S + aW$ to obtain U_W, S_W, and V_W, where a is a parameter which controls the quality of the watermarked image. Finally, the watermarked image, A_W, is obtained by $U\ S_W\ V^T$.

In watermark extraction, the system must save three matrices which are U_W, S, and V_W, and input the possible distorted image, $\overline{A_W}$. First, the SVD of $\overline{A_W}$ is computed to obtain $\overline{S_W}$. Then, the extracted watermark is obtained by $(U_W\ \overline{S_W}\ V_W^T - S) / a$.

2.3 Chandra's Watermarking Scheme

Chandra proposed two SVD-based watermarking schemes [1]. One is global-based scheme, and another is block-based scheme. These two schemes will be stated in

Subsection 2.3.1 and 2.3.2, respectively. Suppose that A is the original image whose size is N × N, and W is the watermark whose size is P × P.

2.3.1 Global-Based Scheme

For the embedding process, *A* and *W* must first be computed their SVD which are $U\ S_a\ V^T$ and $U_w\ S_w\ V_w^T$, respectively. The diagonal elements of S_a and S_w are represented by $\sigma_a = [\ \sigma_{a1}, \sigma_{a2}, \ldots, \sigma_{aN}\ ]$ and $\sigma_w = [\ \sigma_{w1}, \sigma_{w2}, \ldots, \sigma_{wN}\ ]$. Then, the watermark is hidden into the singular values of *A* by the following formula,

$$\sigma_{bi} = \sigma_{ai} + \alpha\sigma_{wi}, \tag{2}$$

where α is a parameter which is chosen to maintain the quality of the watermarked image. Let S_b be the diagonal matrix whose diagonal elements are represented by σ_{bi}. Finally, the watermarked image is obtained by $U\ S_b\ V^T$.

For the extracting process, the extracting algorithm needs the original image's S_a and the watermark's U_w and V_w. First, the computed SVD of the watermarked image is $\overline{U}\ \overline{S_b}\ \overline{V}^T$. Then, the diagonal matrix, $\overline{S_w}$, of the extracted watermark is computed as follows.

$$\overline{S_w} = (\ \overline{S_b} - S_a\ ) / \alpha. \tag{3}$$

Finally, the extracted watermark is obtained by $U_w\ \overline{S_w}\ V_w^T$.

2.3.2 Block-Based Scheme

The original image is divided into non-overlapping blocks, A_X, whose size is $k \times k$. To add security, the watermark must be permuted by using a secret key before embedding. For the embedding process, the SVD of each block is computed first. Let S_X represent the diagonal matrix for A_X. Then, embed the watermark bit, W_X, into the largest singular value, σ_{a1}^X, of the block as follows.

$$\sigma_{b1}^X = \sigma_{a1}^X + \alpha\ W_X \tag{4}$$

Here, α is a parameter which is chosen to maintain the quality of the watermarked image, and σ_{b1}^X is the largest singular value of the watermarked block. Finally, reconstruct the watermarked block and image.

The extracting process needs the original image to extract the watermark according to Equation (4). First, the watermarked image which is processed by image processing scheme is also divided into non-overlapping block, $\overline{B_X}$, whose size is $k \times k$. Then, compute the SVD of each watermarked block to obtain every largest singular value, σw_{b1}^X. The watermark bit is obtained by $(\ \sigma w_{b1}^X - \sigma_{a1}^X\ ) / \alpha$ where σ_{a1}^X is computed from the original block. Finally, the watermark is re-permuted by using the original secret key.

3 The Proposed Scheme

Our proposed watermarking scheme for binary logo is based on the SVD of image blocks. Suppose that the sizes of an image and a binary watermark are $N \times N$ and $P \times P$, respectively. This image is divided into $(N / 4) \times (N / 4)$ non-overlapping blocks whose size is 4×4.

To start work, we exploit a one-way hash function [9] based on Rabin's scheme [18] to decide the embedding block's position. First, choose two large prime number, p and q. Let Z = p × q, where p and q are secret and Z is public. Then, randomly choose two values, k_1 and k_2, as secret seeds to compute embedding block's position as follows.

$$X_i = X_{i-1}^2 \bmod Z,\ X_0 = k_1^2 \bmod Z, \tag{5}$$

$$Y_i = Y_{i-1}^2 \bmod Z,\ Y_0 = k_2^2 \bmod Z, \tag{6}$$

$$x_i = X_i \bmod (N / 4), \text{ and} \tag{7}$$

$$y_i = Y_i \bmod (N / 4) \tag{8}$$

(x_i, y_i) is the embedding block's position of the *i*th bit in the bit stream of the watermark. In addition, two arbitrary positions must be different. The above process can be considered as a secure pseudo random number generator.

Then, the algorithms of the watermark embedding and extracting will be described in the following two subsections.

3.1 Embedding Process

According to the previous statement, there are $(N / 4) \times (N / 4)$ non-overlapping blocks. The size is 4×4 for each block. And there are $P \times P$ binary values that need to be hidden. In order to achieve high robustness, every binary value of the watermark should be hidden in three different blocks. The reason why we do so will be explained in next subsection. Therefore, the bit stream of the watermark must be copied three times to get $P \times P \times 3$ binary values, and there must be $P \times P \times 3$ different positions which are generated by pseudo random number generator, that is, $1 \le i \le P \times P \times 3$. Assume that B_j, where $j = x_i \times (N/4) + y_i$ and $1 \le j \le (N / 4) \times (N / 4)$, is the corresponding block of (x_i, y_i) in this image and W_i is the *i*th binary value in the bit stream of three original watermarks. So, $(N / 4) \times (N / 4)$ must be larger than $P \times P \times 3$. The algorithm of our block-based SVD watermarking scheme is as follows.

[The Embedding Algorithm]

Input: Coordinate (x_i, y_i), block B_j, and watermark bit W_i

Output: a watermarked image

Step 1: Let i = 1.

Step 2: Compute the SVD of the corresponding block, B_j, of (x_i, y_i). Obtain three matrices which are U_j, S_j, and V_j. Assume that $S_j = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 \\ 0 & 0 & 0 & \sigma_4 \end{bmatrix}_j$

Step 3: Let σ_3 be equal to σ_2. Obtain $S_j^{'}$ which is $\begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \sigma_3^{'} & 0 \\ 0 & 0 & 0 & \sigma_4 \end{bmatrix}_j$.

Step 4: Let σ_2 be equal to $\sigma_2 + \delta \times W_i$, where δ is a constant. If $\sigma_1 < \sigma_2 + \delta \times W_i$, $\sigma_1 = \sigma_2 + \delta \times W_i$. Obtain $S_j^{''}$ which is $\begin{bmatrix} \sigma_1^{'} & 0 & 0 & 0 \\ 0 & \sigma_2^{'} & 0 & 0 \\ 0 & 0 & \sigma_3^{'} & 0 \\ 0 & 0 & 0 & \sigma_4 \end{bmatrix}_j$.

Step 5: Reconstruct the watermarked block, BW_j, which is equal to $U_j \times S_j^{''} \times V_j^T$.

Step 6: Let $i = i + 1$. Go to Step 2 until $i = P \times P \times 3$.

According to our proposed embedding algorithm, the number of the singular values that are modified is at most 3. In addition, the largest singular value, σ_1, is usually greatly larger than σ_2 for a block in an image. So, the condition of $\sigma_1 < \sigma_2 + \delta \times W_i$ seldom occurs. In our proposed algorithm, the difference between the watermarked block and the original block is small. The reason is that we almost only modify the second and the third singular values when $W_i = 1$ and only modify the third singular value when $W_i = 0$.

By the experimental results in Section 4, the quality of the watermarked image is still good when δ is set to 20. The quality of the watermarked image is measured by *PSNR* (Peak Signal-to-Noise Ratio). And the formula of *PSNR* is illustrated as follows.

$$PSNR = 10 \times \log_{10} \frac{255^2}{MSE}, \tag{9}$$

where *MSE* is the mean square error between the original and the corresponding watermarked pixel values. In general, it is acceptable that PSNR is greater than 30dB.

3.2 Extracting Process

In extracting process, we can directly extract watermark from the watermarked image, and extract the watermark accurately if the watermarked image is not modified by

some image processing techniques. To begin the extracting process, we must give original secret key which is (k_1, k_2) to get (x_i, y_i) by Equations (5) to (8), and assume that the corresponding block of (x_i, y_i) is BW_j' which is possibly distorted from BW_j in the watermarked image.

In our proposed embedding algorithm, the difference between the second and the third singular values is set to 0 or δ. So, if the difference between the second and the third singular values that are computed from the watermarked image is larger than a threshold, the bit of the watermark will be considered as 1, otherwise it will be considered as 0. According to the results of the experiments, it is better that this threshold is set to δ / 2. Furthermore, because of three times' embedding watermark, the watermark can be simply determined by these three extracted watermarks. We can determine .the binary value of the extracted watermark by a heavy vote in "1" or "0" from these three extracted watermarks. So, the correction rate of the extracted watermark will be promoted. The correction rate of the extracted watermark is measured by BCR (Bit Correction Ratio). The formula of BCR is demonstrated as follows.

$$BCR = \frac{\sum_{i=1}^{P\times P} \overline{W_i \oplus W_i'}}{P \times P} \times 100\% \tag{10}$$

where W_i is the *i*th binary value in the bit stream of the original watermark, W_i' is the *i*th binary value in the bit stream of the extracted watermark, and $\oplus$ represents an operator of exclusive-OR. Obviously, when *BCR* is larger, the similarity between original and extracted watermark is higher.

The algorithm of watermark extraction is stated as follows.

[**The Extracting Algorithm**]

Input: (x_i, y_i) and BW_j'

Output: the watermark

Step 1: Let $i = 1$.

Step 2: Compute the SVD of the corresponding block, BW_j', of (x_i, y_i). Obtain three matrices which are UW_j, SW_j, and VW_j. Assume that SW_j = $\begin{bmatrix} \sigma w_1 & 0 & 0 & 0 \\ 0 & \sigma w_2 & 0 & 0 \\ 0 & 0 & \sigma w_3 & 0 \\ 0 & 0 & 0 & \sigma w_4 \end{bmatrix}_j$.

Step 3: If $\sigma w_2 - \sigma w_3 > \delta / 2$, let $WT_i = 1$; Otherwise, let $WT_i = 0$.

Step 4: Let $i = i + 1$. Go to Step 2 until $i = P \times P \times 3$.

Step 5: Let $i = 1$.

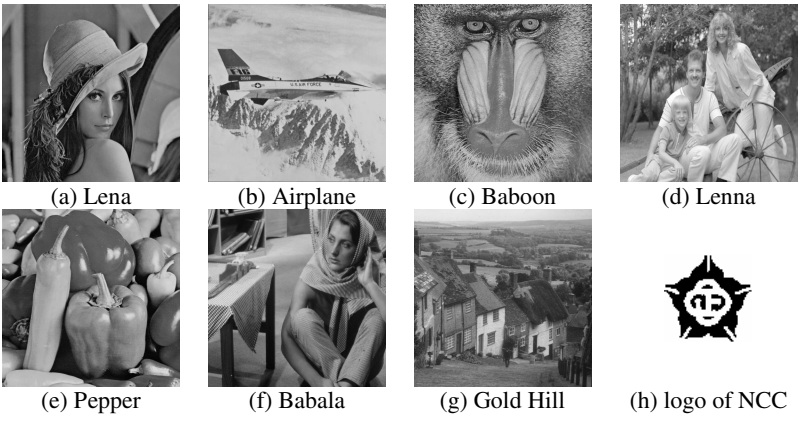
Step 6: If $WT_i + WT_{i+p\times p} + WT_{i+p\times p\times 2} \geq 2$, let $W_i^{'} = 1$; Otherwise, let $W_i^{'} = 0$.

Step 7: Let $i = i + 1$. Go to Step 6 until $i = P \times P$.

Because the singular values of a block in an image do not change greatly when a small interference is added to an image, by using the extraction algorithm, the watermark can still be extracted from the watermarked image that is processed by some image processing schemes. In addition, the robustness of our proposed watermarking scheme will be shown in the experimental results in the next section.

4 Experimental Result and Discussions

The experiments were achieved on seven gray-level images whose size is 512 × 512 and a binary watermark image whose size is 64 × 64. These images are shown in Fig. 1. All of our experiments were executed on a personal computer with 256 KB RAM by using a Pentium III 1 GHz CPU. Photoshop 7.0 was the tool of image processing exploited in our experiments.



(a) Lena (b) Airplane (c) Baboon (d) Lenna

(e) Pepper (f) Babala (g) Gold Hill (h) logo of NCC

Fig. 1. Seven test images and a binary watermark

This watermark image was embedded into these seven gray-level images by using our proposed watermarking scheme. The constant, δ, in our algorithm was set to 20. Obviously, if δ is larger, PSNR of the watermarked image will be smaller, and the robustness of this watermarking scheme will be higher. Table 1 shows the embedding and extracting results of every original image. According to Table 1, we can find that the quality of the watermarked image is acceptable and the BCR of our proposed scheme is 100%.

Table 1. The embedding and extracting results of every original image

Result / Original Image	*PSNR* of the watermarked image	*BCR* of the watermark
Lena	35.37 dB	100%
Airplane	35.44 dB	100%
Baboon	31.34 dB	100%
Lenna	35.24 dB	100%
Pepper	35.94 dB	100%
Babala	32.19 dB	100%
Gold Hill	35.50 dB	100%

The robustness of our proposed watermarking scheme was tested under five image processing techniques which are JPEG compressing, sharpening, blurring, cropping, and adding noise, respectively. Table 2 shows the robustness test of our method against JPEG compressing for every watermarked image. We can find that the watermark is still extracted correctly from JPEG compressed image.

Table 2. The compression and extracting results of every watermarked image after JPEG compression

Result / Watermarked image	*PSNR* of the JPEG compressed image	*BCR* of the watermark
Lena	35.87 dB	87.92%
Airplane	36.82 dB	90.16%
Baboon	35.08 dB	86.45%
Lenna	37.86 dB	94.07%
Pepper	36.81 dB	91.82%
Babala	35.73 dB	88.82%
Gold Hill	35.27 dB	88.77%

Table 3 shows the results of the robustness test against the sharpening process for every watermarked image. So, we can find that our proposed scheme can still detect correct watermark after image sharpening. Table 4 displays the experimental results of the robustness test against image blurring for every watermarked image. From Table 4, we know that our proposed method can also resist the distortion of burring process.

Table 3. The sharpening and extracting results of every watermarked image

Result / Watermarked image	*PSNR* of the sharpened image	*BCR* of the watermark
Lena	29.93 dB	95.73%
Airplane	30.46 dB	95.83%
Baboon	23.51 dB	81.49%
Lenna	31.91 dB	90.11%
Pepper	31.43 dB	97.17%
Babala	24.29 dB	88.13%
Gold Hill	28.44 dB	89.84%

Table 4. The blurring and extracting results of every watermarked image

Result / Watermarked image	*PSNR* of the blurred image	*BCR* of the watermark
Lena	35.60 dB	92.41%
Airplane	36.39 dB	94.68%
Baboon	29.23 dB	84.42%
Lenna	37.76 dB	95.39%
Pepper	37.19 dB	93.48%
Babala	29.43 dB	86.99%
Gold Hill	34.40 dB	86.82%

Table 5 displays the results of the robustness to image cropping for every watermarked image. From Table 5, we can clearly see that the BCR of every extracted watermark is the same.

Table 5. The cropping and extracting results of every watermarked image

Result / Watermarked image	*PSNR* of the cropped image	*BCR* of the watermark
Lena	10.11 dB	90.19%
Airplane	14.65 dB	90.19%
Baboon	10.95 dB	90.19%
Lenna	11.19 dB	90.19%
Pepper	10.83 dB	90.19%
Babala	11.27 dB	90.19%
Gold Hill	11.76 dB	90.19%

Table 6. The noisy and extracting results of every watermarked image

Result / Watermarked image	*PSNR* of the noisy image	*BCR* of watermark
Lena	32.54 dB	89.28%
Airplane	32.48 dB	87.96%
Baboon	32.50 dB	83.15%
Lenna	32.53 dB	88.01%
Pepper	32.49 dB	89.04%
Babala	32.50 dB	86.65%
Gold Hill	32.50 dB	86.67%

Furthermore, we compare our proposed scheme with a spatial-domain-based scheme [9] and three SVD-based schemes [1,15]. The comparison results are shown in Table 7. The scheme of Hwang et al. [9] is implemented in spatial domain. Its embedding quality is very high, while its computational complexity is quite low. Though the watermark can still be extracted from the watermarked image after randomly changing every pixel's least significant 3 bits, it cannot be extracted clearly from the watermarked image after other image processing attacks.

Table 7. Comparison between our proposed scheme and other schemes

Scheme / Items	Hwang et al. [9]	Liu and Tan [15]	Chandra (Global-based) [1]	Chandra (Block-based) [1]	Proposed scheme
Processing domain	Spatial domain	SVD domain	SVD domain	SVD domain	SVD domain
Extracting watermarks with original image	No	No	No	Yes	No
Storing some matrices to extract watermark	No	Yes	Yes	No	No
Robustness	Low	High	High	High	High
Embedding quality	Very high	High	High	High	High

Both these four SVD-based watermarking schemes and our proposed one all own good embedding quality and high robustness. Liu and Tan's watermarking scheme is global based [15]. For its embedding process, the watermark must be resized to the size of the original image for the computation of matrices. For its extracting process, the system must store three matrices whose size is the same as that of original image. Chandra's global-based watermarking scheme [1] also needs to store three matrices to extract watermark. Finally, Chandra's block-based watermarking scheme must use the original image to extract watermark. As is shown, these three schemes do not make their watermarking systems own good efficiency. However, our proposed scheme can extract watermarks correctly without any additional information including the original image. Therefore, our proposed scheme has better efficiency.

5 Conclusions

In this paper, we proposed a watermarking scheme based on SVD for binary logo. Our proposed scheme has good efficiency because it can directly extract watermarks from the watermarked image without storing extra matrices and the original image. In addition, we also exploited a secure pseudo random number generator to decide the embedded block's position in order to enhance security of our watermarking system.
According to our experimental results, our proposed scheme still maintains good embedding quality and high robustness against some image processing distortions. The *PSNR* values of the watermarked images are all greater than 31 dB for every original image. And the embedding quality of our proposed scheme is good because the difference between the original and watermarked images is unnoticeable in vision. In addition, even though the watermarked image is modified as a little distorted image, our system can still extract the watermark correctly. Therefore, our watermarking system is very suitable for the protection of rightful ownership of digital images.

References

1. Chandra, D.V.S.: Digital Image Watermarking Using Singular Value Decomposition. In: MWSCAS 2002. Proceedings of 45th Midwest Symposium on Circuits and Systems, August 4-7, 2002, vol. 3, pp. 264–267 (2002)
2. Chu, W.C.: DCT-Based Image Watermarking Using Subsampling. IEEE Transactions on Multimedia 5(1), 34–38 (2003)
3. Chung, K.L., Shen, C.H., Chang, L.C.: A Novel SVD- and VQ-Based Image Hiding Scheme. Pattern Recognition Letters 22(9), 1051–1058 (2001)
4. Cox, I.J., Kilian, J., Leighton, F.T., Shamoon, T.: Secure Spread Spectrum Watermarking for Multimedia. IEEE Transactions on Image Processing 6(12), 1673–1687 (1997)
5. Golub, G.H., Reinsch, C.: Singular Value Decomposition and Least Squares Solutions. Numerische Mathematik 14, 403–420 (1970)
6. Hou, Z.: Adaptive Singular Value Decomposition in Wavelet Domain for Image Denoising. Pattern Recognition 36(8), 1747–1763 (2003)
7. Hsieh, M.S., Tseng, D.C., Huang, Y.H.: Hiding Digital Watermarks Using Multiresolution Wavelet Transform. IEEE Transactions on Industrial Electronics 48(5), 875–882 (2001)
8. Hsu, C.T., Wu, J.L.: Hidden Digital Watermarks in Images. IEEE Transactions on Image Processing 8(1), 58–68 (1999)
9. Hwang, M.S., Chang, C.C., Hwang, K.F.: A Watermarking Technique Based on One-Way Hash Functions. IEEE Transactions on Consumer Electronics 45(2), 286–294 (1999)
10. Iwata, M., Shiozaki, A.: Watermarking Method for Embedding Index Data into Images Utilizing Features of Wavelet Transform. IEICE Transactions on Fundamentals E84-A(7), 1772–1778 (2001)
11. Konstantinides, K., Natarajan, B., Yovanof, G.S.: Noise Estimation and Filtering Using Blocked-Based Singular Value Decomposition. IEEE Transactions on Image Processing 10(3), 479–483 (1997)
12. Kutter, M., Jordan, F., Bossen, F.: Digital Watermarking of Color Images Using Amplitude Modulation. Journal of Electronic Imaging 7(2), 326–332 (1998)
13. Langelaar, G.C., van der Lubbe, J.C.A., Lagendijk, R.L.: Robust Labeling Methods for Copy Protection of Images. In: Proceedings of SPIE Electronic Imaging '97, Storage and Retrieval for Image and Video Database V, February 1997, pp. 298–309. San Jose, CA (1997)
14. Lee, C.H., Lee, Y.K.: An Adaptive Digital Watermarking Technique for Copyright Protection. IEEE Transactions on Consumer Electronics 45(4), 1005–1015 (1999)
15. Liu, R., Tan, T.: An SVD-Based Watermarking Scheme for Protecting Rightful Ownership. IEEE Transactions on Multimedia 4(1), 121–128 (2002)
16. Rabin, M.O.: Digitalized Signatures and Public-Key Functions as Intractable as Factorization. Technical Report MIT/LCS/TR212, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA (January 1979)
17. Voyatzis, G., Pitas, I.: Embedding Robust Watermarks by Chaotic Mixing. In: DSP'97. Proceedings of 13th International Conference on Digital Signal Processing, vol. 1, pp. 213–216 (1997)
18. Yang, J.F., Lu, C.L.: Combined Techniques of Singular Value Decomposition and Vector Quantization for Image Coding. IEEE Transactions on Image Processing 4(8), 1141–1146 (1995)

A New $(t, n)-$Threshold Scheme Based on Difference Equations

Chao-Wen Chan[1] and Chin-Chen Chang[2]

[1] Graduate School of Computer Science and Information Technology
National Taichung Institute of Technology, 129 Sec. 3, Sanmin Rd., Taichung 404, Taiwan R.O.C
ccwen@ntit.edu.tw
[2] Department of Information Engineering and Computer Science
Feng Chia University, No. 100 Wenhwa Rd. Seatwen, Taichung, Taiwan 40724, R.O.C
ccc@cs.ccu.edu.tw

Abstract. In the paper, we propose three threshold secret sharing schemes that are based on difference equations. The first scheme is a $(t, n)-$threshold scheme which is an ideal perfect secure. The other two schemes add the restricted order structure to the set of shadows and the access structure of secret sharing policy. The basis of the access structure of one scheme allows that only subsets that contain consecutive shadows can compute the broken secret, but no other subset of shadows can do so. The basis of the access structure of the other scheme allows that shadow subsets contain an imperfect consecutive shadow subset that has one gap of size 1, and can compute the original secret.

1 Introduction

Suppose that a company has a vault in which the company stores the business secrets. All business operations require the secrets that are stored in the vault. The company employs n employees and it does not trust any single employee to access the secrets. However, the company must perform the everyday business operations. Thus, the company requires a secret sharing system to realize its secret sharing policy and control the retrieval of the business secret by these employees.

Assume that the secret sharing policy established by the company is any t employees can retrieve the business secret but no t' employees can do so where $t' < t$. In such a situation, the company may apply the Shamir's $(t, n)-$threshold scheme ([1],[2]) to implement the secret sharing system.

In general, employees take vacations but the company can control how many employees will be one of rotation at any given time. Thus, we may assume that the set $\{1, 2, \cdots, n\}$ denotes the order structure of vacations employees being taking. We also suppose that everyday business operations of the company are performed by at least t consecutive employees who are on duty. The company therefore establishes the secret sharing policy, and every t consecutive employees can retrieve the business secret. Because the Shamir's $(t, n)-$threshold scheme

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 94–106, 2007.

doesn't impose order structure into its access structure, such a policy is not easily implemented. A secret sharing scheme having the access structure we will call it a $(0, t, n)-$threshold scheme.

Another situation may happen in the company. Some of the t consecutive employees may need to make business trips thus causing more absent employees. If this happens, the business secret is not likely be retrieved from the vault to perform everyday business operations. Thus the company may allow any t employees of $t + i$ consecutive employees retrieving the secret. A secret sharing scheme with such access structure is called an $(i, t, n)-$threshold scheme. With the same reason as before, such a secret sharing policy is not easily implemented using the Shamir's $(t, n)-$threshold scheme.

In Section II, we review some important notions relative to secret sharing schemes and some important results of them. We introduce the notions of difference equations and present our main idea in Section III. The proposed schemes based on difference equations are presented in Section IV. In Section V, we will analyze the securities of the proposed schemes. Section VI is our conclusions to the research.

2 Preliminary

In a practical view, a secret sharing system is a method for breaking a secret into a set of shadows such that any legal subset of shadows can recover the original secret but no single shadow can recover it completely. Let $\mathcal{K}$ be the key space. $K \in \mathcal{K}$ is the secret to be broken into a set of shadows S_K. If $S_K \subset \mathcal{K}$, we call the corresponding secret sharing system ideal. The legal subsets of shadows are defined by the access structure $\Gamma \subset 2^{S_K}$ of the secret sharing system. That is, Γ defines the secret sharing policy of a secret sharing system. We therefore need to study the description ability of Γ. A secret sharing scheme for breaking K into $\mathcal{S}_K$ is designed to perform the access structure Γ.

In general, Γ should satisfy the monotone property:

$$\text{if } A \in \Gamma \text{ and } A \subseteq B, \text{ then } B \in \Gamma.$$

If Γ is a monotone access structure, it can be depicted by a subset Γ_0 of 2^{S_K} called the basis of Γ. $A \in \Gamma$ is called a minimal legal subset if any proper subset of A is not a member of Γ. The basis Γ_0 of Γ is defined by the set of minimal subsets of Γ. Because Γ is monotone, we have

$$\Gamma = closure(\Gamma_0) = \{B \in 2^{S_K} : A \subseteq B, A \in \Gamma_0\}.$$

If $\Gamma_0 = \{A \in 2^{S_K} : |A| = t\}$, we say that Γ is a threshold access structure. A threshold scheme therefore is a counting scheme which decides whether the number of shadows is equal to or larger than the threshold. In addition, we require that $|S_K| = n$ for all $K \in \mathcal{K}$, Γ is called $(t, n)-$threshold access structure. A well-known $(t, n)-$threshold scheme is Shamir's $(t, n)-$threshold scheme which is based on the theory of interpolating polynomials over $\mathbb{Z}_p$, where p is a prime

number. Because a polynomial of degree $t-1$ can be uniquely determined by exactly t distinct points of itself and $t-1$ points compute nothing about the original polynomial, Shamir's $(t,n)-$threshold scheme is a perfect secret sharing scheme.

A perfect secret sharing scheme realizes an access structure Γ if any $A \in \Gamma$ contains sufficient information to compute K and any $B \notin \Gamma$ computes nothing about K.

Because S_K will be distributed to participants, S_K therefore may have order structure. That is, $S_K = \{y_1, y_2, \cdots, y_n\}$ for some positive integer n. For the order structure of shadows, we may define an access structure by the language of order structure. For example, the basis Γ_0 is the set of all t consecutive shadows. That is,

$$\Gamma_0 = \{\{y_i, y_{i+1}, \cdots, y_{i+t-1}\} : \text{ all possible integer } i\}.$$

Shamir's $(t,n)-$threshold scheme realizes $(t,n)-$threshold access structure but does not easily create access structure with an order structure description. In this paper, we propose threshold schemes based on difference equations whose access structure can contain some order structure description.

3 Main Idea

In this section, we investigate the "initial value problem" of the linear difference equations over a finite field. Let p represent a prime number and $t \in \mathbb{Z}_p$. Consider the following linear difference equation of order t over $\mathbb{Z}_p$:

$$a_t f(x+t) + a_{t-1} f(x+t-1) + \cdots + a_0 f(x) \equiv b \pmod{p}, \tag{1}$$

where $b, a_0, \cdots, a_t \in \mathbb{Z}_p$ and $a_t \in \mathbb{Z}_p^*$. Let $x_0, y_0, y_1, \cdots, y_{t-1} \in \mathbb{Z}_p$. Suppose that $y_i \equiv f(x_0+i) \pmod{p}$, for all, $i = 0, 1, \cdots, t-1$. Then we have

$$f(x_0+t) \equiv (b - a_{t-1}y_{t-1} - a_{t-2}y_{t-2} - \cdots - a_0 y_0)a_t^{-1} \pmod{p}. \tag{2}$$

We can therefore compute the value of $f(x)$ for all $x \in \mathbb{Z}_p$ by Equation (2). In addition, if $a_0 \in \mathbb{Z}_p^*$, we have

$$f(x_0) \equiv (b - a_t y_t - a_{t-1}y_{t-1} - \cdots - a_1 y_1)a_0^{-1} \pmod{p}. \tag{3}$$

By Equations (2) and (3), we have the following theorem about the "initial value problem" of linear difference equation over $\mathbb{Z}_p$:

Theorem 3.1. *Let p be a prime number and $t < p$ be a positive integer. Assume that $a_0, a_1, \cdots, a_t \in \mathbb{Z}_p$ and $a_t \in \mathbb{Z}_p^*$. Then for any $x_0 \in \mathbb{Z}_p$ and any $y_0, y_1, \cdots, y_{t-1} \in \mathbb{Z}_p$, there exists a unique function $f(x)$ over $\mathbb{Z}_p$ that satisfies Equation (1) for $x \in \mathbb{Z}_p$ and $f(x_0+i) \equiv y_i \pmod{p}$, for all,$i = 0, 1, \cdots, t-1$.*

Corollary 3.2. *Suppose that $a_i \neq 0$ for all i. Let g be a solution of Equation (1). Then, for every x_0, any t points of the set $\{(x_0+i, g(x_0+i)) | 0 \leq i \leq t\}$ can recover g with Equation (1).*

It is easy to see that fo r any $b, a_0, a_1, \cdots, a_t \in \mathbb{Z}_p$ and $a_t \in \mathbb{Z}_p^*$, Equation (1) defines a family of functions over $\mathbb{Z}_p$. To characterize the general solution of Equation (1), we associate it with a homogeneous equation by letting $b \equiv 0 \pmod{p}$ as follows:

$$a_t g(x+t) + a_{x-1} g(x+t-1) + \cdots + a_0 g(x) \equiv 0 \pmod{p}. \tag{4}$$

Then we have the following theorem about the properties of Equations (1) and (4)'s solutions:

Theorem 3.3. *Below, all functions are defined over* $\mathbb{Z}_p$.

1. *If* $g_1(x)$ *and* $g_2(x)$ *solve Equation (4), then so does* $\alpha g_1(x) + \beta g_2(x)$ *for all* $\alpha, \beta \in \mathbb{Z}_p$.
2. *If* $f(x)$ *solves Equation (1) and* $g(x)$ *solves Equation (4), then* $f(x) + g(x)$ *solves Equation (1).*
3. *If* $f_1(x)$ *and* $f_2(x)$ *solve Equation (1), then* $f_1(x) - f_2(x)$ *solve Equation (4).*

Part 1 of Theorem 3.3 points out that the set of solutions for Equation (4) is a subspace of the function space of $\mathbb{Z}_p$. By the fact and Theorem 3.1, we may guess the dimension of the solution subspace of Equation (4) is t. To find the dimension and basis of the subspace, we need the following operators:

Definition 3.4. *Below, all functions are defined over the number field* $\mathbb{Z}_p$.

1. *The identity operator* I *is defined by the equation*

$$I(g(x)) \equiv g(x) \pmod{p}.$$

2. *The shift operator* E *is defined by the equation*

$$E(g(x)) \equiv g(x+1) \pmod{p}.$$

3. *The difference operator* Δ *is defined by the equation*

$$\Delta(g(x)) \equiv g(x+1) - g(x) \pmod{p}.$$

It is easy to see that the operators, I, E, and Δ satisfy the following relations:

$$\Delta = E - I.$$

or

$$E = \Delta + I.$$

Based on function composition and mathematic induction, we have:

Definition 3.5. *Let* i *be a positive integer.*

1. $E^i = \begin{cases} I & \text{if } i = 0, \text{ and} \\ E \circ E^{i-1} & \text{if } i > 0. \end{cases}$
2. $\Delta^i = \begin{cases} I & \text{if } i = 0, \text{ and} \\ \Delta \circ \Delta^{i-1} & \text{if } i > 0. \end{cases}$

Therefore, $E^i(g(x)) = g(x+i)$. Thus, Equation (4) can be rewritten as

$$(a_t E^t + a_{t-1}E^{t-1} + \cdots + a_0 I)(g(x)) \equiv 0 \pmod{p}. \tag{5}$$

We call the polynomial

$$a_t X^t + a_{t-1}X^{t-1} + \cdots + a_1 X + a_0 \tag{6}$$

the characteristic polynomial for Equation (4). The roots of the characteristic polynomial are called the characteristic roots of Equation (5).

Since $a_t \neq 0$, without loss generality, we may consider the following linear difference equation

$$(E^t + a_{t-1}E^{t-1} + \cdots + a_0 I)(g(x)) \equiv 0 \pmod{p}, \tag{7}$$

and the corresponding characteristic polynomial is:

$$X^t + a_{t-1}X^{t-1} + \cdots + a_1 X + a_0. \tag{8}$$

Suppose that we choose the coefficients a_i's such that

$$X^t + a_{t-1}X^{t-1} + \cdots + a_0 = (X-\lambda_1)^{m_1}(X-\lambda_2)^{m_2}\cdots(X-\lambda_k)^{m_k},$$

where $\sum_{i=1}^{k} m_i = t$. Then Equation (7) can be rewritten as:

$$(E-\lambda_1 I)^{m_1}(E-\lambda_2 I)^{m_2}\cdots(E-\lambda_k)^{m_k} g(x) \equiv 0 \pmod{p}. \tag{9}$$

Note that since $a_0 \in \mathbb{Z}_p^*$ and p is a prime number, we have $\lambda_i \in \mathbb{Z}_p^*$ for all i. Consider the following equation:

$$(E-\lambda_1 I)^{m_1} g(x) \equiv 0 \pmod{p}. \tag{10}$$

It is observed that any solution of Equation (10) will also be a solution of Equation (9). If $m_1 = 1$, then function λ_1^x is a solution of Equation (10). If $m_1 > 1$, $\lambda_1^x, x\lambda_1^x, \cdots, x^{m_1-2}\lambda^x$ and $x^{m_1-1}\lambda_1^x$ are m_1 functional linear independent solutions of Equation (10). Hence, if Equation (5) has characteristic roots $\lambda_1, \lambda_2 \cdots, \lambda_k$ with multiplicities $m_1, m_2 \cdots, m_k$, respectively, then the functions $\lambda_1^x, \cdots, x^{m_1-1}\lambda_1^x, \cdots, \lambda_k^x, \cdots, x^{m_k-1}\lambda_k^x$ are t functional linear independent solutions of Equation (5). By Theorem 3.1, these functions are the basis of the solution space of Equation (5). That is, we have the following theorem:

Theorem 3.6. *Suppose that Equation (5) has k distinct characteristic roots $\lambda_1, \lambda_2, \cdots, \lambda_k$ with multiplicities $m_1, m_2, \cdots, m_k$. Then the general solution of Equation (5) is*

$$c_{1,0}\lambda_1^x + \cdots + c_{1,m_1-1}x^{m_1-1}\lambda_1^x + \cdots + c_{k,m_k-1}x^{m_k-1}\lambda_k^x \pmod{p},$$

where the t $c_{i,j}$'s are constants to be determined with the initial conditions of Equation (5).

With the same setting as the coefficients in Equation (1), the homogeneous Equation (1) can be rewritten as:

$$(E - \lambda_1 I)^{m_1}(E - \lambda_2 I)^{m_2} \cdots (E - \lambda_k I)^{m_k} \equiv b \pmod{p}. \tag{11}$$

Because b is a constant, i.e. $\Delta b = (E - I)(b) = 0$. Thus, Equation (11) can be transformed into a homogeneous one:

$$(E - I)(E - \lambda_1 I)^{m_1}(E - \lambda_2 I)^{m_2} \cdots (E - \lambda_k I)^{m_k} \equiv 0 \pmod{p}. \tag{12}$$

Thus, we have the following theorem:

Theorem 3.7. *Let Equation (1) have k distinct characteristic roots $\lambda_1, \lambda_2, \cdots, \lambda_k$ with multiplicities $m_1, m_2, \cdots, m_k$. Then the general solution of Equation (1) is*

$$c_0 + c_{1,0}\lambda_1^x + \cdots + c_{1,m_1-1}x^{m_1-1}\lambda_1^x + \cdots + c_{k,m_k-1}x^{m_k-1}\lambda_k^x \pmod{p},$$

where c_0 can be determined by Equation (1) and the t $c_{i,j}$'s are constants to be determined by the initial conditions of Equation (1).

In conclusion, the characteristic roots $\lambda_1, \lambda_2, \cdots, \lambda_k$ and their multiplicities $m_1, m_2, \cdots, m_k$, define the coefficients of Equation (5), where $m_1 + m_2 + \cdots + m_k = t$. In addition, the parameter c_0 defines the constant coefficient b in Equation (1). In such a setting, the parameters $c_{i,j}$'s are one-one correspondence to the initial conditions of the Equation (1). Based on Theorem 3.1, we can break a secret, some parameter $c_{i,j}$, into a set of shadows, the function values of the solution of Equation (1), such that any t of $t+1$ consecutive shadows can recover the secret by solving Equation (11).

In the next section, we will present three secret sharing schemes. Two of the three schemes are special cases of traditional $(t, n)-$threshold scheme. One is a secret sharing scheme with the access structure where any t of $t+1$ consecutive shadows can recover the original secret. Another is a secret sharing scheme with the access structure where only t of consecutive shadows can recover the original secret.

4 Proposed Schemes

Recall the case that the company with its vault of secrets. Suppose that the company employs n employees, denoted by $\mathcal{P}$ but they do not trust the combination to any individual employee. These employees take their vacations in turns. Suppose that each employee is assigned an ID that corresponds with the order of vacations he will take. The company may require that any t employees can open the vault to perform the business procedure. The corresponding secret sharing scheme is called $(t, n)-$threshold scheme. Or, the company may require that any t consecutive employees of $\mathcal{P}$ can open the vault to perform the business procedure. We call the corresponding secret sharing scheme a $(0, t, n)-$threshold scheme. Another possibility is the company may require that any t employees

of any $t+1$ consecutive employees can open the vault to execute their jobs. Hence, we would like to design a system to fulfill the company's secret sharing policies. The corresponding secret sharing scheme we will introduce is called a $(1,t,n)-$threshold scheme. The informal definitions of these schemes are as follows.

Definition 4.1. *A $(t,n)-$threshold scheme is a method of sharing a key K among a set of n participants with the access structure that any subset of t or more than than t participants can compute the value K, but no other subset of participants can do so.*

Definition 4.2. *A consecutive $(0,t,n)$-threshold scheme is a restricted (t,n)-threshold scheme. It is a method of sharing a key K among an ordered set of n participants with the access structure that only certain subsets of participants that contain any t or more than t consecutive participants can compute the value K, but no other subset of participants can do so.*

Definition 4.3. *A consecutive $(1,t,n)$-threshold scheme is a restricted (t,n)-threshold scheme. It is a method of sharing a key K among an ordered set of n participants with the access structure that only certain subsets of participants that contain any t participants of any $t+1$ or more consecutive participants can compute the value K, but no other subset of participants can do so.*

We will design threshold schemes based on difference equations. First, a $(t,n)-$threshold scheme will be presented. Second, a $(1,t,n)-$threshold scheme is presented. Third, a $(0,t,n)-$threshold scheme is presented. Let n and t represent two positive integers with $t \leq n$. The ordered set $\{1,2,\cdots,n\}$ represents the set of participant's IDs. The dealer selects a large prime p, where $n \ll p$. Let K represent the key which will be broken into n shadows. Below, we present a $(t,n)-$threshold scheme.

A $(t,n)-$Threshold scheme

1. The dealer randomly selects integers $\lambda_1, \lambda_2, \cdots, \lambda_k \in \mathbb{Z}_p^*$ and selects positive integers $m_1, m_2, \cdots, m_k$ such that $\sum_{i=1}^{k} m_i = t$.
2. The dealer computes the polynomial $F(X)$, where

$$\begin{aligned} F(X) &= (X-\lambda_1)^{m_1}(X-\lambda_2)^{m_2}\cdots(X-\lambda_k)^{m_k} \\ &= a_t X^t + a_{t-1}X^{t-1} + \cdots + a_0. \end{aligned}$$

3. The dealer randomly selects $t+1$ secret integers c_0, $c_{1,0}$, $\cdots$, c_{1,m_1-1}, $\cdots$, $c_{k,0}$, $\cdots$, $c_{k,m_k-1} \in \mathbb{Z}_p$ such that $c_{1,m_1-1} = K$.
4. The dealer constructs the function f,

$$f(x) = c_0 + \sum_{i_1=0}^{m_1-1} c_{1,i_1} x^{i_1} \lambda_1^x + \cdots + \sum_{i_k=0}^{m_k-1} c_{k,i_k} x^{i_k} \lambda_k^x \pmod{p}. \qquad (13)$$

5. The dealer uses the function $f(x)$ and the following difference equation to compute the constant b:

$$a_t f(x+t) + a_{t-1} f(x+t-1) + \cdots + a_0 f(x) \equiv b \pmod{p}. \tag{14}$$

6. The dealer computes the shadows $y_1, y_2, \cdots, y_n$, where

$$y_i = f(i) \pmod{p} \qquad \text{for all } i = 1, \cdots, n.$$

7. The dealer broadcasts Equation (14) with the unknown function $f(x)$. Then the dealer distributes these shadows $y_1, y_2, \cdots, y_n$ to the participants separately and secretly.

Suppose that there are t participants polling their shadows $y_{j_1}, y_{j_2}, \cdots, y_{i,t}$. They can use the following procedure to recover the function $f(x)$, and then retrieve the key K.

1. They use Equation (12) to construct the polynomial $F(X)$, and then compute $\lambda_1, \lambda_2, \cdots, \lambda_k$ and $m_1, m_2, \cdots, m_k$.
2. They construct the following system of linear equations:

$$\begin{aligned}
c_0 + \sum_{i_1=0}^{m_1-1} c_{1,i_i} {j_1}^{i_1} \lambda_1^{j_1} + \cdots + \sum_{i_k=0}^{m_k-1} c_{k,i_k} {j_1}^{i_k} \lambda_k^{j_1} &\equiv y_{j_1} \pmod{p} \\
c_0 + \sum_{i_1=0}^{m_1-1} c_{1,i_i} {j_2}^{i_1} \lambda_1^{j_2} + \cdots + \sum_{i_k=0}^{m_k-1} c_{k,i_k} {j_2}^{i_k} \lambda_k^{j_2} &\equiv y_{j_2} \pmod{p} \\
&\vdots \\
c_0 + \sum_{i_1=0}^{m_1-1} c_{1,i_i} {j_t}^{i_1} \lambda_1^{j_t} + \cdots + \sum_{i_k=0}^{m_k-1} c_{k,i_k} {j_t}^{i_k} \lambda_k^{j_t} &\equiv y_{j_t} \pmod{p}
\end{aligned}$$

Then, they solve $c_0, c_{i,j}$'s. And obtain $K = c_{1,m_1-1}$.

Both the key space and the shadow space of the proposed $(t,n)-$threshold scheme are the number field $\mathbb{Z}_p$. Let Γ^1 represent the access structure of the proposed $(t,n)-$threshold scheme and Γ_0^1 represent the basis of Γ^1. Then, by the theory of linear equation systems, we have

$$\Gamma_0^1 = \{A \in 2^{Z_p} : |A| = t\}.$$

Thus, the proposed $(t,n)-$threshold scheme is the same as the Shamir's $(t,n)-$threshold scheme.

Next, we present a $(1,t,n)-$threshold scheme as below:

The $(1,t,n)-$Threshold Scheme

1. The dealer selects a polynomial $F(X)$ over $\mathbb{Z}_p$,

$$F(X) = a_t X^t + a_{t-1} X^{t-1} + \cdots + a_1 X + a_0,$$

such that $a_t, a_{t-1}, \cdots, a_0 \in \mathbb{Z}_p^*$ and $F(X)$ contains an irreducible factor of degree d over $\mathbb{Z}_p$, where $d \geq 2$.

2. The dealer constructs the following difference equation by randomly selecting an integer b:

$$a_t f(x+t) + a_{t-1} f(x+t-1) + \cdots + a_0 f(x) \equiv b \pmod{p}. \quad (15)$$

3. The dealer randomly selects an integer $M \in \mathbb{Z}_p^* \setminus \{1, 2, \cdots, n\}$ and $t-1$ secret integers $z_1, z_2, \cdots, z_{t-1}$.
4. The dealer uses $f(M) = K, f(M+1) = z_1, \cdots, f(M+t-1) = z_{t-1}$ as the initial conditions of Equation (15) to compute the shadows $y_1 = f(1), y_2 = f(2), \cdots, y_n = f(n)$.
5. The dealer broadcasts Equation (15) and the parameter M, and distributes the shadows to participants separately and secretly.

Suppose that there are t participants of $t+1$ consecutive participants polling their shadows. Then, they can use the following procedure to recover the key K.

1. They use Equation (2) to compute the lost initial value. Hence, they will have $t+1$ consecutive initial values of the function $f(x)$. Let $y_i, y_{i+1}, \cdots, \breve{y}_{i+j}, \cdots, y_{i+t}$ be the t initial values of $f(x)$, where $\breve{y}_{i+j}$ denotes the absence of y_{i+j}. Then,

$$y_j \equiv (b - a_t y_{i+t} - a_{t-1} y_{i+t-1} - \cdots - \breve{y}_{i+j} - \cdots - a_0 y_i) a_{i+j}^{-1}.$$

2. Use Equation (2) to compute the function value $f(M)$.
3. $K = f(M)$.

It is easy to see that both the key space and the shadow space of the proposed $(1, t, n)-$threshold scheme are the number field $\mathbb{Z}_p$. Let Γ^2 represent the access structure of the proposed $(1, t, n)-$threshold scheme and Γ_0^2 represent the basis of Γ^2. Thus,

$$\Gamma_0^2 = \{y_{i+1}, y_{i+2}, \cdots, \breve{y}_j, \cdots, y_{i+t+1} : \text{ for suitable } i\},$$

where $\breve{y}_j$ denotes the absence of y_j.

We can transform a $(1, t, n)-$threshold scheme into a $(0, t, n)-$threshold scheme by modifying the generation of shadows.

The $(0, t, n)-$Threshold Scheme

1. The dealer selects a polynomial $F(X)$ over $\mathbb{Z}_p$,

$$F(X) = a_t X^t + a_{t-1} X^{t-1} + \cdots + a_1 X + a_0,$$

such that $a_t, a_{t-1}, \cdots, a_0 \in \mathbb{Z}_p^*$ and $F(X)$ contains an irreducible factor of degree d over $\mathbb{Z}_p$, where $d \geq 2$.
2. The dealer constructs the following difference equation by randomly selecting an integer b:

$$a_t f(x+t) + a_{t-1} f(x+t-1) + \cdots + a_0 f(x) \equiv b \pmod{p}. \quad (16)$$

3. The dealer randomly selects an integer $M \in \mathbb{Z}_p^* \setminus \{1, 2, \cdots, n\}$ and $t-1$ secret integers $z_1, z_2, \cdots, z_{t-1}$.
4. The dealer uses $f(M) = K, f(M+1) = z_1, \cdots, f(M+t-1) = z_{t-1}$ as the initial conditions of Equation (16) to determine the function $f(x)$ uniquely. Let $i \in \{1, 2, \cdots, n\}$. Then the shadow y_i is computed as $y_i = f(i + \lfloor \frac{i}{t-1} \rfloor)$.
5. The dealer broadcasts Equation (16) and the parameter M, keeps z_1, z_2, $\cdots$, z_{t-1} secret, and distributes the shadows to participants separately and secretly.

It is easy to see that no subset of participants can contain t consecutive initial conditions of Equation (16). The best possible case is that a participant subset contains t initial conditions with $t-1$ consecutive initial conditions of Equation (16) and the other initial condition with a gap of size 1 to the consecutive initial condition. Suppose that there is a participant subset satisfying this threshold condition. They can compute K as follows:

1. They use Equation (2) to compute the lost initial value. Hence, they will have t consecutive initial values of the function $f(x)$. The computation of the lost initial value of $f(x)$ is the as same as the case in the $(1, t, n)-$Threshold Scheme.
2. Use Equation (2) to compute the function value $f(M)$.
3. $K = f(M)$.

Let Γ^3 denote the access structure of the proposed scheme $(0, t, n)-$threshold scheme and Γ_0^3 denote the basis of Γ^3. Then,

$$\Gamma_0^3 = \{y_i, y_{i+1}, \cdots, y_{i+t-1} | \text{ for suitable } i\}.$$

In the next section, we will present the security analysis of the proposed schemes.

5 Analysis

First, we shall show that all proposed threshold schemes are ideal secret sharing schemes. Because $K, y_1, \cdots, y_n \in \mathbb{Z}_p$, the size of the shadow y_i is the same as the size of the key K. By the definition of ideal secret sharing schemes, all proposed schemes are ideal secret sharing schemes.

It is easy to see that for any choice of $\lambda_1, \lambda_2, \cdots, \lambda_k \in \mathbb{Z}_p^*$ and $m_1, m_2, \cdots, m_k$ such that $\sum_{i=1}^{k} m_i = t$, the set of functions $1, \lambda_1^x, x\lambda_1^x, x^2\lambda_1^x, \cdots, x^{m_1-1}\lambda_1^x, \cdots, \lambda_k^x, \cdots, x^{m_k-1}\lambda_k^x$ is functional linear independent. Thus, any t linear independent initial conditions uniquely determines a solution of Equation (1). We need to guess an initial condition in addition to $t-1$ linear independent initial conditions to solve Equation (1). However, this also determines a unique solution. This is not easier than guessing the key K directly. The proposed $(t, n)-$threshold scheme is therefore perfect and secure. The proof of the following theorem is shown.

Theorem 5.1. *The proposed $(t,n)-$threshold scheme is an ideal perfect secret sharing scheme.*

Proof. Suppose that the characteristic polynomial of Equation (1) is

$$F(X) = \prod_{i=1}^{k} (X - \lambda_i I)^{m_i},$$

where $\sum_{i=1}^{k} m_i = t$. Then, the general solution of Equation (1) is

$$c_0 + c_{1,0}\lambda_1^x + c_{1,1}x\lambda_1^x + \cdots + c_{k,0}\lambda_k^x + \cdots c_{k,m_k-1}x^{m_k-1}\lambda_{m_k-1}^x,$$

where the t coefficients, $c_{i,j}$'s, are constants to be determined and c_0 can be determined from Equation (1). Note that the set of $t+1$ functions 1, λ_1^x, $x\lambda_1^x$, $\cdots$, $x^{m_k-1}\lambda_k^x$ is linear independent.

Suppose that $y_{i_0} = f(x_{i_0}), y_{i_1} = f(x_{i_1}), \cdots, y_{i_{t-1}} = f(x_{t-1})$ are t function values of $f(x)$. Then, we have the following system of linear equations

$$\begin{aligned}
y_{i_0} &= c_0 + c_{1,0}\lambda_1^{x_{i_0}} + \cdots + c_{k,m_k-1}{x_{i_0}}^{m_k-1}\lambda_k^{x_{i_0}},\\
y_{i_1} &= c_0 + c_{1,0}\lambda_1^{x_{i_1}} + \cdots + c_{k,m_k-1}{x_{i_1}}^{m_k-1}\lambda_k^{x_{i_1}},\\
&\vdots\\
y_{i_{t-1}} &= c_0 + c_{1,0}\lambda_1^{x_{i_{t-1}}} + \cdots + C_{k,m_k-1}{x_{i_{t-1}}}^{m_k-1}\lambda_k^{x_{i_{t-1}}}.
\end{aligned}$$

There are t unknowns and t linear independent equations. By the theory of linear algebra, there is a unique solution $(c_{1,0}, c_{1,1}, \cdots, c_{k,m_k-1})$ to determine the function $f(x)$. And $K = c_{1,m_1-1}$. Suppose that we only have $t-1$ y_i's. Then we can only get $t-1$ linear equations with t unknowns. Because, we can't use $t-1$ linear independent equations to construct the one more required linear equation for $(c_{1,0}, c_{1,1}, \cdots, c_{k,m_k-1})$. Thus, we need to guess a function value to have the required linear equation. However, there is no help to the original solution $f(x)$ by Theorem 3.1. Thus, having only $t-1$ shadows, we can not compute any information to the original solution $f(x)$. The proposed scheme therefore is perfect secure.

In the other two proposed threshold schemes, we choose the polynomial $F(X) = a_tX^t + a_{t-1}X^{t-1} + \cdots + a_0$ containing an irreducible factor of degree d over $\mathbb{Z}_p$, where $d \geq 2$. The explicit form of the basis of the solution space to Equation (4) is unknown. That is, the general solution of Equation (1) is unknown. It has a general solution or not, one can use Equation (2) or Equation (3) to define the solution $f(x)$ and compute its values completely. Thus, we assume the following hypothesis.

Hypothesis
Equation (1) with the polynomial $F(X)$ and having an irreducible factor of degree d, which is greater than or equal to 2, has no explicit general solution over $\mathbb{Z}_p$.

Now, we have the following proposition:

Proposition 5.2. *Suppose that* $F(X) = a_t X^t + a_{t-1} X^{t-1} + \cdots + a_0$ *containing an irreducible factor of degree* d *over* $\mathbb{Z}_p$*, where* $d \geq 2$*. If there is no closed form for the general solution of Equation (1), then the proposed* $(0, t, n)-$*threshold scheme and the* $(1, t, n)-$*threshold scheme are perfect and secure.*

It is observed that Equation (2) and Equation (3) are linear dependent. That is, they are the same linear equation. However, they are equivalent to Equation (1). In the following, we assume that $F(X)$ contains an irreducible factor of degree d, where $d \geq 2$. To prove Proposition 5.2, we need to define the following notion:

Definition 5.3. *Given a set* I *of initial conditions of Equation (1), we call the member of* I *a known function value. An absent function value* $f(x')$ *is 1-recoverable using Equation (1), if* $\{f(i), f(i+1), \cdots, f(x'-1), f(x'+1), \cdots, f(i+t)\}$*,* $\{f(x'+1), \cdots, f(x'+t)\}$*, or* $\{f(x'-t), \cdots, f(x'-1)\}$ *are known for some* $i \in \mathbb{Z}_p$*. And, we say that* I *admits the 1-recoverable absent function value* $f(x')$*.*

According to Theorem 3.1 and Theorem 3.3, it is easy to see that $f(x)$ is recoverable from Equation (1) if we have an absent function value $f(x')$ that is 1-recoverable from Equation (1). Note that an initial condition contains a value of $f(x)$, so we will use the notion, initial condition and initial value interchangeable for convenience. Because the set of initial values to Equation (1) is a subset of the function values of the solution $f(x)$, we call it an initial value set that is recoverable if it admits a 1-recoverable absent function value from I. Or, we say that $f(x)$ is recoverable from I. A set of initial values admitting an 1-recoverable absent function value is a certificate of $f(x)$ that is recoverable. The following lemma is obvious.

Lemma 5.4. *Let* $F(X)$ *contain an irreducible factor of degree* d*, where* $d \geq 2$*. Suppose that* I *is a set of initial values to the Equation (1). Then,* I *admits an 1-recoverable absent function value if and only if* $f(x)$ *is recoverable from* I*.*

Proof. (Proposition 5.2) It is easy to see that the $(0, t, n)$-threshold scheme can be reduced to a $(1, t, n)-$threshold scheme. If the $(1, t, n)-$threshold scheme is true, then the $(0, t, n)-$threshold scheme is also true. Thus, we will only consider $(1, t, n)-$threshold scheme.

Suppose that a shadow subset S_1 contains the shadows $y_i = f(x_0 + i)$ for all $i = 0, 1, \cdots, t-1$. By the assumption $a_t, a_0 \in \mathbb{Z}_p^*$, we may use Equation (2) and Equation (3) to compute all the values of the function $f(x)$. And, these values uniquely determine the function $f(x)$. Thus, the shadow subset S_1 can compute the secret K uniquely.

Suppose that a shadow subset S_2 contains the shadows $y_i = f(x_0 + i)$ for all $i = 0, 1, \cdots, j-1, j+1, \cdots, t$. By the assumption $a_t, a_0 \in \mathbb{Z}_p^*$, we can repeatedly use Equation (2) or Equation (3) to compute the value $y_j = f(x_0 + j)$. That is y_j is a 1-recoverable absent function value. Then, we may use Equation (2) and Equation (3) to compute all the values of the function $f(x)$. These values

uniquely determine the function $f(x)$. Thus, the shadow subset S_2 can compute the secret K uniquely.

The basis Γ_0^2 of the $(1,t,n)-$threshold scheme can be defined as:

$$\begin{aligned}\Gamma_0^2 = \quad & \{\{y_i, y_{i+1}, \cdots, y_{i+t-1}\} : \text{ all possible integer } i \in \mathbb{Z}_p\} \\ & \cup \{\{y_i, y_{i+1}, \cdots, y_{j-1}, y_{j+1}, \cdots, y_{j+t}\} : \text{ all possible integer } i \in \mathbb{Z}_p\}.\end{aligned}$$

Let $\Gamma = closure(\Gamma_0)$. Suppose that $S_3 \notin \Gamma$ and S_3 admits a 1-recoverable absent function value $f(j)$. Because we can use the initial values in S_3 to recover the absent value $f(j)$, S_3 must have a subset $A \in \Gamma_0$. Thus, $S_3 \in \Gamma$ a contradiction. S_3 therefore can't admit an absent function value. Thus, we need to guess some initial values to combine S_3 for $f(x)$. But such guessing is no different from the guessing of K directly. Thus, the proposed scheme is a perfect and secure scheme.

6 Conclusions

There are many business situations in which secret sharing schemes can apply. There is a special case of secret sharing schemes called threshold schemes. A threshold scheme defines a threshold as any number of independent shadows larger than or equal to the threshold that can compute the original secret K, but no other number of shadows can do so. Thus, traditional threshold secret sharing schemes count the number of legal shadows to decide that the polled shadow set can compute the secret. However, there are also many secret sharing policies which need to use order structure of some form to depict themselves. One may impose the checking procedure of the order restriction for the legal shadow subset to the upper layer of the $(t,n)-$threshold scheme. But, it would be better that the secret sharing scheme may contains the ability to describe the order structure restriction into the definition of the legal shadow subsets.

In the proposed schemes, the order structure of the shadow set plays an important role under the hypothesis in the previous section. If the solving procedures of difference equations or recurrent equations are only depend on the order structure of their initial values, then the proposed schemes can impose the order structure into the descriptions of the access structures of secret sharing policies. Based on the hypothesis, the proposed schemes use the order structure of shadow set to depict the order restriction of the legal shadow subset.

The proposed $(0,t,n)-$threshold scheme and $(1,t,n)-$threshold scheme require consecutive shadows to recover the secret K. This is very different from the traditional schemes. In the future, we will investigate whether there is a $(i,t,n)-$threshold scheme for all possible i.

References

1. Shamir, A.: how to share a secret. Communications of the ACM (22), 612–613 (1979)
2. Stinson, D.R.: cryptography–theory and practice. CRC Press, Boca Raton, London, Tokyo (1995)

Clique-Transversal Sets in Cubic Graphs*

Zuosong Liang[1], Erfang Shan[1,2,**], and T.C.E. Cheng[2]

[1] Department of Mathematics, Shanghai University, Shanghai 200444, P.R. China
efshan@shu.edu.cn
[2] Department of Logistics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
lgtcheng@inet.polyu.edu.hk

Abstract. A clique-transversal set S of a graph G is a set of vertices of G such that S meets all cliques of G. The clique-transversal number, denoted $\tau_c(G)$, is the minimum cardinality of a clique-transversal set in G. In this paper we present an upper bound and a lower bound on $\tau_c(G)$ for cubic graphs, and characterize the extremal cubic graphs achieving the lower bound. In addition, we present a sharp upper bound on $\tau_c(G)$ for claw-free cubic graphs.

Keywords: Clique-transversal number; Cubic graph; Claw-free; Bound.

1 Introduction

All graphs considered here are finite, simple and nonempty. For standard graph theory terminology not given here we refer the reader to [5].

Let $G = (V, E)$ be a graph with vertex set V and edge set E. For a vertex $v \in V$, the *degree* of v is denoted by $d(v)$ and a vertex of degree 0 is said to be an isolated vertex. If $d(v) = k$ for all $v \in V$, then we call G *k-regular*. In particular, a 3-regular graph is also called a *cubic graph*. For a subset $S \subseteq V$, the subgraph induced by S is denoted by $G[S]$, and let $d_S(v)$ denote the number of vertices in S that are adjacent to v. For two disjoint subsets T and S of V, write $e[T, S]$ for the number of edges between T and S.

The *matching number* of G is the maximum cardinality among the independent sets of edges of G and is denoted by $\alpha_1(G)$. A *perfect matching* in G is a matching with the property that every vertex in G is incident with an edge of the matching. A set $U \subseteq V$ is called a *vertex cover* of G if every edge of G is incident with a vertex in U. The covering number, denoted by $\alpha_0(G)$, is the minimum cardinality of a vertex cover of G. A subset S of V is called a *dominating set* if every vertex of $V - S$ is adjacent to some vertex in S. The *domination*

* This research was partially supported by The Hong Kong Polytechnic University under grant number G-YX69, the National Nature Science Foundation of China under grant 10571117, the ShuGuang Plan of Shanghai Education Development Foundation under grant 06SG42 and the Development Foundation of Shanghai Education Committee under grant 05AZ04.

** Corresponding author.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 107–115, 2007.

number $\gamma(G)$ of G is the minimum cardinality taken over all dominating sets of G. Domination in graphs has been well studied (see [12]).

The concept of the clique-transversal set in graphs can be regarded as a special case of the transversal set in hypergraph theory, which is closely related to domination and seems to have been introduced in [1]. A *clique* C of a graph G is a complete subgraph maximal under inclusion and $|C| \geq 2$. A set $D \subseteq V$ in G is called a *clique-transversal set* if for every clique C of G, $D \cap V(C) \neq \emptyset$. The *clique-transversal number*, denoted $\tau_c(G)$, is the minimum cardinality of a clique-transversal set of G. By definitions, each clique-transversal set in G is clearly a dominating set, so $\gamma(G) \leq \tau_c(G)$.

To motivate the study of the clique-transversal set in graphs, we present two examples of application where this concept may be used. An application is in terms of communication networks. Consider a graph associated with a communication network where the vertices in the graph correspond to the sites of the network, a clique usually represents a cluster of sites that has the best possible ability to rapidly exchange information among the members of the cluster. The clique-transversal set in the graph is faster to control all clusters and keeps the ability of dominating the whole network. Another application may be found in social networks theory. Every vertex of a graph represents an actor and an edge represents a relationship between two actors. A clique can be viewed as a maximal group of members that have the same property, while a clique-transversal set can be regarded as some kind of organization in the social networks. Then a clique-transversal set claims that each clique in the social networks owns at least one position in this organization.

Erdős, Gallai and Tuza [10] observed that the problem of finding a minimum clique-transversal set for an arbitrary graph is NP-hard. Further, it has been proved that the problem is still NP-hard on split graphs (a subclass of chordal graphs) [7], cocomparability, planar, line and total graphs [11], undirected path graphs, and k-trees with unbounded k [6]. However, there are polynomial time algorithms to find τ_c for comparability graphs [4], strongly chordal graphs [7], Helly circular-arc graphs [16], and distance-hereditary graphs [14]. In [10], Erdős et al. investigated the bounds on τ_c, and showed that every graph of order n has clique-transversal number at most $n - \sqrt{2n} + \frac{3}{2}$, and if all cliques are relatively large, then a sightly better upper bound can be obtained. However, they also observed that $\tau_c(G)$ can be very close to $n = |V(G)|$, namely $\tau_c = n - o(n)$ can hold. It is interesting to note that τ_c drastically decreases when some assumptions are put on the graph G. From this point of view, Tuza [17] and Andreae [2] established upper bounds on τ_c for chordal graphs. Andreae et al. [3] studied classes of graphs G of order n for which $\tau_c(G) \leq n/2$. They showed that (i) all connected line graphs with the exception of odd cycles, and (ii) all complements of line graphs with the exception of five small graphs have clique-transversal numbers at most one-half their orders. For other investigations on the clique-transversal number of graphs, we refer the reader to [6, 8, 9, 13, 15, 18].

In this paper we shall focus on cubic graphs. We show that if G is any cubic graph, then $5n/14 \leq \tau_c(G) \leq 2n/3$, and the extremal graphs attaining the lower

bound are characterized. Also, we show that a claw-free cubic graph has clique-transversal number at most one-half its order, and the upper bound is sharp.

2 Clique-Transversal Number in Cubic Graphs

In this section we give lower and upper bounds on the clique-transversal number of a cubic graph in terms of its order and we characterize the graph attaining the lower bound. For this purpose, we define a family $\mathcal{F}$ of graphs as follows: For each integer $k \geq 1$, let J_0 be the graph obtained from a complete graph K_4 on four vertices by deleting one edge, and let J be the disjoint union of $J_1, J_2, \ldots, J_{3k}$ of $3k$ copies of J_0. Let F_k be a family of cubic graphs obtained from J by adding $2k$ new vertices and $6k$ edges that join each new vertex exactly to three vertices of degree 2 of J so that each vertex has degree 3. Let $\mathcal{F} = \{F_k \mid k \geq 1\}$. The graph F_1 is shown in Fig. 1.

For notational convenience, every clique of order m of a graph G is called a K_m-*clique* of G, and a component of G is called a H-*component* if it is isomorphic to a given graph H.

Theorem 1. *If G is a cubic graph of order $n \geq 5$, then*

$$\frac{5}{14}n \leq \tau_c(G) \leq \frac{2}{3}n$$

with the left equality if and only if $G \in \mathcal{F}$.

Proof. We may assume that G is connected, otherwise we look at each (connected) component separately. Since $n \geq 5$, G contains only K_2-cliques and K_3-cliques. Let D be a minimum clique-transversal set of G. We have the following claims.

Claim 1. Each component of $G[V - D]$ is a P_i-component, where $1 \leq i \leq 3$.

Let v be a vertex in $V - D$. If $d_D(v) = 3$, then v is an isolated vertex in $G[V - D]$, so v is a P_1-component of $G[V - D]$. If $d_D(v) = 2$, then there exists another vertex v' of $V - D$ such that v and v' are adjacent. Since every clique of G is dominated by some vertex of D, it follows that the edge $e = vv'$ lies in at least one K_3-clique and there exists one vertex v_D of D that is adjacent to both v and v'. So if $d_D(v') = 2$, then the vertices v, v' induce a P_2-component of $G[V - D]$. If $d_D(v') = 1$, then there is another vertex v'' of $V - D$ that is adjacent to v'. Clearly, the edge $v'v''$ is contained in a K_3-clique, and thus v'' is adjacent to v_D as $d_D(v') = 1$. This implies that the third neighbor of v'' is distinct from v', and v_D belongs to D, so the vertices v, v' and v'' induce a P_3-component of $G[V - D]$. If $d_D(v) = 1$, following the discussion similar to the case $d_D(v') = 1$, we can show that there exist two vertices v' and v'' of $G[V - D]$ such that v is adjacent to both v' and v'' and there exist one vertex v_D of D such that v_D is adjacent to v, v' and v''. Then the third neighbor of v' as well as v'' distinct from v, v_D lies in D, so the vertices v, v' and v'' induce a P_3-component of $G[V - D]$ and Claim 1 follows.

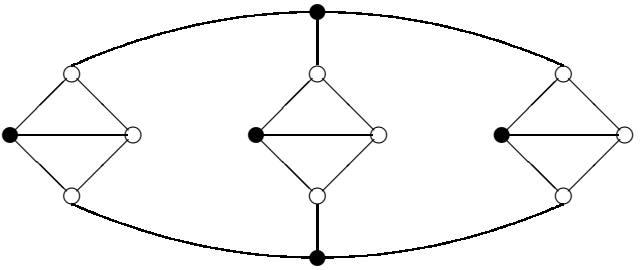


Fig. 1. The graph F_1 with $\tau_c(F_1) = \frac{5}{14}|V(F_1)|$

Claim 2. For every vertex v of D, $d_D(v) \leq 2$.

The minimality of D implies that v is adjacent to some vertex of $V - D$ for otherwise $D - \{v\}$ would be a clique-transversal set of G, so Claim 2 follows.

First, we present a lower bound on $\tau_c(G)$. Let l_i be the number of P_i-components of $G[V - D]$ for $i = 1, 2, 3$. So $|V - D| = l_1 + 2l_2 + 3l_3$. By counting the number of edges between D and $V - D$, we immediately have

$$e[D, V - D] = 3l_1 + 4l_2 + 5l_3 \leq 3|D|, \tag{1}$$

hence $|D| \geq l_1 + \frac{4}{3}l_2 + \frac{5}{3}l_3$. So

$$\frac{9}{5}|D| \geq \frac{9}{5}l_1 + \frac{12}{5}l_2 + 3l_3 \geq |V - D|, \tag{2}$$

that is,

$$\frac{14}{5}|D| = |D| + \frac{9}{5}|D| \geq |D| + |V - D| = n.$$

Consequently, $\tau_c(G) = |D| \geq \frac{5}{14}n$.

We show next that for a cubic graph G of order n, $\tau_c(G) = \frac{5}{14}n$ if and only if $G \in \mathcal{F}$. Suppose $G \in \mathcal{F}$, then for some positive integer k, $G = F_k$ and thus $|V(G)| = 14k$. We choose one vertex of degree 3 in each J_i, together with the $2k$ new vertices added to J. These vertices clearly form a clique-transversal set of F_k with cardinality $5k$, so $\tau_c(F_k) = \frac{5}{14}|V(F_k)|$. The darkened vertices of F_1 indicated in Fig. 1 form a minimum clique-transversal set of F_1 with $\tau_c(F_1) = \frac{5}{14}|V(F_1)|$. Conversely, suppose that $\tau_c(G) = \frac{5}{14}n$ for a graph G, then there exists some integer k such that $n = 14k$ and let D be its minimum clique-transversal set. Then $|D| = 5k$. By the above proof, the equalities hold in inequalities (1) and (2). The equality in (1) implies that $G[D]$ has no edges, while the equality in (2) implies that $l_1 = l_2 = 0$, and $G[V - D]$ is a collection of P_3-components of cardinality $3k$. By the proof of Claim 1, we can see that there is a set of vertices of cardinality $3k$ of D such that each vertex of the set is precisely adjacent to the three vertices of one P_3-component in $G[V - D]$, which results in $3k$ copies of J_0. The remainders of D have exactly $2k$ vertices that are adjacent to the endpoints of all P_3-components in $G[V - D]$. So $G \in \mathcal{F}$.

Now we present an upper bound on $\tau_c(G)$. By Claim 2, we can partition D into sets $D_0 = \{v \in D \mid d_D(v) = 0\}$, $D_1 = \{v \in D \mid d_D(v) = 1\}$ and $D_2 = \{v \in D \mid d_D(v) = 2\}$, and let x, y and z be the cardinality of D_0, D_1 and D_2, respectively. Then every vertex of D_2 dominates precisely a vertex of $V - D$. On the other hand, every vertex of $V - D$ is adjacent to at most one vertex of D_2. Otherwise, suppose there exists a vertex v of $V - D$ that is adjacent to both vertices v_1 and v_2 of D_2, then $D \cup \{v\} - \{v_1, v_2\}$ is a clique-transversal set with order smaller than $\tau_c(G)$, a contradiction. Thus we have

$$|V - D| \geq |D_2| = z. \tag{3}$$

Further, observing that $e[D, V - D] = 3x + 2y + z \leq 3|V - D|$, we have

$$|V - D| \geq x + \frac{2}{3}y + \frac{1}{3}z. \tag{4}$$

If $z < x + \frac{2}{3}y + \frac{1}{3}z$, then $z < \frac{3}{2}x + y$, so

$$\begin{aligned} 2n &= 2(|D| + |V - D|) \\ &\geq 2(x + y + z) + 2(x + \frac{2}{3}y + \frac{1}{3}z) \quad (\text{ by (4)}) \\ &= 3(x + y + z) + \frac{1}{3}(3x + y - z) \\ &\geq 3\tau_c(G). \end{aligned}$$

Consequently, $\tau_c(G) \leq \frac{2}{3}n$. If $z \geq x + \frac{2}{3}y + \frac{1}{3}z$, then $z \geq \frac{3}{2}x + y \geq x + y$, so

$$\begin{aligned} \tau_c(G) = |D| &= x + y + z \\ &\leq \frac{2}{3}(x + y + 2z) \\ &\leq \frac{2}{3}(|D| + |V - D|) \quad (\text{ by (3)}) \\ &= \frac{2}{3}n, \end{aligned}$$

the desired result follows. □

If the cubic graph is claw-free, then the upper bound in Theorem 2 can be improved.

Theorem 2. *If G is a connected claw-free cubic graph of order n, then $\tau_c(G) \leq \frac{n}{2}$ and the bound is sharp.*

Proof. If $n \leq 4$, then $G = K_4$ and the result holds. So we may assume that $n \geq 5$, thus G contains only the K_2-cliques and K_3-cliques. Let E' be the set of edges of all K_2-cliques in G, and V' the set of vertices incident with edges of E'. We have the following claims.

Claim 1. E' is a matching in G.

Otherwise, there exist two edges of E' that share a common vertex, say v, and it would yield a claw at v, a contradiction.

Claim 2. $G[V - V']$ consists of only P_2-components.

For any vertex v of $V - V'$, there are two K_3-cliques in G containing v for otherwise v would belong to V'. Since G is claw-free, it follows that the two cliques share a common edge, say uv, which is incident with v. So $u \in V - V'$. But then the other vertices in the two K_3-cliques are contained in V'. So both u and v induce a P_2-component of $G[V - V']$ and Claim 2 follows.

By Claims 1 and 2, $E' \cup E(G[V - V'])$ is exactly a perfect matching in G, so $\alpha'(G) = \frac{n}{2}$.

Claim 3. For any vertex $v \in V'$, there exactly is a K_3-clique containing v.

Otherwise, it must be the case that $v \in V - V'$.

Claims 2 and 3 imply that for each K_3-clique in G, either its three vertices are from different K_2-cliques in G or it shares a common edge with another K_3-clique.

Now we construct a clique-transversal set S of G satisfying Property (A) as follows:

(i) For every P_2-component P_2 in $G[V - V']$, $|S \cap P_2| = 1$;
(ii) for every K_2-clique K_2 in G, $|S \cap K_2| \geq 1$;
(iii) for every K_3-clique K_3 in G, $|S \cap K_3| \geq 1$.

We can easily see that S is a clique-transversal set of G. Choose an S such that $|S|$ is minimum. We next show that $|S \cap K_2| = 1$ for every K_2-clique K_2 in G.

Suppose to the contrary that there exist $u_0, v_1 \in S$ that are in the same K_2-clique in G. By Claim 3, there exists a K_3-clique in G containing v_1. Let $\{v_1, w_1, u_1\}$ be the set of vertices of the K_3-clique. The minimality of S implies that $w_1, u_1 \notin S$ for otherwise $S - \{v_1\}$ is a smaller set that satisfies Property (A), a contradiction. Furthermore, w_1 and u_1 must belong to V'. Suppose it is not the case, then w_1 and u_1 in $G[V - V']$ would induce a P_2-component, so it follows from (i) that $S - \{v_1\}$ is a smaller set that satisfies Property (A). Hence $w_1, u_1 \in V' - S$. Let $\{u_1, v_2\}$ be the set of vertices of the K_2-clique in G containing u_1. Then v_2 must be in S by (ii). By Claim 3 again, we may assume that $\{v_2, w_2, u_2\}$ is the set of vertices of the K_3-clique in G containing v_2. Then we claim that only the vertex v_2 in the K_3-clique belongs to S. Without loss of generality, suppose $u_2 \in S$, then we would obtain a new set $S' = S \cup \{u_1\} - \{v_1, v_2\}$. Obviously, S' still satisfies Property (A), and we arrive in a contradiction again. So u_2 and w_2 belong to $V' - S$ by (i). Let $\{u_2, v_3\}$ be the set of vertices of the K_2-clique in G containing u_2. Then $v_3 \in S$ by (ii) again. We further consider that the K_3-clique in G contains v_3. Let $\{v_3, w_3, u_3\}$ be the set of vertices of the K_3-clique in G containing v_3. Similarly, we can show that $v_3 \in S$ and $w_3, u_3 \in V' - S$. We continue the above process by searching alternately for the K_2-cliques and K_3-cliques in G. Finally, we obtain a sequence of K_2-cliques and K_3-cliques alternately occurring in G. In the t-th step, we get a K_3-clique, say $K_3(v_t, w_t, u_t)$,

and we claim that only the vertex v_t in the K_3-clique belongs to S. Without loss of generality, suppose $u_t \in S$. Write the sequence as

$$K_2(u_0, v_1), K_3(v_1, w_1, u_1), K_2(u_1, v_2), K_3(v_2, w_2, u_2), \ldots, K_2(u_{t-1}, v_t), K_3(v_t, w_t, u_t).$$

In the sequence the last K_3-clique contains two vertices of S and the others contain exactly one vertex of S. Let

$$N = \{v_1, v_2, \ldots, v_t\},$$
$$N' = \{u_1, u_2, \ldots, u_{t-1}\}.$$

According to our construction, we have $N \subseteq S$ and $N' \cap S = \emptyset$. Let $S' = S \cup N' - N$. Clearly S' satisfies the Property (A) but $|S'| < |S|$, a contradiction. So we can continue the sequence without end, which is impossible as there are a limited number of K_3-cliques in G. So $|S \cap K_2| = 1$ for every K_2-clique K_2 in G. Therefore,

$$\tau_c(G) \leq |S| = |V' \cap S| + |(V - V') \cap S| = |V'|/2 + |V - V'|/2 = \frac{n}{2}.$$

From the above proof, it is easy to see that if G satisfies $V = V'$, i.e., $V - V' = \emptyset$, then $\tau_c(G) = \frac{n}{2}$ by Claim 1. Fig. 2 shows an example H_1 of a claw-free cubic graph satisfying $\tau_c(H_1) = \frac{n}{2}$ in which the darkened vertices indicated in Fig. 2 form a minimum clique-transversal set of H_1. □

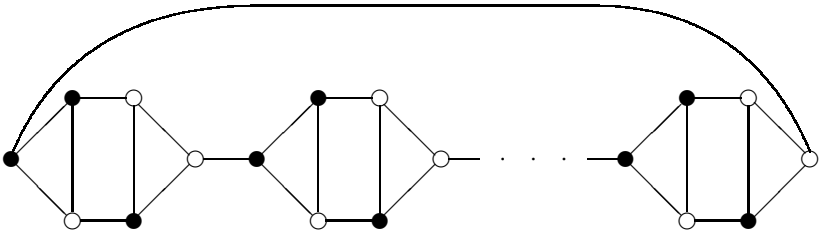


Fig. 2. The graph H_1 with $\tau_c(H_1) = \frac{n}{2}$

As an immediate consequence of Theorem 2, we have the following result, which shows that for a claw-free cubic graph G the clique-transversal number $\tau_c(G)$ and matching number $\alpha_1(G)$ are comparable.

Corollary 1. *For any claw-free cubic graph G, $\tau_c(G) \leq \alpha_1(G)$.*

In Section 3, however, we give an example H_2 shown in Fig. 3 of a cubic graph for which $\tau_c(H_2)$ is equal to $\frac{3}{5}|V(H_2)|$. This shows the above result is not true for arbitrary cubic graphs.

A graph G is called *weakly m-colorable* if its vertices can be colored with m colors such that G has no monochromatic cliques. Andreae et al. [3] observed that there is a straightforward relationship between weakly 2-colorable graphs

and the bound of $n/2$ for the clique-transversal number, namely $\tau_c(G) \leq n/2$ for each weakly 2-colorable graph of order n without isolated vertices, since both color-classes are clique-transversal sets and one of them must have cardinality no larger than $n/2$. Therefore, we have the following obvious corollary.

Corollary 2. *Every claw-free cubic graph G is weakly 2-colorable.*

3 Conclusion

In Section 2 we could not find an extremal graph G for which $\tau_c(G) = \frac{2}{3}n$, which means that the upper bound in Theorem 1 may be improved. We close this paper with the following problem.

Problem. Is it true that $\tau_c(G) \leq \lceil \frac{3}{5}n \rceil$ for cubic graphs ?

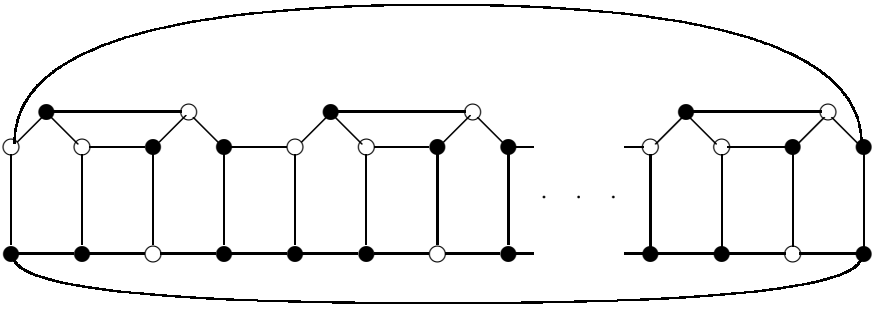


Fig. 3. The graph H_2 with $\tau_c(H_2) = \lceil \frac{3}{5}n \rceil$

If it is true, then the upper bound is sharp. It is easy to check that the Peterson Graph attains exactly the upper bound. Furthermore, an example H_2 is also shown in Fig. 3. It is not difficult to check that the darkened vertices of H_2 indicated in Fig. 3 form a minimum clique-transversal set of H_2.

References

1. Aigner, M., Andreae, T.: Vertex-sets that meet all maximal cliques of a graph, manusript (1986)
2. Andreae, T.: On the clique-transversal number of chordal graphs. Discrete Math. 191, 3–11 (1998)
3. Andreae, T., Schughart, M., Tuza, Z.: Clique-transversal sets of line graphs and complements of line graphs. Discrete Math. 88, 11–20 (1991)
4. Balachandhran, V., Nagavamsi, P., Pandu Rangan, C.: Clique transversal and clique independence on comparability graphs. Inform. Process. Lett. 58, 181–184 (1996)
5. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. Elsevier North Holland, New York (1986)

6. Chang, M.S., Chen, Y.H., Chang, G.J., Yan, J.H.: Algorithmic aspects of the generalized clique-transversal problem on chordal graphs. Discrete Appl. Math. 66, 189–203 (1996)
7. Chang, G.J., Farber, M., Tuza, Z.: Algorithmic aspects of neighbourhood numbers. SIAM J. Discrete Math. 6, 24–29 (1993)
8. Dahlhaus, E., Mannuel, P.D., Miller, M.: Maximum h-colourable subgraph problem in balanced graphs. Inform. Process. Lett. 65, 301–303 (1998)
9. Durán, G., Lin, M.C., Szwarcfiter, J.L.: On clique-transversals and clique-independent sets. Annals of Operations Research 116, 71–77 (2002)
10. Erdős, P., Gallai, T., Tuza, T.: Covering the cliques of a graph with vertices. Discrete Math. 108, 279–289 (1992)
11. Guruswami, V., Pandu Rangan, C.: Algorithmic aspects of clique-transversal and clique-independent sets. Discrete Appl. Math. 100, 183–202 (2000)
12. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: Fundamentals of Domination in Graphs. Marcel Dekker, New York (1998)
13. Lai, F., Chang, G.J.: An upper bound for the transversal numbers of 4-uniform hypergraphs. J. Combin. Theory Ser. B 50, 129–133 (1990)
14. Lee, C.M., Chang, M.S.: Distance-hereditary graphs are clique-perfect. Discrete Appl. Math. 154, 525–536 (2006)
15. Lonc, Z., Rival, I.: Chains, antichains and fibers. J. Combin. Theory Ser. A 44, 207–228 (1987)
16. Prisner, E.: Graphs with few cliques. In: Alavi, Y., Schwenk, A. (eds.) Graph Theory, Combinatorics and Applications. Proceedings of the 7th Quadrennial International Conference on the Theory and Applications, pp. 945–956. Wiley, Chichester, New York (1995)
17. Tuza, Z.: Covering all cliques of a graph. Discrete Math. 86, 117–126 (1990)
18. Xu, G.J., Shan, E.F., Kang, L.Y., Cheng, T.C.E.: The algorithmic complexity of the minus clique-transversal problem. Appl. Math. Comput. 189, 1410–1418 (2007)

On the $L(h,k)$-Labeling of Co-comparability Graphs*

Tiziana Calamoneri[1], Saverio Caminiti[1], Stephan Olariu[2], and Rossella Petreschi[1]

[1] Dipartimento di Informatica, Università degli Studi di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
{calamo, caminiti, petreschi}@di.uniroma1.it
[2] Department of Computer Science, Old Dominion University,
Norfolk,VA 23529-0162, U.S.A
olariu@cs.odu.edu

Abstract. Given two non negative integers h and k, an $L(h,k)$-*labeling* of a graph $G=(V,E)$ is a map from V to a set of labels such that adjacent vertices receive labels at least h apart, while vertices at distance at most 2 receive labels at least k apart. The goal of the $L(h,k)$-labeling problem is to produce a legal labeling that minimizes the largest label used. Since the decision version of the $L(h,k)$-labeling problem is NP-complete, it is important to investigate classes of graphs for which the problem can be solved efficiently.

Along this line of though, in this paper we deal with co-comparability graphs and two of its subclasses: interval graphs and unit-interval graphs. Specifically, we provide, in a constructive way, the first upper bounds on the $L(h,k)$-number of co-comparability graphs and interval graphs. To the best of our knowledge, ours is the first reported result concerning the $L(h,k)$-labeling of co-comparability graphs.

In the special case where $k=1$, our result improves on the best previously-known approximation ratio for interval graphs.

Keywords: $L(h,k)$-Labeling, co-comparability graphs, interval graphs, unit-interval graphs.

1 Introduction

Graph coloring is, without doubt, one of the most fertile and widely studied areas in graph theory, as evidenced by the list of solved and unsolved problems in Jensen and Toft's comprehensive book on graph coloring [22]. The classic problem of (vertex) coloring, asks for an assignment of non-negative integers

* This research was supported, in part, by the European Research Project *Algorithmic Principles for Building Efficient Overlay Computers* (AEOLUS). Most of the work reported here was performed while Professor Olariu visited with the Department of Computer Science, University of Rome "La Sapienza". Support through a Visiting Fellowship from the University of Rome "La Sapienza" is gratefully acknowledged.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 116–127, 2007.

(colors) to the vertices of a graph in such a way that adjacent vertices receive distinct colors. Of interest, of course, are assignments (colorings) that minimize the number of colors used.

In this paper we focus on a generalization of the classic vertex coloring problem – the so-called $L(h,k)$-*labeling problem* – that asks for the smallest λ for which it is possible to assign integer labels $\{0,\ldots,\lambda\}$ to the vertices of a graph in such a way that vertices at distance at most two receive colors at least k apart, while adjacent vertices receive labels at least h apart. The *span* of a $L(h,k)$-labeling is the difference between the maximum and the minimum label used. In the remainder of this work we shall follow established practice and refer to the largest label in an optimal $L(h,k)$-labeling for graph G as $\lambda_{h,k}(G)$.

We note that for $k=0$, the $L(h,k)$-labeling problem coincides with the usual vertex coloring; for $h=k$, we obtain the well-known 2-distance coloring, which is equivalent to the vertex coloring of the square of a graph.

The $L(h,k)$-labeling problem arises in many applications, including the design of wireless communication systems [20], radio channel assignment [8,21], data distribution in multiprocessor parallel memory systems [4,34], and scalability of optical networks [1,36], among many others.

The decision version of the vertex coloring problem is NP-complete in general [16], and it remains so for most of its variations and generalizations. In particular, it has been shown that the decision version of the $L(h,k)$-labeling problem is NP-complete even for $h=k=1$ [20,27]. Therefore, the problem has been widely studied for many particular classes of graphs. For a survey of recent results we refer the interested reader to [9].

In this paper we deal with co-comparability graphs and two of its subclasses: interval graphs and unit-interval graphs. The literature contains a plethora of papers describing applications of these graphs to such diverse areas as archaeology, biology, psychology, management and many others (see [18,17,23,29,30]).

In the light of their relevance to practical problems, it is somewhat surprising to note the dearth of results pertaining to the $L(h,k)$-labeling of these graph classes. For example, a fairly involved web search has only turned up no results on the $L(h,k)$-labeling of co-comparability graphs and, as listed below, only two results on the $L(h,k)$-labeling of interval graphs and unit-interval graphs.

- In [33] the special case $h=2$ and $k=1$ is studied; the author proves that $2\chi(G)-2 \leq \lambda_{2,1}(G) \leq 2\chi(G)$ for unit-interval graphs, where $\chi(G)$ is the chromatic number of G. In terms of the maximum degree Δ, as $\chi(G) \leq \Delta+1$, the upper bound becomes $\lambda_{2,1}(G) \leq 2(\Delta+1)$, and this value is very close to be tight, as the clique K_n, that is an interval graph, has $\lambda_{2,1}(K_n) = 2(n-1) = 2\Delta$.
- In [3] the authors present a 3-approximate algorithm for $L(h,1)$-labeling interval graphs and show that the same approximation ratio holds for the $L(h,k)$-labeling problem of unit-interval graphs.

One of our main contributions is to provide, in a constructive way, the first upper bounds on the $L(h,k)$-number of co-comparability and interval graphs. In

the special case where $k = 1$, our result improves on the best previously-known approximation ratio for interval graphs.

This remainder of the paper is organized as follows: Section 2 is devoted to definitions and a review of preliminary results; in particular we show that the $L(1,1)$-labeling problem is polynomially solvable for the three classes of graphs discussed in this work. Sections 3 and 4 focus, respectively, on the $L(h,k)$-labeling problem on co-comparability and interval graphs. Finally, Section 5 offers concluding remarks and open problems.

2 Preliminaries

The graphs in the work are simple, with no self-loops or multiple edges. We follow standard graph-theoretic terminology compatible with [18] and [5].

(a) (b)

Fig. 1. Illustrating two forbidden configurations

Vertex orderings have proved to be useful tools for studying structural and algorithmic properties of various graph classes. For example, Rose, Tarjan and Lueker [32] and Tarjan and Yannakakis [35] have used the well known simplicial ordering of the vertices of a chordal graph to obtain simple recognition and optimization algorithms for this class of graphs. To make this work as self-contained as possible, suffice it to say that a graph $G = (V, E)$ has a *simplicial ordering* if its vertices can be enumerated as $v_1, v_2, \ldots, v_n$ in such a way that for all subscripts $i,\ j,\ k$, with $1 \leq i < j < k \leq n$, the presence of the edges v_iv_k and v_jv_k implies the existence of the edge v_iv_j. Refer to Figure 1(a) for a forbidden configuration for a simplicial order.

Figure 1(b) illustrates a broader forbidden configuration that we shall refer to as the *umbrella*. Kratsch and Stewart [24] have shown that a graph is a *co-comparability graph* if and only if its vertices can be enumerated as $v_1, v_2, \ldots, v_n$ in such a way that for all subscripts $i,\ j,\ k$, with $1 \leq i < j < k \leq n$, the presence of the edges v_iv_k implies the presence of at least one of the edges and v_iv_k or v_iv_j. For alternate definitions of co-comparability graphs we refer to [19].

Later, Olariu [28] proved that a graph is an *interval graph* if and only if its vertices can be ordered as $v_1, v_2, \ldots, v_n$ in such a way that for all subscripts $i,\ j,\ k$, with $1 \leq i < j < k \leq n$, the presence of the edge v_iv_k implies the presence of the edge v_iv_j.

Finally, for the sake of completeness, we recall that Looges and Olariu [26] showed that a graph is a *unit-interval graph* if its vertices can be ordered as $v_1, v_2, \ldots, v_n$ in such a way that for all subscripts $i,\ j,\ k$, with $1 \leq i < j < k \leq n$, the presence of the edge v_iv_k implies the presence of the edges v_iv_j and v_jv_k.

The next proposition summarizes of previous discussion.

Proposition 1. *Let $G = (V, E)$ be a graph.*

1. *G is a co-comparability graph if and only if there exists an ordering of its vertices $v_0 < \ldots < v_{n-1}$ such that if $v_i < v_j < v_l$ and $(v_i, v_l) \in E$ then either $(v_i, v_j) \in E$ or $(v_j, v_l) \in E$ [24];*
2. *G is an interval graph if and only if there exists an ordering of its vertices $v_0 < \ldots < v_{n-1}$ such that if $v_i < v_j < v_l$ and $(v_i, v_l) \in E$ then $(v_i, v_j) \in E$ [28];*
3. *G is a unit-interval graph if and only there exists an ordering of its vertices $v_0 < \ldots < v_{n-1}$ such that if $v_i < v_j < v_l$ and $(v_i, v_l) \in E$ then $(v_i, v_j) \in E$ and $(v_j, v_l) \in E$ [26].*

In the remainder of this work we shall refer to a linear orders satisfying the above proposition as *canonical* and to the property that characterizes which edges must exist in a certain class as the *umbrella property* of that class. Also, Figure 2 summarizes the umbrella properties for co-comparability, interval, and unit-interval graphs. Observe that Proposition 1 confirms the well-known fact that unit-interval graphs $\subseteq$ interval graphs $\subseteq$ co-comparability graphs.

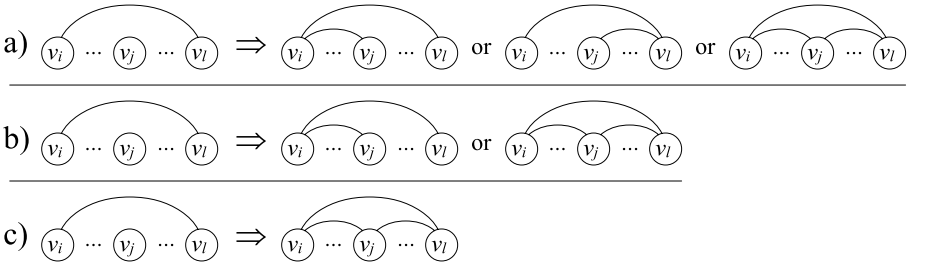


Fig. 2. Illustrating the umbrella properties for a) co-comparability, b) interval, and c) unit-interval graphs

Before proving general results concerning the $L(h,k)$-labeling of the above classes of graphs, we make a few observations about the corresponding $L(1,1)$-labelings. To begin, we observe that unit-interval, interval and co-comparability graphs are all perfect graphs and hence the vertex-coloring problem is polynomially solvable [18]. As already mentioned, the $L(1,1)$-labeling problem for a graph G is exactly the vertex-coloring problem for its square graph G^2 (i.e. the graph having the same vertex set as G and having an edge connecting u to v if and only if u and v are at distance at most 2 in G). Since all these classes are closed under powers [13,31], the following holds:

Remark 1. The $L(1,1)$-labeling problem is polynomially solvable for unit-interval, interval and co-comparability graphs.

3 The $L(h,k)$-Labeling of Co-comparability Graphs

Given a co-comparability graph $G = (V, E)$ of maximum degree Δ, in view of the umbrella property (Proposition 1 item 1), if $(v_i, v_l) \in E$ and $v_i < v_l$ then all the $l - i - 1$ vertices between v_i and v_l must be connected with at least one of these two vertices: d' are connected to v_i and d'' are connected to v_l, with $l - i - 1 \leq d' + d''$.

The degree, $d(v_i)$, of v_i is at least $d' + 1$, analogously $d(v_l) \geq d'' + 1$. Since the maximum degree is Δ we have $2\Delta \geq d' + d'' + 2 \geq l - i + 1$. Let us formalize this fact in the following proposition:

Proposition 2. *Given a co-comparability graph of maximum degree Δ, if there is an edge (v_i, v_l) such that $v_i < v_l$ then $l - i < 2\Delta$; if v_i and v_l are at distance 2 and $v_i < v_l$ then $l - i < 4\Delta$.*

Lemma 1. *A co-comparability graph G can be $L(h,k)$-labeled with span at most $2\Delta h + k$ if $k \leq \frac{h}{2}$.*

Proof. Let us consider the following ordered set of labels: $0, h, 2h, \ldots, 2\Delta h, k, h + k, 2h + k, \ldots, 2\Delta h + k$.

Let us label all vertices of G with labels in the given order following a canonical order of G's vertices; once the labels have been finished, we start again from label 0.

We will now prove that such a labeling is a feasible $L(h,k)$-labeling by showing that adjacent vertices are labeled with colors at least h apart and that vertices at distance 2 are labeled with colors at least k apart. The proofs are by contradiction and v_i and v_l are any two vertices with $i < l$.

Distance 1. Let v_i and v_l be adjacent vertices, assume by contradiction that their labels $l(v_i)$ and $l(v_l)$ differ by less than h. Then only two cases are possible:

(1.1) $l(v_i) = sh$, for some s such that $0 \leq s \leq 2\Delta$. Then $|l(v_l) - l(v_i)|$ can be smaller than h only if either $l(v_l) = sh + k$ or $l(v_l) = (s-1)h + k$. In both cases $l - i \geq (2\Delta - s) + (s-1) + 1 = 2\Delta$ as illustrated in Figure 3. This is impossible, for otherwise either v_i or v_l would have degree greater than Δ (see Proposition 2.) Notice that $l(v_l)$ cannot be sh because there are 4Δ distinct labels and $l - i$ is bounded by 2Δ.

(1.2) $l(v_i) = sh + k$, with $0 \leq s \leq 2\Delta$. Then $l(v_l)$ must be either sh or $(s+1)h$. In both cases $l - i \geq (2\Delta - s) + s + 1 = 2\Delta + 1$. Again, this is impossible. As in the previous case, $l(v_l)$ cannot be equal to $l(v_i)$.

Distance 2. Let v_i and v_l be at distance two with labels $l(v_i)$ and $l(v_l)$ that differ by less than k. Since $k \leq \frac{h}{2}$ it must be $l(v_i) = l(v_l)$, and therefore that $l - i = 4\Delta + 2$, i.e. the number of the different labels. This contradicts Proposition 2.

Lemma 2. *A co-comparability graph G can be $L(h,k)$-labeled with span at most $4k\Delta + k$, if $k \geq \frac{h}{2}$.*

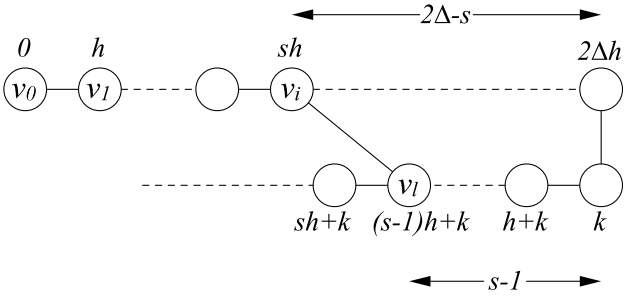


Fig. 3. Scheme for labeling vertices of a co-comparability graph

Proof. The proof is analogous to the one of Lemma 1. The only difference is the ordered set of labels used: $0, 2k, 4k, \ldots, 4k\Delta, k, 3k, 5k, \ldots, 4k\Delta + k$.

We can summarize both previous results in the following theorem:

Theorem 1. *A co-comparability graph G can be $L(h,k)$-labeled with span at most $2\Delta \max\{h, 2k\} + k$.*

4 The $L(h,k)$-Labeling of Interval Graphs

If the graph G is an interval graph, we can exploit its particular umbrella property to derive better bounds on $\lambda_{h,k}(G)$.

First observe that the degree of any vertex v_i connected to a vertex v_l, $v_i < v_l$, is at least $l - i$; furthermore, if $i \neq 0$ then the degree of v_i is at least $l - i + 1$ if G is connected, because at least one edge must reach v_i from vertices preceding it in the ordering.

Proposition 3. *Given a connected interval graph of maximum degree Δ, if $(v_i, v_l) \in E$ and $v_i < v_l$ then $l - i \leq \Delta$ and, if $i \neq 0$ then $l - i < \Delta$; if v_i and v_l are at distance 2 and $v_i < v_l$ then $l - i \leq 2\Delta - 1$.*

Lemma 3. *An interval graph G can be $L(h,k)$-labeled with span at most Δh, if $k \leq \frac{h}{2}$.*

Proof. Without loss of generality, we focus on connected graphs. We proceed as in Lemma 1 with the difference that the set of labels is $0, h, 2h, \ldots, \Delta h, k, h + k, 2h + k, \ldots, (\Delta - 1)h + k$.

Distance 1. Let v_i and v_l be adjacent vertices, assume by contradiction that their labels $l(v_i)$ and $l(v_l)$ differ by less than h. Then only two cases are possible:

(1.1) $l(v_i) = sh$, for some s, and $l(v_l)$ is either $sh + k$ or $(s-1)h + k$. Then $l - i \geq (\Delta - s) + (s-1) + 1 = \Delta$. In view of Proposition 3 this is impossible because G has maximum degree Δ. If $i = 0$ then l can be at most Δ; hence, $l(v_l) - l(v_i)$ is never smaller than h.

(1.2) $l(v_i) = sh + k$, for some s, and $l(v_l)$ is either sh or $(s+1)h$. Then i cannot be 0 and $l - i \geq (\Delta - 1 - s) + s + 1 = \Delta$. This is in contradiction with Proposition 3.

Distance 2. Let v_i and v_l be vertices at distance two with labels $l(v_i)$ and $l(v_l)$ that differ by less than k. Since $k \leq \frac{h}{2}$ it must be $l(v_i) = l(v_l)$, and therefore $l - i = 2\Delta + 1$, i.e. the number of the different labels. This contradicts Proposition 3.

From the previous proof it easily follows:

Corollary 1. *If an interval graph G has a canonical order such that the degree of v_0 is strictly less than Δ, G can be $L(h,k)$-labeled with span at most $(\Delta-1)h+k$, if $k \leq \frac{h}{2}$.*

The bound stated in the previous lemma is the best possible, as shown by the following:

Theorem 2. *There exists an interval graph requiring at least span Δh to be $L(h,k)$-labeled.*

Proof. Consider $K_{\Delta+1}$, the clique on $\Delta + 1$ vertices. As all vertices are adjacent a span of Δh is necessary.

Lemma 4. *An interval graph G can be $L(h,k)$-labeled with span at most $2k\Delta$, if $k \geq \frac{h}{2}$.*

Proof. The proof is analogous to the one of Lemma 3. The only difference is the ordered set of labels used: $0, 2k, 4k, \ldots, 2k\Delta, k, 3k, 5k, \ldots, 2k(\Delta - 1) + k$.

Again, it easily follows:

Corollary 2. *If the canonical order of an interval graph G is such that the degree of v_0 is strictly less than Δ, G can be $L(h,k)$-labeled with span at most $2k(\Delta - 1) + k$, if $k \geq \frac{h}{2}$.*

Unfortunately, we are not able to exhibit an interval graph requiring at least span $2k\Delta$, if $k \geq \frac{h}{2}$, so it remains an open problem to understand if this result is tight or not.

We can summarize the previous results in the following theorem:

Theorem 3. *An interval graph G can be $L(h,k)$-labeled with span at most $\max(h, 2k)\Delta$ and, if G has a canonical order such that the degree of v_0 is strictly less than Δ, G can be $L(h,k)$-labeled with span at most $max(h, 2k)(\Delta - 1) + k$.*

Next theorem allows us to compute another bound for $\lambda_{h,k}(G)$, introducing also the dimension of the maximum clique ω.

Theorem 4. *An interval graph G can be $L(h,k)$-labeled with span at most $min((\omega - 1)(2h + 2k - 2), \Delta(2k - 1) + (\omega - 1)(2h - 2k))$.*

Proof. Consider the following greedy algorithm that generalizes the coloring algorithm for interval graphs:

ALGORITHM `Greedy-Interval`
consider the canonical order $v_0, v_1, \ldots v_{n-1}$
FOR $i = 0$ TO $n-1$ DO
label v_i with the first available label, taking into account
the labels already assigned to neighbors of v_i
and to vertices at distance 2 from v_i.

At the i-th step of this $O(n^2)$ time algorithm, consider the vertex set C_i constituted by all the labeled neighbors of v_i and the vertex set D_i constituted by all the labeled vertices at distance 2 from v_i. It is straightforward that $C_i \cap D_i = \emptyset$. As an example consider the graph of Figure 4, when $i = 3$ we have $C_3 = \{v_1, v_2\}$ and $D_3 = \{v_0\}$.

Let v_{min} be the vertex in C_i with the minimum index; in view of the umbrella property for interval graphs, v_{min} is connected to all vertices inside C_i. On the other hand, each vertex v_k in D_i must be adjacent to some vertex in C_i, as it is at distance 2 from v_i; therefore the umbrella property implies that all vertices in D_i are connected to v_{min}. It follows that $\Delta \geq d(v_{min}) \geq |D_i| + (|C_i| - 1) + 1$.

Observe also that both $C_i \cup \{v_i\}$ and $D_i \cup \{v_{min}\}$ are cliques, and hence $|C_i| \leq \omega - 1$ and $|D_i| \leq \omega - 1$.

So, when vertex v_i is going to be labeled, each labeled vertex in C_i forbids at most $2h-1$ labels and each labeled vertex in D_i forbids at most $2k-1$ labels. Hence the number f of forbidden labels is at most $|C_i|(2h-1) + |D_i|(2k-1)$. About f we can also say:

$f \leq (\omega-1)(2h+2k-2)$ due to the inequalities $|C_i| \leq \omega-1$ and $|D_i| \leq \omega-1$;
$f \leq (|C_i| + |D_i|)(2k-1) + |C_i|2(h-k) \leq \Delta(2k-1) + (\omega-1)(2h-2k)$ for the inequalities $\Delta \geq |D_i| + |C_i|$ and $|C_i| \leq \omega - 1$.

As the previous reasoning does not depend on i, the maximum span is bounded by $\min((\omega-1)(2h+2k-2), \Delta(2k-1) + (\omega-1)(2h-2k))$.

In Figure 4 it is shown a graph $L(2,1)$-labeled with the greedy algorithm. It is easy to see that in this case the bounds given in the previous theorem, arguments of the min function, coincide and are exactly equal to the required span.

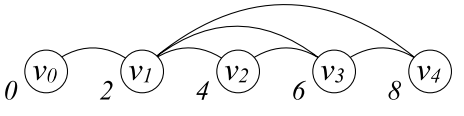


Fig. 4. A graph $L(2,1)$-labeled with the greedy algorithm

Observe that a trivial lower bound for $\lambda_{h,k}(G)$ is $(\omega-1)h$. So, when $k = 1$ the previous theorem provides a 2-approximate algorithm for interval graphs, improving the approximation ratio of [3].

5 Concluding Remarks and Open Problems

In the literature there are no results concerning the $L(h,k)$-labeling of general co-comparability graphs. It is neither known whether the problem remains NP-complete when restricted to this class or to some subclasses, as interval or unit-interval graphs.

In this paper we offered the first known upper bounds for $\lambda_{h,k}$ of co-comparability and interval graphs. The presented proofs are constructive and give the following upper bounds:

$$\lambda_{h,k}(G) \leq \max(h, 2k)2\Delta + k$$

if G is a co-comparability graph, and

$$\lambda_{h,k}(G) \leq \max(h, 2k)\Delta$$

if G is an interval graph.

Moreover, for interval graphs with certain restrictions, we have reduced this latter bound to $\max(h, 2k)(\Delta - 1) + k$. We have also shown a greedy algorithm that guarantees, for all interval graphs, a new upper bound on $\lambda_{h,k}(G)$ in terms of both ω and Δ, that is:

$$\lambda_{h,k}(G) \leq \min((\omega - 1)(2h + 2k - 2), \Delta(2k - 1) + (\omega - 1)(2h - 2k))$$

This bound is provided by a 2-approximate algorithm, improving the approximation ratio in [3].

Finally, we have shown that the $L(1,1)$-labeling problem is polynomially solvable for co-comparability graphs.

Many open problems are connected to this research. Here we list just some of them:

- Is the $L(2,1)$-labeling polynomially solvable on co-comparability graphs?
- Is it possible to find some lower bounds to understand how much our results are tight?
- Circular-arc graphs are a natural super-class of interval graphs. Is it possible to extend the results achieved in this paper in order to find an $L(h,k)$-labeling of circular-arc graphs? What about the complexity of the $L(1,1)$-labeling on circular-arc graphs?
- What can we say about the $L(h,k)$-labeling of comparability graphs? It is easy to see that their $L(1,1)$-labeling is polynomially solvable as they are perfect and the square of a comparability graph is still a comparability graph; does the $L(2,1)$-labeling remain polynomially solvable?

Last, but not least, we wish to point out the connection between the linear orderings of co-comparability, interval and unit-interval graphs with a more general concept, namely that of a *dominating pair*, introduced by Corneil, Olariu

and Stewart [11]. Considerable attention has been paid to exploiting the linear structure exhibited by various graph families. Examples include interval graphs [25], permutation graphs [15], trapezoid graphs [10,14], and co-comparability graphs [19].

The linearity of these four classes is usually described in terms of ad-hoc properties of each of these classes of graphs. For example, in the case of interval graphs, the linearity property is traditionally expressed in terms of a linear order on the set of maximal cliques [6,7]. For permutation graphs the linear behavior is explained in terms of the underlying partial order of dimension two [2], for co-comparability graphs the linear behavior is expressed in terms of the well-known linear structure of comparability graphs [24], and so on.

As it turns out, the classes mentioned above are all subfamilies of a class of graphs called the asteroidal triple-free graphs (AT-free graphs, for short). An independent set of three vertices is called an *asteroidal triple* if between any pair in the triple there exists a path that avoids the neighborhood of the third. AT-free graphs were introduced over three decades ago by Lekkerkerker and Boland [25] who showed that a graph is an interval graph if and only if it is chordal and AT-free. Thus, Lekkerkerker and Boland's result may be viewed as showing that the absence of asteroidal triples imposes the linear structure on chordal graphs that results in interval graphs. Recently, the authors [11] have studied AT-free graphs with the stated goal of identifying the "agent" responsible for the linear behavior observed in the four subfamilies. Specifically, in [11] the authors presented evidence that the property of being asteroidal triple-free is what is enforcing the linear behavior of these classes.

One strong "certificate" of linearity is the existence of a *dominating pair* of vertices, that is, a pair of vertices with the property that every path connecting them is a dominating set. In [11], the authors gave an existential proof of the fact that every connected AT-free graph contains a dominating pair.

In an attempt to generalize the co-comparability ordering while retaining the AT-free property, Corneil, Koehler, Olariu and Stewart [12] introduced the concept of *path orderable graphs*. Specifically, a graph $G = (V, E)$ is path orderable if there is an ordering $v_1, \ldots, v_n$ of its vertices such that for each triple v_i, v_j, v_k with $i < j < k$ and $v_i v_k \notin E$, vertex v_j intercepts each v_i, v_k-path of G; such an ordering is called a *path ordering*.

It is easy to confirm that co-comparability graphs are path orderable. It is also clear that path orderable graphs must be AT-free. It is a very interesting open question whether the results in this paper about the $L(h,k)$-labeling of co-comparability graph can be extended to

- graphs that have an induced dominating pair, and/or
- graphs that are path orderable.

This promises to be an exciting area for further investigation.

References

1. Aly, K.A., Dowd, P.W.: A class of scalable optical interconnection networks through discrete broadcast-select multi-domain WDM. In: Proc. IEEE INFOCOM, pp. 392–399. IEEE Computer Society Press, Los Alamitos (1994)
2. Baker, K.A., Fishburn, P.C., Roberts, F.S.: Partial orders of dimension two. Networks 2, 11–28 (1971)
3. Bertossi, A.A., Pinotti, C.M., Rizzi, R.: Channel assignment on strongly-simplicial graphs. In: IPDPS '03. Proc. of Int. l Parallel and Distributed Processing Symposium, 222b (2003)
4. Blelloch, G.E., Gibbons, P.B., Mattias, Y., Zagha, M.: Accounting for memory bank contentions and delay in high-bandwidth multiprocessors. IEEE Trans. on Parallel and Distributed Systems 8, 943–958 (1997)
5. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. North-Holland, Amsterdam (1976)
6. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms. Journal of Comput. Syst. Sci. 13, 335–379 (1976)
7. Booth, K.S., Lueker, G.S.: A linear time algorithm for deciding interval graph isomorphism. Journal of the ACM 26, 183–195 (1979)
8. Calamoneri, T.: Exact Solution of a Class of Frequency Assignment Problems in Cellular Networks and Other Regular Grids. Discrete Mathematics & Theoretical Computer Science 8, 141–158 (2006)
9. Calamoneri, T.: The $L(h,k)$-labelling problem: a survey. The Computer Journal 49(5), 585–608 (2006), A continuously updated version is available on `http://www.dsi.uniroma1.it/~calamo/survey.html`
10. Corneil, D.G., Kamula, P.A.: Extensions of permutation and interval graphs. In: Proceedings 18th Southeastern Conference on Combinatorics, Graph Theory and Computing, pp. 267–276 (1987)
11. Corneil, D.G., Olariu, S., Stewart, L.: Asteroidal triple-free graphs. SIAM Journal on Discrete Mathematics 10, 399–430 (1997)
12. Corneil, D.G., Koehler, E., Olariu, S., Stewart, L.: On Subfamilies of AT-Free Graphs. SIAM Journal on Discrete Mathematics 20(1), 241–253 (2006)
13. Damaschke, P.: Distance in cocomparability graphs and their powers. Disc. Applied Math. 35, 67–72 (1992)
14. Degan, I., Golumbic, M.C., Pinter, R.Y.: Trapezoid graphs and their coloring. Discrete Applied Mathematics 21, 35–46 (1988)
15. Even, S., Pnueli, A., Lempel, A.: Permutation graphs and transitive graphs. Journal of the ACM 19, 400–410 (1972)
16. Garey, M.R., Johnson, D.S.: Computers and Intractability - A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., San Francisco (1979)
17. Goldberg, P.W., Golumbic, M.C., Kaplan, H., Shamir, R.: Four strikes against physical mapping of DNA. Journal of Computational Biology 2, 139–152 (1995)
18. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, New York (1980)
19. Golumbic, M.C., Monma, C.L., Trotter Jr., W.T.: Tolerance graphs. Discrete Applied Mathematics 9, 157–170 (1984)
20. Griggs, J.R., Yeh, R.K.: Labeling graphs with a condition at distance 2. SIAM Journal of Discrete Mathematics 5, 586–595 (1992)

21. van den Heuvel, J., Leese, R.A., Shepherd, M.A.: Graph Labelling and Radio Channel Assignment. Journal of Graph Theory 29, 263–283 (1998)
22. Jensen, T.R., Toft, B.: Graph Coloring Problems. John Wiley & Sons, New York (1995)
23. Karp, R.M.: Mapping the genome: some combinatorial problems arising in molecular biology. In: STOC '93. Proc. 25th Ann. ACM Symp. on Theory of Comp., pp. 278–285. ACM Press, New York (1993)
24. Kratsch, D., Stewart, L.: Domination on cocomparability graphs. SIAM Journal on Discrete Mathematics 6, 400–417 (1993)
25. Lekkerkerker, C.G., Boland, J.C.: Representation of a finite graph by a set of intervals on the real line. Fundamenta Mathematicae 51, 45–64 (1962)
26. Looges, P., Olariu, S.: Optimal Greedy Algorithms for Indifference Graphs. Computers and Mathematics with Application 25, 15–25 (1993)
27. McCormick, S.T.: Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem. Mathematical Programming 26, 153–171 (1983)
28. Olariu, S.: An optimal greedy heuristic to color interval graphs. Information Processing Letters 37(1), 21–25 (1991)
29. Pe'er, I., Shamir, R.: Interval graphs with side (and size) constraints. In: Spirakis, P.G. (ed.) ESA 1995. LNCS, vol. 979, pp. 142–154. Springer, Heidelberg (1995)
30. Pe'er, I., Shamir, R.: Realizing interval graphs with side and distance constraints. SIAM Journal of Discrete Mathematics 10, 662–687 (1997)
31. Raychauduri, A.: On powers of interval and unit interval graphs. Conressus Numererantium 59, 235–242 (1987)
32. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM Journal on Computing 5, 266–283 (1976)
33. Sakai, D.: Labeling chordal graphs: distance two condition. SIAM Journal of Discrete Mathematics 7, 133–140 (1994)
34. Shapiro, H.D.: Theoretical limitations on the efficient use of parallel memories. IEEE Transactions on Computers 17, 421–428 (1978)
35. Tarjan, R.E., Yannakakis, M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. SIAM Journal on Computing 13, 566–579 (1984)
36. Wan, P.J.: Near-optimal conflict-free channel set assignments for an optical cluster-based hypercube network. Journal of Combinatorial Optimization 1, 179–186 (1997)

An Approximation Algorithm for the General Mixed Packing and Covering Problem*

Florian Diedrich** and Klaus Jansen

Institut für Informatik, Christian-Albrechts-Universität zu Kiel,
Olshausenstr. 40, 24098 Kiel, Germany
{fdi,kj}@informatik.uni-kiel.de

Abstract. We present a price-directive decomposition algorithm to compute an approximate solution of the mixed packing and covering problem; it either finds $x \in B$ such that $f(x) \leq c(1+\epsilon)a$ and $g(x) \geq (1-\epsilon)b/c$ or correctly decides that $\{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset$. Here f, g are vectors of $M \geq 2$ convex and concave functions, respectively, which are nonnegative on the convex compact set $\emptyset \neq B \subseteq \mathbb{R}^N$; B can be queried by a feasibility oracle or *block solver*, a, $b \in \mathbb{R}^M_{++}$ and c is the block solver's approximation ratio. The algorithm needs only $O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}))$ iterations, a runtime bound independent from c and the input data. Our algorithm is a generalization of [16] and also approximately solves the fractional packing and covering problem where f, g are linear and B is a polytope; there, a *width-independent* runtime bound is obtained.

1 Introduction

We study the approximate general mixed packing and covering problem

$$\begin{array}{c}\text{compute } x \in B \text{ such that } f(x) \leq c(1+\epsilon)a, \quad g(x) \geq (1-\epsilon)b/c \\ \text{or correctly decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset\end{array} \qquad (MPC_{c,\epsilon})$$

where $\emptyset \neq B \subseteq \mathbb{R}^N$ is a convex compact set, $f, g : B \to \mathbb{R}^M_+$ are vectors of convex and concave functions, respectively, which are nonnegative on B; $a, b \in \mathbb{R}^M_{++}$ are positive vectors. Note that $a = e = b$ holds without loss of generality where $e \in \mathbb{R}^M_+$ denotes the unit vector; otherwise, similar to [16,24], we replace f_m byf_m/a_m and g_m by g_m/b_m. Let $[M] := \{1, \ldots, M\}$ and $c \geq 1$ is a bound for the approximation ratio of the *block solver*, an algorithm which queries B by an approximate feasibility oracle of the form

$$\begin{array}{c}\text{find } \hat{x} \in B \text{ such that } p^T f(\hat{x})/Y(c,t) - q^T g(\hat{x}) Y(c,t) \leq \alpha \\ \text{or correctly decide that there is no such } x \in B\end{array} \qquad (ABS_c(p,q,\alpha,t))$$

* Research supported in part by DFG Project "Entwicklung und Analyse von Approximativen Algorithmen für Gemischte und Verallgemeinerte Packungs- und Überdeckungsprobleme" JA 612/10-1, in part by EU Project AEOLUS IST-15964, and in part by a PPP funding "Scheduling in Communication Networks" D/05/06936 of the DAAD.

** Research supported in part by a grant "DAAD Doktorandenstipendium" of the DAAD. Part of this work done while visiting the ID-IMAG, ENSIMAG, Grenoble.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 128–139, 2007.

where $t \in (0,1)$, $Y(c,t) := c(1+8/3t)(1+t)$ is a parameter defined for ease of exposition, $p, q \in \mathbb{R}_+^M$ such that $e^T p + e^T q = 1$ and α depends on p and q as defined later. In contrast to [13,14,18], $ABS_c(p,q,\alpha,t)$ solves only a feasibility problem and not an optimization problem; however, this can be done by minimizing a convex function over B. Our measure of runtime complexity is the number of iterations or *coordination steps*, which is the number of calls to the block solver. Price-directive decomposition algorithms have several potential advantages in comparision to classical methods – they can be faster, easier to implement, and often result in column-generation algorithms which can efficiently solve models with an exponential number of variables. Applications for $(MPC_{c,\epsilon})$ are multicommodity flow [24], capacitated network design with fixed total cost [6,8] and the network access regulation problem [2,3]. See Fig. 1 for some intuition behind our approach.

Related problems and previous results. Related problems are so-called *pure* packing and covering problems. Approximation algorithms for *pure packing problems* with convex functions were studied in [11,12,22] where finally a runtime bound of $O(M(\epsilon^{-2}\ln\epsilon^{-1} + \ln M))$ is obtained. Garg & Könemann [9] found a width-independent algorithm with runtime bound $O(M\epsilon^{-2}\ln M)$ for the packing problem with linear constraints. Bienstock & Iyengar [5] described an algorithm for the same problem with runtime bound $O^*(\epsilon^{-1}\sqrt{KN})$ where K is the maximum number of non-zeros per row and in each iteration a quadratic program has to be solved. Similarly, Chudak & Eleutério [7] found an algorithm with coordination complexity $O(\sqrt{N\ln M}(\log\log\sqrt{N} + \epsilon^{-1}))$. Jansen & Zhang [18] presented an algorithm that parallels the result in [13]. They obtain $O(M(\epsilon^{-2}\ln\epsilon^{-1} + \ln M))$ as a runtime bound, although a block solver with ratio c is used; for block solvers with arbitrary precision the runtime is improved to $O(M(\epsilon^{-2} + \ln M))$, matching the bound from [13]. Experiments with this algorithm applied to the multicast congestion problem were done in [21]. For *pure covering problems* with concave functions, in [13] a runtime bound of $O(M(\epsilon^{-2} + \ln M))$ iterations is obtained; here a strong block solver with arbitrarily high precision is needed. This was generalized in [17] where a general block solver is used, resulting in $O(M(\ln M + \epsilon^{-2}\ln c + \epsilon^{-2}))$ coordination steps; however, this bound depends on the ratio c of the block solver. In [14], this shortcoming was removed by presenting an improved algorithm with runtime bound $O(M(\epsilon^{-2}\ln\epsilon^{-1} + \ln M))$ which matches the best known bound for solving the fractional covering problem with a similar approach in [18]. In [1] experiments with the algorithm from [13] were done, where two-dimensional strip-packing

Fig. 1. Comparison of (MPC) with the relaxed version $(MPC_{c,\epsilon})$ where $B \subseteq \mathbb{R}^2$ is a rectangle with 3 linear constraints; hatched areas indicate the feasible regions

from [15] was used as a testbed. Finally, *mixed packing and covering* problems have been studied. An important result here is [24], where with a different technique a running time of $O(Md\epsilon^{-2}\log M)$ was obtained; d is the maximum number of constraints any variable appears in. Here the case with linear f, g and $B = \mathbb{R}^N_+$ is solved; this can be generalized since a polytope B with N vertices can be reduced to $B' = \mathbb{R}^N_+$ by using a variable $x'_i \geq 0$ for each vertex v_i of B that denotes its coefficient in a convex combination of an arbitrary $x \in B$ and studying the problem of finding $x' \in \mathbb{R}^N_+$ such that $\mathcal{P}x' \leq e$, $\mathcal{C}x' \geq e$, $e^T x' \leq 1$ and $e^T x' \geq 1$ where $\mathcal{P}, \mathcal{C}$ are suitable modifications of the packing and covering constraints, respectively [23]. However [24] mentions the case of general B as an open problem which was later solved in [10,16,23]. Garg & Khandekar [10,19] proposed an algorithm for our problem with runtime bound $O(M\epsilon^{-2}\ln M)$; they used the *exponential* potential function and a feasibility oracle that solves the so-called *on-line prediction problem*. We refer the reader to [4] for a survey on the technique and recent theoretical and practical results; however, there the focus is on the *exponential* potential function resulting in different details.

New result. We contribute a generalization of [16]. There, the problem is solved with a more complex block solver. Our algorithm uses an approximate block solver; it solves each instance of $(MPC_{c,\epsilon})$ in only $O(M(\ln M + \epsilon^{-2}\ln\epsilon^{-1}))$ iterations.

Main ideas. To obtain our result, different techniques from the existing body of literature are put into effect. We use a technical improvement of the class of modified logarithmic potential functions from [13,14,16,17] and the resulting price vectors to govern the direction of optimization for the block solver. We use elimination of indices of covering constraints similar to [14], where we later prove suitable bounds for the values of the covering functions. We use a type of stopping rule similar to [14,16,18] for early termination of scaling phases. The usage of a more general block solver is motivated by the results from [14,18]. Finally, we prove that infeasibility of certain instances of the block problem implies infeasiblity of the instance.

In Sect. 2 we discuss the basic techniques. In Sect. 3 we describe the algorithmic details. In Sect. 4 we bound the number of iterations and study the behaviour of the eliminated covering functions; we conclude in Sect. 5.

2 Basic Techniques

In this section we present the used modified logarithmic potential function and its basic properties. As in [13,18] we use notational abbrevations. For $x, \hat{x}$, and $x' \in B$ we use $f, \hat{f}$ and f' to denote the evalutaions of f; $g, \hat{g}$ and g' are defined in a similar way. The algorithm will use sets $A \subseteq [M]$ to define active sets of indices associated with each iterate for the covering functions; these will be denoted as A and A'. In each scaling phase a component g_m of g is eliminated if $g_m \geq T$ where T is a threshold value; for $T \in \mathbb{R}_+$ and $x \in B$ let $A \subseteq [M]$ be the corresponding set of indices of active functions. Furthermore $t \in (0, 1)$ is an additional

accuracy parameter similar to [13] that will be changed in each scaling phase. Here we write $A := A(x,T)$ for short. We use the *potential function*

$$\Phi_t(\theta,x,A) := 2\ln\theta - \frac{t}{CM}[\sum_{m=1}^{M}\ln(\theta - f_m) + \sum_{m\in A}\ln(g_m - \frac{1}{\theta}) + (M-|A|)\ln T]$$

where $C = 8$ is a constant. For fixed $x \in B$, the potential function is defined for $\theta \in (\lambda_A(x),\infty)$ where

$$\lambda_A(x) := \max\{\max_{m\in[M]} f_m, \max_{m\in A} 1/g_m\}.$$

If there is an $m \in A$ such that $g_m = 0$ we let $\lambda_A(x) := \infty$; furthermore denote $\lambda(x) := \lambda_{[M]}(x)$. Additionally we define the *reduced potential function* by $\phi_t(x,A) := \min\{\Phi_t(\theta,x,A)|\theta \in (\lambda_A(x),\infty)\}$ and denote $\lambda_A := \lambda_A(x)$ for short where the dependency is clear. Notice that the corresponding minimizer $\theta \in (\lambda_A(x),\infty)$ can be determined from the first-order optimality condition

$$\frac{t}{2CM}[\sum_{m=1}^{M}\frac{\theta}{\theta - f_m} + \frac{1}{\theta}\sum_{m\in A}\frac{1}{g_m - 1/\theta}] = 1 \tag{1}$$

which can be seen by calculating the derivation with respect to θ of the right hand side of the definition of the potential function. For any fixed $x \in B$ the left-hand side of (1) is strictly monotonically decreasing in $\theta \in (\lambda_A,\infty)$. Hence the minimizer $\theta \in (\lambda_A,\infty)$ is uniquely determined; we denote it by $\theta_A(x)$ and remark that it can be approximated arbitrarily close by binary search. Similar as above, we use θ,θ' and λ,λ' to denote the corresponding minimizers and evalutations of λ for $x,x' \in B$ when the dependency is clear. By Lemma 1, $\theta_A(x)$ approximates λ_A for small values of t; the proof is parallel to the one of Lemma 2.1 in [16].

Lemma 1. $\theta/[1+t/(2CM)] \geq \lambda_A \geq \theta(1-t(M+|A|)/(2CM)] \geq \theta(1-t/C)$.

Lemma 2. *If $g_m \leq T$ for each $m \in A$, then $\phi_t(x,A) \geq 2\ln\theta - t/C\ln(\theta T)$ is satisfied. Furthermore the inequality*

$$\phi_t(x,A) \leq 2\ln\theta - \frac{t(M-|A|)}{CM}\ln(\theta T) - \frac{t(M+|A|)}{CM}\ln[\frac{t(M+|A|)}{2CM}]$$

is valid.

The proof of Lemma 2 is omitted due to space constraints. We define the price vectors in order to ensure that the block solver optimizes in a suitable direction. As in [13,14,16,18] the price vectors are obtained from (1) in a natural way; for $x \in B$, $A \subseteq [M]$ define

$$p_m(x,A) := \frac{t\theta}{2CM(\theta - f_m)}, \quad q_m(x,A) := \frac{t}{2CM(g_m\theta - 1)} \tag{2}$$

for $m \in [M]$ and $m \in A$, respectively, and $p_m(x,A) := 0$ for each $m \in [M] \setminus A$. The components of $p(x,A)$ and $q(x,A)$ are the summands in (1); hence the

entries are nonnegative and we have $e^T p + e^T q = 1$. If the dependency is clear, we write $\bar{p} := e^T p \leq 1$ and $\bar{q} := e^T q \leq 1$. The proof for the next lemma is omitted; it is similar to the one of Lemma 2.3 in [16].

Lemma 3. *Denoting $p := p(x, A)$ and $q := q(x, A)$, we have*

$$p^T f = \theta[\bar{p} - t/(2C)] \leq \theta[1 - t/(2C)] \text{ and}$$
$$q^T g = [\bar{q} + t|A|/(2CM)]/\theta \leq [\bar{q} + t/(2C)]/\theta \leq [1 + t/(2C)]/\theta.$$

3 The Algorithm

First we discuss what happens if one of the instances of the block problem that we are about to generate renders infeasible. Note that each invocation uses $p, q \in \mathbb{R}^M_+$, $t \in (0, 1/8]$ and $\alpha := 2\bar{p} - 1 - 2t$ and the block solver either finds $\hat{x} \in B$ such that $p^T f(\hat{x})/Y(c,t) - q^T g(\hat{x}) Y(c,t) \leq \alpha$ or correctly decides that no such $\hat{x} \in B$ exists. If the original instance is feasible, there is an $x \in B$ such that $f(x) \leq e$ and $g(x) \geq e$; let x be chosen as such. Then for each $p, q \in \mathbb{R}^M_+$ such that $\bar{p} + \bar{q} = 1$ we have $p^T f(x) \leq \bar{p}$ and $q^T g(x) \geq \bar{q} = 1 - \bar{p}$. Furthermore, writing $Y := Y(c,t)$, we have

$$p^T f(x)/Y - q^T g(x) Y \leq \bar{p}/(1 + 8/3t) - (1 - \bar{p})(1 + 8/3t) \leq 2\bar{p} - 1 - 2t$$

where for the last step note that $\bar{p} - (1 - \bar{p})(1 + 8/3t)^2 \leq (2\bar{p} - 1 - 2t)(1 + 8/3t)$ is equivalent to $64/9\bar{p}t^2 \leq 2/3t + 16/9t^2$ and the latter holds since $t \in (0, 1/8]$. This means that in each case in which the block solver reports infeasibility, our algorithm terminates and reports that the initial instance is infeasible.

The initial solution $x^{(0)} \in B$ is computed as follows. First we generate M solutions $x^{[1]}, \ldots, x^{[M]}$ by calling $ABS_c(p, q, \alpha, t)$ with $p := 1/(3M)e \in \mathbb{R}^M_+$, $q_m := 2/3$, $q_i := 0$ for each $i \in [M] \setminus \{m\}$ and $\alpha := -1/3 - 2t$ such that

$$\frac{1}{3MY(c,t)} \sum_{\ell=1}^{M} f_\ell(x^{[m]}) \leq \frac{2Y(c,t)}{3} g_m(x^{[m]}) - \frac{1}{3} - 2t$$

for each $m \in [M]$ and a solution $x^{[0]}$ via $ABS_c(p, q, \alpha, t)$ with $p := 1/Me \in \mathbb{R}^M_+$, $q := 0 \in \mathbb{R}^M_+$ and $\alpha := 1 - 2t$ such that

$$\frac{1}{MY(c,t)} \sum_{\ell=1}^{M} f_\ell(x^{[0]}) \leq 1 - 2t.$$

Then we compute a convex combination

$$x^{(0)} := \sum_{\ell=0}^{M} \mu_\ell x^{[\ell]} \in B; \text{ precisely let } I := \{m \in [M] | \sum_{\ell=1}^{M} f_\ell(x^{[m]}) > 2cM\}$$

and set

$$\mu_\ell := \frac{c}{\sum_{m=1}^{M} f_m(x^{[\ell]})} \leq \frac{1}{M+1} \text{ for each } \ell \in I. \text{ Finally we set}$$

$$\mu_\ell := \frac{1 - \sum_{\ell \in I} \mu_\ell}{M + 1 - |I|} \geq \frac{1}{M+1} \text{ for each } \ell \in ([M] \cup \{0\}) \setminus I.$$

Lemma 4 asserts a quality bound; the proof is omitted for space reasons.

Lemma 4. *If the instance is feasible and $t \leq 1/8$, we have $\lambda(x^{(0)}) \leq 9cM/2$.*

Now we present the algorithm itself. The remainder of Sect. 3 concerns itself with showing what happens if one of the three stopping rules is satisfied and how the step length τ is chosen and reduced in Steps 2.3.4 and 2.3.5 of Algorithm A. As in [13,14,16,18] Algorithm A uses scaling phases to successively reduce ϵ; similar to [13] an analysis without the scaling phases is possible, but yields a worse runtime bound. The main goal of each scaling phase s is to obtain $x^{(s)} \in B$ such that $\lambda(x) \leq c/(1-\epsilon_s)$ holds. Using this approach we gradually reduce ϵ_s until $\epsilon_s \leq \epsilon/2$. Then we have $f_m(x^s) \leq c/(1-\epsilon/2) < c(1+\epsilon)$ and $g_m(x^{(s)}) \geq (1-\epsilon/2)/c > (1-\epsilon)/c$; thus our instance will be solved by the output of the final scaling phase. Since in each scaling phase some components of g are eliminated, more precisely we aim at $\lambda_A(x) \leq c(1+\epsilon_s) \leq c/(1-\epsilon_s)$ where $A \subseteq [M]$ is the set of indices of the active functions. Later we show that the values for the eliminated functions are suitably bounded. The algorithm uses the threshold values

$$T_s := \begin{cases} [M^p(1 - t_s/C)]/\lambda_{[M]}(x^{(s-1)}) & : \quad s = 1 \\ (1 - t_s/C)/[\lambda_{[M]}(x^{(s-1)})\epsilon_s^q] & : \quad s \geq 2 \end{cases}$$

where p, q are constants to be defined later.

Algorithm A

1. Set $s := 0$, $\epsilon_0 := 1$, $t_0 := 1/8$. Compute initial solution $x^{(0)}$.
   If $\lambda(x^{(0)}) \leq c(1+\epsilon/2)$, go to Step 3.
2. Repeat Steps 2.1 – 2.3 {scaling phase s} until $\epsilon_s \leq \epsilon/2$ or $\lambda(x^{(s)}) \leq c(1+\epsilon/2)$.
   2.1. Set $s := s+1$, $\epsilon_s := \epsilon_{s-1}/2$, $x := x^{(s-1)}$, and T_s as above.
   2.2. If Stopping Rule 1 is satisfied, go to Step 2.4.
   Set $A := \{m \in [M] | g_m < T_s\}$.
   2.3. Repeat Steps 2.3.1 – 2.3.6 {coordination phase} forever.
   2.3.1. If Stopping Rule 1 or Stopping Rule 3 is satisfied go to Step 2.4.
   2.3.2. Compute θ, p and q via (2), let $t_s := \epsilon_s/8$, $\alpha := 2\bar{p} - 1 - 2t_s$ and call $\hat{x} := ABS(p, q, \alpha, t_s)$.
   2.3.3. If Stopping Rule 2 is satisfied, go to Step 2.4.
   2.3.4. Compute suitable $\tau \in (0,1)$ and set $x' := (1-\tau)x + \tau\hat{x} \in B$.
   2.3.5. If $\max\{(1-\tau)g_m + \tau\hat{g}_m | m \in A\} > T_s$ then reduce τ to τ' and set $x' := (1-\tau')x + \tau'\hat{x}$.
   2.3.6. Set $A := A \setminus \{m \in [M] | g_m(x') \geq T_s\}$ and $x := x'$.
   2.4. Set $x^{(s)} := x$. {end of scaling phase s}
3. Return the final iterate $x^{(s)} \in B$.

We use three stopping rules for termination of each sacling phase. For Stopping Rule 1 we simply test whether $\lambda_A(x) \leq c(1+\epsilon_s) \leq c/(1-\epsilon_s)$ holds. For Stopping Rule 2 we define similar to [13,14,18] a parameter

$$\nu(x,\hat{x}) := \frac{(p^T f - p^T \hat{f})/\theta + \theta(q^T \hat{g} - q^T g)}{(p^T f + p^T \hat{f})/\theta + \theta(q^T \hat{g} + q^T g)}$$

that depends on the current iterate x and the approximate block solution $\hat{x}$. We terminate the current scaling phase as soon as $\nu(x,\hat{x}) \leq t_s$ holds, where $t_s := \epsilon_s/8$ is an auxiliary parameter. In case of termination x meets the phase requirement, as can be seen in Lemma 5.

Lemma 5. *Let $\epsilon \in (0,1)$ and $t = \epsilon/8$. For a given $x \in B$ let p, q as in (2) and $\hat{x} \in B$ computed by $ABS_c(p,q,\alpha,t)$ using $\alpha := 2\bar{p} - 1 - 2t \leq 2\bar{p} - 1 - t - t/C$. If $\nu(x,\hat{x}) < t$, then $\lambda_A(x) \leq c(1+\epsilon) \leq c/(1-\epsilon)$ holds.*

The proof of Lemma 5 is omitted. The motivation behind Stopping Rule 3 is to estimate the quality of x based on the quality of $x^{(s-1)}$, the input of the current scaling phase. For this the quality is known either because of Lemma 4 or since $x^{(s-1)}$ is the output of the previous scaling phase; this nice intuitive idea is also used in [1,14,16,18]. In order to formulate Stopping Rule 3 we define

$$\omega_s := \begin{cases} 2/[9M(1-\epsilon_1)] & : \quad s = 1 \\ (1-\epsilon_{s-1})/(1-\epsilon_s) & : \quad s \geq 2 \end{cases}$$

which depends on the scaling phase s; for Stopping Rule 3 we terminate the current scaling phase as soon as $\lambda_A(x) \leq \omega_s \lambda(x^{(s-1)})$ is satisfied. The desired result is obtained by Lemma 6.

Lemma 6. *Let $x^{(s-1)}$ be the input for and x an iterate in scaling phase s; furthermore suppose that $\lambda_A(x) \leq \omega_s \lambda(x^{(s-1)})$ holds. If $\lambda(x^{(s-1)}) \leq 9cM/2$ for $s = 1$ and $\lambda(x^{(s-1)}) \leq c/(1-\epsilon_{s-1})$ for $s \geq 2$, we have $\lambda_A(x) \leq c/(1-\epsilon_s)$.*

The proof is omitted; it can be obtained by elementary calculation or following the proof of Lemma 3.4 in [14]. Next we describe how to choose the step length τ and eventually reduce it in Steps 2.3.4 and 2.3.5. In the following we use A to denote the set of indices *before* execution of Step 2.3.6 and A' to denote the set of indices *after* execution of Step 2.3.6. First we present some observations which are independent from the step length that is chosen; for any step length $\gamma \in (0,1)$ we use the convexity of f and the definition of p to obtain

$$\theta - f'_m \;\geq\; \theta - (1-\gamma) f_m - \gamma \hat{f}_m \;=\; (\theta - f_m)[1 + \frac{2\gamma CM}{t\theta} p_m (f_m - \hat{f}_m)]$$

for each $m \in [M]$. In a similar way, since g is concave, we obtain

$$g'_m - 1/\theta \geq (1-\gamma) g_m + \gamma \hat{g}_m - 1/\theta = (g_m - 1/\theta)[1 + \frac{2\gamma CM\theta}{t} q_m (\hat{g}_m - g_m)]$$

for each $m \in A$. We aim at bounding the absolute values of the last summands in the terms in square brackets by $1/2$; to this end, any step length $\gamma \in (0,1)$ will be called *feasible* if and only if

$$\max\{ \max_{m\in[M]} |\frac{2\gamma CM}{t\theta} p_m(f_m - \hat{f}_m)|, \max_{m\in A} |\frac{2\gamma CM\theta}{t} q_m(\hat{g}_m - g_m)| \} \leq \frac{1}{2} \qquad (3)$$

holds. By (3) for a feasible step length γ, any step length $\gamma' \in (0,\gamma)$ is feasible as well. If $\gamma \in (0,1)$ is a feasible step length, using $\theta - f_m > 0$ we obtain $\theta - f'_m > 0$ for each $m \in [M]$. In a similar way we use $g_m - 1/\theta > 0$ to obtain $g'_m - 1/\theta > 0$ for each $m \in A$. Hence $\phi_t(x', A') \leq \Phi_t(\theta, x', A')$ holds in this case, which is important for the further analysis. We use

$$\tau := \frac{t^2}{4CM[(p^T f + p^T \hat{f})/\theta + (q^T \hat{g} + q^T g)\theta]} \qquad (4)$$

to obtain the step length τ mentioned in Step 2.3.4.

Lemma 7. *The step length τ defined by (4) is feasible.*

The proof is omitted. We focus on the case that the condition in Step 2.3.5 is true. In this case we have $\hat{g}_m > T$ for at least one $m \in A$. Then we compute the uniquely determined $\tau' \in (0,1)$ such that $\max\{(1-\tau')g_m + \tau'\hat{g}_m\} = T$ holds, which can be done in $O(M)$ time. By construction $\tau' < \tau$ holds, hence τ' is feasible. The proof of Theorem 1 is omitted for space reasons.

Theorem 1. *In each iteration of the inner loop, we have*

$$\phi_t(x, A) - \phi_t(x', A') > \alpha t^3/(CM)$$

where $\alpha = 1/4$ if the condition in Step 2.3.5 is false and $\phi_t(x, A) - \phi_t(x', A') \geq 0$ otherwise. In the latter case, Step 2.3.6 eliminates at least one index from A.

4 Analysis of Runtime and Eliminated Functions

First we show that within a scaling phase that does not terminate by Stopping Rule 3, the difference of the reduced potentials of the initial solution and the iterate can not be arbitrarily large but is suitably bounded. In the statement of Lemma 8, the constants are $p = 1031$ and $q = 219$; the proof is omitted due to space limitations.

Lemma 8. *Let $x \in B$ be the initial iterate of scaling phase s and let $x' \in B$ arbitrary such that the pair x, x' does not satisfy Stopping Rule 3. Denote by A, A' and θ, θ' the corresponding sets of active indices and the minimizers of the potential function, respectively. Then*

$$D_s := \phi_t(x, A) - \phi_t(x', A') \leq \begin{cases} (6 + p/(8C)\epsilon_s) \ln M + 5/8 & : \quad s = 1 \\ (2 + q)/(8C)\epsilon_s \ln \epsilon_s^{-1} + 6\epsilon_s & : \quad s \geq 2 \end{cases}$$

holds, where p and q are constants.

Lemma 8 states that a scaling phase that is not terminated by Stopping Rule 3 or Stopping Rule 1 must be terminated by Stopping Rule 2 since ϕ_t decreases by at least $\alpha t^3/(CM)$ in each iteration or at least does not increase; in the latter case Step 2.3.6 eliminates indices, which is possible at most M times. The proof is omitted due to space limitations.

Lemma 9. *Let N_s denote the number of iterations in scaling phase s. Then*

$$N_s \leq \begin{cases} [8MC/(\alpha t_s^3) + pM/(\alpha t_s^2)] \ln M & : \quad s = 1 \\ [51MC/(\alpha t_s^2) + qM/(\alpha t_s^2)] \ln \epsilon_s^{-1} & : \quad s \geq 2 \end{cases}$$

holds.

The proof of Lemma 9 is omitted due to space limitations. Before studying the eliminated indices more closely, we present the runtime complexity of the algorithm.

Theorem 2. *The total number of iterations of the algorithm is bounded by $O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}))$.*

Proof. Clearly we have $N_1 \in O(M \ln M)$ and $N_s \in O(M\epsilon_s^{-2} \ln \epsilon_s^{-1})$ for $s \geq 2$ by Lemma 9. Summing over all scaling phases the total number of iterations is

$$O(M(\ln M + \ln \epsilon^{-1} \sum_{s=1}^{\lceil \log \epsilon^{-1} \rceil} 2^{2s})).$$

Since the summation term above is bounded by $O(\epsilon^{-2})$, the claim follows. □

Now we study the eliminated functions; the goal is to show that the values $g_m(x^{(s)})$ for $m \in [M] \setminus A$ at the end of phase s are sufficiently large. The main idea is similar to [14]; since the covering functions g are concave and nonnegative on B, we have $g_m(x') \geq (1-\tau)g_m(x) + \tau g_m(\hat{x}) \geq (1-\tau)g_m(x)$ for any two consecutive iterates x, x' in a scaling phase. Furthermore we have

$$(p^T f + p^T \hat{f})/\theta + (q^T \hat{g} + q^T g)\theta \geq 1 - t/(2C) + t|A|/(2CM) \geq 1/2$$

by Lemma 3, hence in any case we obtain $\tau' \leq \tau \leq t^2/(2CM) =: \tau_s$. This means that in each interpolation step g_m is scaled down by a certain factor which is at least $1 - t^2/(2CM)$, however.

Lemma 10. *Let $x^{(s)}$ be the output of scaling phase s. If $\lambda(x^{(0)}) \leq 9cM/2$ and $\lambda(x^{(s-1)}) \leq c/(1-\epsilon_{s-1})$ for $s \geq 2$, then $g_m(x^{(s)}) \geq (1-\epsilon_s)/c$ for each $m \in [M]$.*

Proof. We consider only the eliminated components $m \in [M] \setminus A$; for the other ones the lemmas above imply the claim. In the worst case an index $m \in [M]$ is eliminated at the beginning of the scaling phase, which means $g_m(x^{(s-1)}) \geq T_s$. In this case we have $g_m(x^{(s)}) \geq (1-\tau_s)^{N_s} T_s$. We aim at proving the stronger inequality $g_m(x^{(s)}) \geq (1-\tau_s)^{N_s} T_s \geq 1/c \geq (1-\epsilon_s)/c$. In the following we use the inequality $(1-z)^\ell \geq (1-\ell z)$ for each $z \in (0,1)$ and $\ell \in \mathbb{N} \cup \{0\}$ which can

be proved by induction on ℓ or found in [20]. First we study phase $s = 1$; with the help of Lemma 9 we obtain

$$(1-\tau_1)^{N_1} \geq (1-\frac{t_1^2}{2CM})^{(\frac{8MC}{\alpha t_1^3}+\frac{pM}{\alpha t_1^2})\ln M} \geq (1-\frac{MC}{t_1^2}\frac{t_1^2}{2CM})^{(\frac{8}{\alpha t_1}+\frac{p}{\alpha C})\ln M}$$
$$= (1/2)^{(\frac{8}{\alpha t_1}+\frac{p}{\alpha C})\ln M} \geq (1/M)^{(\frac{8}{\alpha t_1}+\frac{p}{\alpha C})}$$

where we used $(1/2)^{\ln M} = 1/(2^{\ln M}) \geq 1/(2^{\log M}) = 1/M$ for the last inequality. This means that it is sufficient to show that

$$(1/M)^{(\frac{8}{\alpha t_1}+\frac{p}{\alpha C})} T_1 = (1/M)^{(\frac{8}{\alpha t_1}+\frac{p}{\alpha C})} M^p \frac{1-t_1/C}{\lambda(x^{(0)})} \geq 1/c$$

holds. We use $\lambda(x^{(0)}) \leq 9cM/2$ which holds by Lemma 4 and $1-t_1/C \geq 127/128$; inserting these bounds yields that the inequality above is implied by

$$(1/M)^{(\frac{8}{\alpha t_1}+\frac{p}{\alpha C})} M^p \frac{127}{576cM/2} \geq 1/c$$

which can be rearranged to

$$M^{p-1-\frac{8}{\alpha t_1}-\frac{p}{\alpha C}} \geq 576/127.$$

Since $M \geq 2$, this is satisfied if $p - 1 - 8/(\alpha t_1) - p/(\alpha C) \geq 11/5$ holds. Using $\alpha = 1/4$, $C = 8$ and $t_1 = 1/16$, elementary calculation yields that this can be satisfied by choosing $p = 1031$. Now let $s \geq 2$. Similar to the analysis above we have

$$(1-\tau_s)^{N_s} \geq (1-\frac{t_s^2}{2CM})^{(\frac{51CM}{\alpha t_s^2}+\frac{qM}{\alpha t_s^2})\ln \epsilon_s^{-1}} \geq (1-\frac{CM}{t_s^2}\frac{t_s^2}{2CM})^{(\frac{51}{\alpha}+\frac{q}{\alpha C})\ln \epsilon_s^{-1}}$$
$$= (1/2)^{(\frac{51}{\alpha}+\frac{q}{\alpha C})\ln \epsilon_s^{-1}} \geq \epsilon_s^{(\frac{51}{\alpha}+\frac{q}{\alpha C})}$$

where we used $(1/2)^{\ln \epsilon_s^{-1}} = 1/(2^{\ln \epsilon_s^{-1}}) \geq 1/(2^{\log \epsilon_s^{-1}}) = 1/\epsilon_s^{-1} = \epsilon_s$ for the last estimation. Here it is sufficient to show that

$$\epsilon_s^{(\frac{51}{\alpha}+\frac{q}{\alpha C})} T_s = \epsilon_s^{(\frac{51}{\alpha}+\frac{q}{\alpha C})} \epsilon_s^{-q} \frac{1-t_s/C}{\lambda(x^{(s-1)})} \geq 1/c$$

holds. Parallel to the argumentation before we use $\lambda(x^{(s-1)}) \leq c/(1-\epsilon_{s-1}) \leq 2c$ and $1 - t_s/C \geq 2/3$ to obtain that the inequality above is implied by

$$\epsilon_s^{(\frac{51}{\alpha}+\frac{q}{\alpha C})} \epsilon_s^{-q} \frac{1}{3c} \geq 1/c$$

which can be rearranged to

$$(\epsilon_s^{-1})^{q-\frac{51}{\alpha}-\frac{q}{\alpha C}} \geq 3.$$

We have $\epsilon_s \leq 1/4$, which implies $\epsilon_s^{-1} \geq 4$; hence $q - 51/\alpha - q/(2C) \geq 4/5$ is sufficient. Using $\alpha = 1/4$ and $C = 8$, this is satisfied by $q = 219$. □

5 Conclusion

We assumed above that the price vectors are computed exactly, which is impractical since we cannot solve (11) for θ; however an approximation for which only $O(M \ln(M\epsilon^{-1}))$ arithmetic operations per iteration are necessary is suitable as well, which can be shown with an elementary analysis that is very similar to Subsect. 4.2 in [16]. We contributed an efficient and simple approximation algorithm with data-independent coordination complexity which solves an important class of feasibility problems, namely so-called mixed problems. In each iteration our algorithm needs to approximately solve a feasibility problem over B. Our result also solves a generalization of an open problem from [24]. In contrast to [16], our algorithm does not take into account a part of the history of iterates, which makes it slightly more straightforward to implement; furthermore the block problem is more natural. Within the limitations of the proof of Lemma 10, τ is amenable to line search. We suggest to evaluate whether line search yields a larger difference in the reduced potential similar to [1]. Furthermore it is an open problem whether a reduction of $O(\epsilon^{-2})$ to $O(\epsilon^{-1})$ in the runtime bound is possible.

Acknowledgement. Florian Diedrich thanks Denis Naddef for kind hospitality during the preparation of this article.

References

1. Aizatulin, M., Diedrich, F., Jansen, K.: Implementation of approximation algorithms for the max-min resource sharing problem. In: Àlvarez, C., Serna, M. (eds.) WEA 2006. LNCS, vol. 4007, pp. 207–218. Springer, Heidelberg (2006)
2. Baille, F., Bampis, E., Laforest, C.: Bicriteria scheduling of parallel degradable tasks for network access under pricing constraints. In: INOC 2003. Proceedings of the International Network Optimization Conference, pp. 37–42 (2003)
3. Baille, F., Bampis, E., Laforest, C.: Maximization of the size and the weight of schedules of degradable intervals. In: Chwa, K.-Y., Munro, J.I.J. (eds.) COCOON 2004. LNCS, vol. 3106, pp. 219–228. Springer, Heidelberg (2004)
4. Bienstock, D.: Potential function methods for approximately solving linear programming problems: theory and practice. Kluwer Academic Publishers, Dordrecht (2002)
5. Bienstock, D., Iyengar, G.: solving fractional packing problems in $O^*(1/\epsilon)$ iterations. In: STOC 2004. Proceedings of the 36th ACM Symposium on Theory of Computing, pp. 146–155 (2004)
6. Carr, R.D., Fleischer, L., Leung, V.J., Phillips, C.A.: Strengthening integrality gaps for capacitated network design and covering problems. In: SODA 2000. Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms, pp. 106–115 (2000)
7. Chudak, F.A., Eleutério, V.: Improved approximation schemes for linear programming relaxations of combinatorial optimization problems. In: Jünger, M., Kaibel, V. (eds.) Integer Programming and Combinatorial Optimization. LNCS, vol. 3509, pp. 81–96. Springer, Heidelberg (2005)
8. Fleischer, L.: A fast approximation scheme for fractional covering problems with variable upper bounds. In: SODA 2004. Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms, pp. 1001–1010. ACM Press, New York (2004)

9. Garg, N., Könemann, J.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In: FOCS 1998. Proceedings of the 39th Annual IEEE Computer Society Conference on Foundations of Computer Science, pp. 300–309 (1998)
10. Garg, N., Khandekar, R.: Personal communication (2004)
11. Grigoriadis, M.D., Khachiyan, L.G.: Coordination complexity of parallel price-directive decomposition. Mathematics of Operations Research 2, 321–340 (1996)
12. Grigoriadis, M.D., Khachiyan, L.G.: Fast approximation schemes for convex programs with many blocks and coupling constraints. SIAM Journal on Optimization 4, 86–107 (1994)
13. Grigoriadis, M.D., Khachiyan, L.G., Porkolab, L., Villavicencio, J.: Approximate max-min resource sharing for structured concave optimization. SIAM Journal on Optimization 41, 1081–1091 (2001)
14. Jansen, K.: Approximation algorithm for the general max-min resource sharing problem. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, Springer, Heidelberg (2004), Mathematical Programming Series A 106, 547–566 (2006) see also http://www.springerlink.com/content/fv2p781736gk4856/
15. Jansen, K.: Approximation algorithms for min-max and max-min resource sharing problems and applications. In: Bampis, E., Jansen, K., Kenyon, C. (eds.) Efficient Approximation and Online Algorithms. LNCS, vol. 3484, pp. 156–202. Springer, Heidelberg (2006)
16. Jansen, K.: Approximation algorithm for the mixed fractional packing and covering problem. In: IFIP TCS 2004. Proceedings of the 3rd IFIP Conference on Theoretical Computer Science, pp. 223–236. Kluwer Publisher, Dordrecht (2006) and SIAM Journal on Optimization, 17, 331–352 (2006)
17. Jansen, K., Porkolab, L.: On preemptive resource constrained scheduling: polynomial time approximation schemes. In: Cook, W.J., Schulz, A.S. (eds.) IPCO 2002. LNCS, vol. 2337, pp. 329–349. Springer, Heidelberg (2002) and SIAM Journal on Discrete Mathematics 20, 545–563 (2006)
18. Jansen, K., Zhang, H.: Approximation algorithms for general packing problems with modified logarithmic potential function. In: TCS 2002. Proceedings of the 2nd IFIP International Conference on Theoretical Computer Science. Foundations of information technology in the era of network and mobile computing, pp. 255–266. Kluwer, Dordrecht (2002)
19. Khandekar, R.: Lagrangian relaxation based algorithms for convex programming problems, PhD Thesis, Indian Institute of Technology, New Delhi (2004)
20. Knuth, D.E.: The art of computer programming, Volume I: Fundamental Algorithms. Addison-Wesley, Reading (1968)
21. Lu, Q., Zhang, H.: Implementation of approximation algorithms for the multicast congestion problem. In: Nikoletseas, S.E. (ed.) WEA 2005. LNCS, vol. 3503, pp. 152–164. Springer, Heidelberg (2005)
22. Villavicencio, J., Grigoriadis, M.D.: Approximate Lagrangian decomposition with a modified Karmarkar logarithmic potential. In: Network Optimization. Lecture Notes in Economics and Mathematical Systems, vol. 450, pp. 471–485. Springer, Heidelberg (1997)
23. Young, N.E.: personal communication (2004)
24. Young, N.E.: Sequential and parallel algorithms for mixed packing and covering. In: FOCS 2001. Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, pp. 538–546 (2001)

Extending the Hardness of RNA Secondary Structure Comparison

Guillaume Blin[1], Guillaume Fertin[2], Irena Rusu[2], and Christine Sinoquet[2]

[1] IGM-LabInfo - UMR CNRS 8049 - Université de Marne-la-Vallée - France
gblin@univ-mlv.fr
[2] LINA - FRE CNRS 2729 - Université de Nantes - France
{fertin,rusu,sinoquet}@lina.univ-nantes.fr

Abstract. In molecular biology, RNA structure comparison is of great interest to help solving problems as different as phylogeny reconstruction, prediction of molecule folding and identification of a function common to a set of molecules. Lin *et al.* [6] proposed to define a similarity criterion between RNA structures using a concept of edit distance ; they named the corresponding problem Edit. Recently, Blin *et al.* [3] showed that another problem, the Longest Arc-Preserving Common Subsequence problem (or Lapcs), is in fact a subproblem of Edit. This relationship between those two problems induces the hardness of what was the last open case for the Edit problem, Edit(Nested,Nested), which corresponds to computing the edit distance between two secondary structures without pseudoknots. Nevertheless, Lapcs is a very restricted subproblem of Edit: in particular, it corresponds to a given system of editing costs, whose biological relevance can be discussed ; hence, giving a more precise categorization of the computational complexity of the Edit problem remains of interest. In this paper, we answer this question by showing that Edit(Nested,Nested) is **NP**-complete for a large class of instances, not overlapping with the ones used in the proof for Lapcs, and which represent more biologically relevant cost systems.

Keywords: computational biology, RNA structures, arc-annotated sequences, edit distance, NP-hardness.

1 Introduction

The understanding of biological mechanisms, at a molecular scale, is induced by the discovery and the study of various RNA functions. It is established that the conformation of an RNA molecule (a single strand composed of bases A, U, C and G also called primary structure) partially determines the function of the molecule. This conformation results from the molecule folding due to local pairings between complementary bases ($A-U$ and $C-G$, connected by a hydrogen bond). Thus, such a molecule has both double-stranded areas (stems) and various types of loops or areas with unpaired bases. A model underlying a given RNA conformation is the secondary structure, with its stems, bulges, and various loops.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 140–151, 2007.

RNA secondary structure comparison is essential for (*i*) identification of highly conserved structures during evolution (which cannot always be detected in the primary sequence, since it is often unpreserved) which suggest a significant common function for the studied RNA molecules, (*ii*) RNA classification of various species (phylogeny), (*iii*) RNA folding prediction by considering a set of already known secondary structures and (*iv*) identification of a consensus structure and consequently of a common role for molecules.

At a theoretical level, the RNA structure comparison problem can be modeled by the class of problems Edit(T_1,T_2) which consist in computing the minimum number of edit operations needed to transform a structure of type T_1 into a structure of type T_2, where T_1, T_2 take values in {plain, nested, crossing, unlimited} (cf. Section 2 for more details). Lin *et al.* [6] proposed to take simultaneously into account primary and secondary structures in RNA comparison by jointly considering a base and its potential hydrogen bond with another base in the similarity computation. They proposed in [6] exact and approximate polynomial algorithms for some classes of problems Edit(T_1,T_2). They also gave some complexity proofs for some other classes. The complexity of the Edit(Nested, Nested) problem was left as an open problem.

Recently, Blin *et al.* [3] showed that the complexity of this last problem is actually closed since it simply follows from the complexity of the Longest Arc-Preserving Common Subsequence problem [4] (Lapcs for short). However, a sharp analysis of the equivalence between the Lapcs and the Edit problems shows in fact that only a very restricted number of instances of Edit(Nested, Nested) are shown to be **NP**-complete. Moreover, the cost system should satisfy restrictions which can be biologically discussed. Therefore, as another step towards establishing the precise complexity landscape of the Edit problem, it is of interest to consider a more accurate class of instances – but not overlapping with the one used in the proof from Lapcs –, for determining more precisely what makes the problem hard. For that purpose, we propose after defining some notations (Section 2) a non-trivial reduction via a 2-page book embedding with some special requirements on the costs for the edit operations (Section 3).

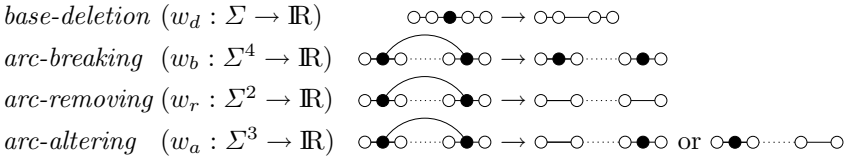
2 Notations and Problem Description

An RNA structure can be described by the sequence of its bases together with the set of hydrogen bonds possibly connecting bases A to bases U or bases C to bases G. This structure is commonly represented by an arc-annotated sequence. Given a finite alphabet Σ, an arc-annotated sequence is defined by a pair (S, P), where S is a string on Σ^* and P is a set of arcs connecting pairs of characters of S. In the following, we will refer to the characters as *bases* in reference to RNA structures. Any base with no arc incident to it is called *free*. As usually considered in arc-annotated sequences comparison, we distinguish four levels of arc structure [4]:

- Plain: there is no arc,
- Nested: no base is incident to more than one arc and no arcs are crossing,

– Crossing: no base is incident to more than one arc,
– Unlimited: no restriction at all.

Those four levels respect an obvious inclusion relation denoted by the $\subset$ operator: Plain $\subset$ Nested $\subset$ Crossing $\subset$ Unlimited. In order to compare two arc-annotated sequences, we consider the set of edit operations (and their associated costs) introduced in [6]. There are four substitution operations which induce renaming of bases in the arc-annotated sequence. They are listed together with their associated cost: *base-match* ($w_m : \Sigma^2 \to \mathbb{R}$), *base-mismatch* ($w_m : \Sigma^2 \to \mathbb{R}$), *arc-match* ($w_{am} : \Sigma^4 \to \mathbb{R}$), *arc-mismatch* ($w_{am} : \Sigma^4 \to \mathbb{R}$). Moreover, there are four deletion operations which induce deletion of bases and/or of arcs, which we list together with their associated cost:



In the following, given two arc-annotated sequences (S, P) and (T, Q), an *edit script* from (S, P) to (T, Q) will refer to a series of non-oriented edit operations transforming (S, P) into (T, Q). The *cost of an edit script* from (S, P) to (T, Q) is the sum of the costs of each operation involved in the edit script. We define the *edit distance* between (S, P) and (T, Q) as the minimum cost of an edit script from (S, P) to (T, Q). Finding this edit distance is called the Edit problem. To any edit script from (S, P) to (T, Q) corresponds an *alignment* of the characters of S and T such that (i) if a base is inserted or deleted in a sequence, it is aligned with a *gap* (indicated by $-$) and (ii) if a base of one sequence is (mis)matched with a base of the other sequence, there are aligned together. In the following, we will call *cost* of an alignment A, denoted by $cost(A)$, the cost of the edit script from which the alignment A is obtained. An *optimal alignment* A is an alignment of minimum cost, that is an alignment whose cost is equal to the edit distance.

Lin *et al.* proved in [6] that the problem Edit(Crossing, Plain) is *MAX-SNP hard*. Thus, any harder problem (in terms of restriction levels) is also *MAX-SNP hard*. Moreover, they gave a polynomial dynamic programming algorithm for the problem Edit(Nested, Plain), while Sankoff [7] had previously solved the problem Edit(Plain, Plain). The complexity of the Edit(Nested, Nested) problem was left as an open problem (see Table 1).

Recently, Blin *et al.* [3] showed that the complexity of this last problem was in fact closed, since it directly follows from the complexity of a different problem, called Longest Arc-Preserving Common Subsequence (Lapcs for short). As introduced by Evans in [4], the Lapcs problem is defined as follows: given two arc-annotated sequences (S, P) and (T, Q), find the longest – in terms of sequence length – common arc-annotated subsequence (R, U) of (S, P) and (T, Q) such that there exists an arc in U iff it corresponds to an existing arc both in P and in Q. In [3], Blin *et al.* proved that the Lapcs problem is a specific case of the Edit

Table 1. EDIT problem complexity (n and m are the number of bases of each sequence). • m is the size of the PLAIN sequence. ⋆ when $w_d = w_a = 2w_r$, the hardness follows from [5]; when $w_a > w_d$ (with some additional restrictions), our contribution.

	UNLIMITED	CROSSING	NESTED	PLAIN
UNLIMITED	Max-SNP hard			
CROSSING		Max-SNP hard		
NESTED			NP-Complete ⋆	$O(nm^3)$ •
PLAIN				$O(nm)$

problem provided that the cost system for edit operations is correctly chosen. The cost system is the following: $w_r = 2w_d = 2w_a$, and all substitutions operations as well as arc-breakings are prohibited (that is, they are given arbitrary high costs). The main idea is to penalize the deletion operations proportionally to the number of bases that are deleted.

Since the LAPCS problem is **NP**-complete for arc-annotated sequences of NESTED types, so does the last open case for the EDIT problem, EDIT(NESTED, NESTED). Nevertheless, LAPCS is a very specific subproblem of EDIT: it corresponds to instances of EDIT for which $w_r = 2w_d = 2w_a$. In particular, this means that the cost for deleting an unpaired base or a base linked to an hydrogen bound is the same. However, this model would be more realistic if we had $w_a > w_d$, as breaking a hydrogen bond requires more energy. More generally, considering a larger class of instances (not overlapping with the one used in the proof from LAPCS), would help us determine more precisely what makes the problem hard. Hence, we suggest a more general categorization of EDIT problem complexity by defining a non-trivial reduction which provides a larger and non-overlapping class of instances leading to the hardness.

3 Hardness of RNA Secondary Structure Comparison

As mentioned before, the main contribution of this paper is the proof of the hardness of the RNA secondary structure comparison for a large class of instances not considered previously. The proof relying on the **NP**-completeness of LAPCS requires that $w_r = 2w_a = 2w_d$. In this article, we investigate a more accurate, and non-intersecting class of instances. More precisely, we will prove that the problem is also **NP**-complete when the cost system respects the following requirements:

$$w_a > w_b > w_d > 0 \tag{1}$$

$$w_r > w_a + w_d \tag{2}$$

$$w_b + \frac{w_d}{3} > w_a \tag{3}$$

$$w_m > 2w_r \tag{4}$$

Our hardness result thus holds no matter how the costs are chosen so as to satisfy the above constraints. The decision problem is defined formally as follows.

INPUT: *Two arc-annotated sequences* (S, P) *and* (T, Q) *of* NESTED *type, a set of costs for the edit operations that satisfy inequalities (1) to (4), and an integer* ℓ.
QUESTION: *Is there an alignment of the two sequences* (S, P) *and* (T, Q) *whose cost is less than or equal to* ℓ *?*

We initially notice that this problem is in **NP** since given an alignment we can check polynomially if its cost is less than or equal to ℓ. In order to prove that it is **NP**-complete, we propose a polynomial reduction from the **NP**-complete problem MIS-3P [2].

MIS-3P
INPUT: *A cubic planar bridgeless connected graph* $G = (V, E)$ *and an integer* k.
QUESTION: *Is there an independent set of cardinality greater than or equal to* k *in* G *?*

A graph $G = (V, E)$ is said to be a *cubic planar bridgeless connected* graph if any vertex of V is of degree three (cubic), G can be drawn in the plane in such a way that no two edges of E cross (planar), and there are at least two edge-disjoint paths connecting any pair of vertices of V (bridgeless connected). The idea of the proof is to transform any cubic planar bridgeless connected graph into two arc-annotated sequences. The construction first uses the notion of 2-page book embedding: a *2-page book embedding* of a graph G is a linear ordering of the vertices of G along a line and an assignment of the edges of G to the two half-planes delimited by the line – called the *pages* – such that no two edges assigned to the same page cross. For convenience, we will refer to the page above (resp. below) the line as the *top-page* (resp. *bottom-page*). In the following, a *2-page s-embedding* will denote a 2-page book embedding with the additional property that in each page, every vertex has degree at least one.

Theorem 1 (Bernhart *et al.* [1]). *Given any cubic planar bridgeless connected graph* G*, it is possible to find, in polynomial time, a 2-page s-embedding of* G.

Given a 2-page s-embedding of a cubic planar bridgeless connected graph $G = (V, E)$, we construct two arc-annotated sequences of NESTED type (S, P) and (T, Q). The underlying raw sequences S and T are defined as follows:

$$S = S_c\ S_1\ S_c\ S_2\ \ldots\ S_c\ S_n\ S_c$$
$$T = T_c\ T_1\ T_c\ T_2\ \ldots\ T_c\ T_n\ T_c$$

where (i) $n = |V|$, (ii) for each $1 \leq i \leq n$, S_i (resp. T_i) is a segment UAUAGG if the degree of the vertex $v_i \in V$ in the top-page (resp. bottom-page) is equal to two, a segment GGUAUA otherwise, and (iii) S_c and T_c are segments made of a given number q of bases C, where $q > \frac{3nw_r}{w_d}$ (the value of q will be justified in the proof of Lemma 2).

Now that the sequences S and T are defined, we have to copy the arc configuration of the top-page (resp. bottom-page) on S (resp. T). Each edge (v_i, v_j), $i < j$, of the top-page is represented by two arcs in P. More precisely, one arc a_1 links a base U of S_i and a base A of S_j. The second arc a_2 is nested in the

first one : it links the base A directly to the right of the base U of a_1 to the base U directly to the left of the base A of a_1. We proceed in a similar way for the bottom-page by adding, for each edge in that page, two arcs in Q. Moreover, we impose that when a vertex v_i is of degree one on the top-page (resp. bottom-page), the two corresponding arcs in P (resp. Q) are incident to the rightmost bases A and U of the segment S_i (resp. T_i). It is easy to check that it is always possible to reproduce on (S, P) and (T, Q) the non-crossing edge configuration of each page. An example of such a construction is given in Figure 1. The size of the sequences is clearly polynomial in n: the length of S and T is $6n + (n + 1)q$ and the total number of arcs is $3n$. In the following, we will refer to any such construction as a *UA-construction*.

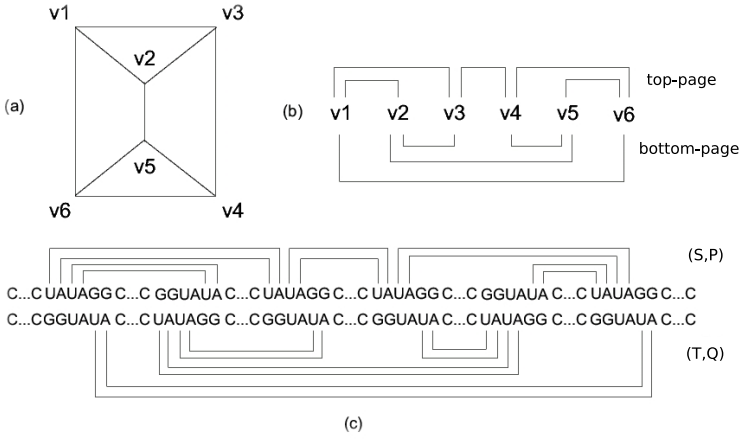


Fig. 1. Example of a UA-construction. Graph (a) is a cubic planar bridgeless connected graph having 6 vertices. Graph (b) is a 2-page s-embedding of graph (a). (c) The two arc-annotated sequences of NESTED type obtained from graph (a) by a UA-construction.

In order to complete the instance of the EDIT(NESTED,NESTED) problem, we define the parameter $\ell = 3n(w_b + \frac{4w_d}{3}) - p(6w_b + 2w_d - 6w_a)$ (p will be formally defined later on).

We start the proof that the reduction from MIS-3P to EDIT(NESTED, NESTED) is correct by giving some properties (Lemmas 1 to 5) about optimal alignments of the sequences (S, P) and (T, Q). Then, these results will be used in Lemma 6 to conclude. We consider in all these lemmas that the conditions imposed by the inequalities (1) to (4) are satisfied.

Lemma 1. *In any optimal alignment of (S, P) and (T, Q), there is no base substitution.*

Proof. Note that base substitution is an operation on bases which occurs either independently (no arc operation is involved) or following an arc-breaking/arc-altering. As the cost of non-pairing base alignment is included in the cost of an

arc-mismatch, a base involved in a base substitution cannot be incident to an arc inducing an arc-mismatch. The principle of this proof is to show that, under the conditions imposed by inequalities (1) to (4), starting from an alignment A containing a base substitution, we can build an alignment A' which does not contain this substitution, satisfying $cost(A') < cost(A)$. Base substitution can occur in three different configurations :

- substitution between two bases non incident to an arc : then A' is obtained from A by changing the base substitution into a base insertion and a base deletion. Thus, we have $cost(A') - cost(A) = 2w_d - w_m$.
- substitution between a base non incident to an arc and a base incident to an arc a. There are two subcases : a induces an (i) arc-breaking or (ii) an arc-altering in A. Let A' be an alignment obtained from A by aligning each base concerned by the substitution with a gap. Then any arc-breaking (resp. arc-altering) is transformed into an arc-altering (resp. arc-removing). Therefore, in case (i) we have $cost(A') - cost(A) = w_a + w_d - (w_b + w_m)$, while in case (ii) we have $cost(A') - cost(A) = w_r + w_d - (w_a + w_m)$.
- substitution between a base incident to an arc a_1 and a base incident to an arc a_2. There are three subcases : a_1 and a_2 induce (i) two arc-breakings, (ii) two arc-alterings, (iii) an arc-altering and an arc-breaking in A. Let A' be an alignment obtained from A by aligning each base concerned by the substitution with a gap. In case (i), we have $cost(A') - cost(A) = 2w_a - (2w_b + w_m)$. In case (ii) we have $cost(A') - cost(A) = 2w_r - (2w_a + w_m)$. Finally, in case (iii) we have $cost(A') - cost(A) = w_r + w_a - (w_b + w_m + w_a) = w_r - (w_b + w_m)$.

Since $w_m > 2w_r$ and $w_r > w_a > w_b > w_d$ (see inequalities (1), (2) and (4)), we deduce that $cost(A') - cost(A) < 0$ in every case. Thus, for any given alignment A with at least one substitution, it is possible to find an alignment A' without this substitution such that $cost(A') < cost(A)$. This proves the lemma. □

Definition 1. *A canonical alignment of two sequences (S, P) and (T, Q) obtained from a UA-construction is an alignment where, for each $1 \leq i \leq n + 1$, the i^{th} segment S_c in (S, P) is aligned base to base to the i^{th} segment T_c in (T, Q).*

Note that, by construction, no arc-match or arc-mismatch can be present in a canonical alignment of (S, P) and (T, Q).

Lemma 2. *Any optimal alignment of (S, P) and (T, Q) is canonical.*

Proof. Let A be a non canonical alignment. We will show that this is not an optimal alignment. By Lemma 1, we assume that A does not contain any substitution. In that case, non canonicity can arise for two reasons:

`Case 1.` There exists a *crossing alignment Up-Down* or *Down-Up* in the alignment A. We denote by crossing alignment Up-Down (resp. Down-Up), an alignment where at least one base of S_k (resp. T_k) is aligned with a base of T_m (resp. S_m) or with a gap situated between two bases of T_m (resp. S_m) and such that $k < m$.

Let A' be a canonical alignment without substitution. According to the conditions imposed by inequalities (1) to (4), the cost associated with any operation on a base not incident to an arc can be upper bounded by $\frac{w_r}{2}$ (since $w_r > w_a + w_d > 2w_d$) and the cost associated with an operation on any base incident to an arc can be upper bounded by $\frac{w_r}{2}$ as well (by equitably distributing the cost of the arc on its two incident bases : the cost w_b of an arc can be seen as composed of the cost $\frac{w_b}{2}$ on each of its incident bases ; since $w_b < w_r$, then $\frac{w_b}{2} < \frac{w_r}{2}$). Since any vertex of G is represented by two segments (one in (S, P) and one in (T, Q)) containing six bases each, the cost of the alignment of S_i and T_i for any vertex v_i is strictly less than $6w_r$, thus $cost(A') < 6nw_r$.

Let A be a crossing non canonical alignment, and let us suppose first that this crossing is Up-Down. In such an alignment the crossing imposes that at least q ($q = |S_c|$) bases C of (T, Q) on the left of T_m must be inserted and that at least as many bases C of (S, P) on the right of S_k must be deleted. Thus we have $cost(A) \geq 2qw_d$. Therefore $cost(A') < 6nw_r < 2qw_d \leq cost(A)$ since $q > \frac{3nw_r}{w_d}$. The alignment A is thus non optimal in case 1. In the case where the crossing is Down-Up, the proof is similar and the same result follows.

`Case 2.` There is no crossing alignment in A. In this case, for any k, any base of S_k is aligned either with a base of T_k or with a gap situated between two bases of T_k. Let us denote by ξ the sum of the alignment costs of the segments S_k and T_k for $k = 1, \ldots, n$ representing the n vertices of G. Thus, we have $cost(A) = \xi + R$ where R is the total cost of base to base alignments of segments S_c and T_c. The initial assumption (*i.e.* A is a non canonical alignment) imposes that at least one base C is deleted and one base C is inserted in A. Thus, we have $R \geq 2w_d$. Consequently, $cost(A) \geq \xi + 2w_d$. Now, let A' be a canonical alignment in which, for any k, any base of S_k is aligned exactly as in A, *i.e.* with the same base of T_k or with a gap. We have $cost(A') = \xi < cost(A)$, therefore A is not optimal. □

By Lemma 2, the cost of an optimal alignment depends on the local alignments of the segments S_k and T_k representing the vertices of G. By Lemma 1, a case analysis leads to a set of exactly eighteen types of local alignments, as illustrated in Figure 2. It is easy to see that any other alignment of the segments representing a vertex is equivalent, in terms of cost, to one of the above mentioned eighteen types.

Definition 2. *We call symmetric of a type of alignment t_i, denoted by t_{iSym}, the type of alignment obtained from t_i by inverting the two segments (i.e. such that the segment on (S, P) is now on (T, Q) and vice versa).*

Lemma 3. *An optimal alignment A' of (S, P) and (T, Q) contains only local alignments of types t_g, t_{ua}, t_{gSym} and t_{uaSym}.*

Proof. First, we notice that by definition of the operations on bases and arcs, two symmetric local alignments have the same cost. Thus, to prove Lemma 3, we will only show that any canonical alignment containing a local alignment of

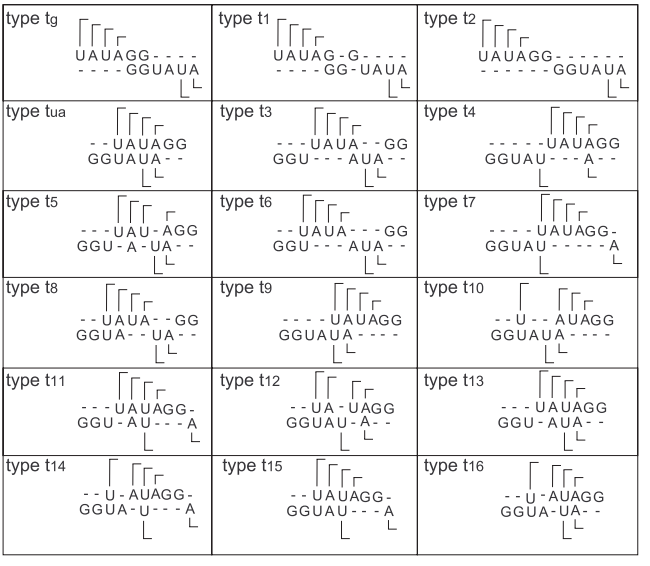


Fig. 2. The eighteen types of local alignments for the segments S_k and T_k

S_i, T_i of type t_1 or t_2 (resp. t_3, t_4, ... or t_{16}) has a cost higher than the cost of the same alignment where this local alignment is of type t_g (resp. t_{ua}). The similar conclusion for symmetric local alignments will then follow.

Let A and A' be two canonical alignments that differ only on the local alignment of S_i and T_i for a given $1 \leq i \leq n$. More precisely, let this local alignment be of type t_{ua} or t_g in A' and of any different type in A. The cost difference between A and A' can only be due to the local alignment of S_i and T_i. Let us notice that this difference is due locally to the alignment of a subset of bases of S_i and T_i. The alignments of bases of S_i and T_i common to A and A' will thus not be taken into account in the computation of the cost difference between A and A'. Moreover, if a change affects a base incident to an arc (say an arc between a base of S_i and a base of S_j), it is necessary to consider not only the base affected (say the base of S_i), but both bases incident to this arc.

The principle of the following proof is to show that from the conditions imposed by inequalities (1) to (4) and for the alignments A, A' defined below, we always have $cost(A') - cost(A) < 0$, meaning that the alignment A is not optimal.

`Case 1.` Let A be a canonical alignment containing a local alignment of S_i and T_i of type t_j for some $3 \leq j \leq 16$. Let A' be the alignment obtained from A by replacing the local alignment of S_i and T_i by a local alignment of type t_{ua}. Let us denote by k_1 (resp. k_2) the number of bases in S_i and T_i in A which induce an arc-removing (resp. arc-altering) and which do not induce this arc-removing (resp. arc-altering) but an arc-altering (resp. arc-breaking) in A'.

Let us denote by k_0 the number of bases deleted or inserted in S_i and T_i in A and which are not deleted or inserted anymore in A'.

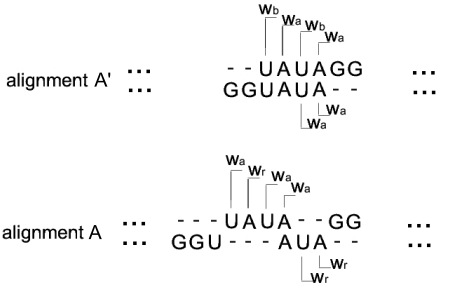


Fig. 3. Example of a canonical alignment A containing a local alignment of type t_3 and its corresponding alignment A' where a local alignment of type t_{ua} replaces the former local alignment. Here, $k_0 = 1, k_1 = 3, k_2 = 2$.

We obtain $k_0 + k_1 + k_2 > 0$, $k_0, k_1, k_2 \geq 0$ and $cost(A') - cost(A) = k_1(w_a - w_r) + k_2(w_b - w_a) + k_0(-w_d)$. According to the conditions imposed by inequalities (1) and (2), $0 < w_d$ and $w_b < w_a < w_r$, we deduce that $cost(A') - cost(A) < 0$.

`Case 2.` Let A be an alignment containing a local alignment of S_i and T_i of type t_j for $j \in \{1, 2\}$. Let A' be the alignment obtained from A by replacing the local alignment of S_i and T_i by a local alignment of type t_g. Let us denote by k_3 the number of bases deleted or inserted in S_i and T_i in A and which are not deleted or inserted any more in A'. We obtain $k_3 > 0$ and $cost(A') - cost(A) = k_3(-w_d)$. According to the conditions imposed by inequality (1), $0 < w_d$ therefore we deduce that $cost(A') - cost(A) < 0$.

Hence, any canonical alignment containing a local alignment of S_i and T_i of type t_1 or t_2 (resp. t_3, t_4, ... or t_{16}) has a cost strictly greater than the cost of the same alignment where this local alignment is of type t_g (resp. t_{ua}). More generally, since symmetric types have the same cost, we conclude that an optimal alignment of (S, P) and (T, Q) is canonical (from Lemma 2) and contains only local alignments of types t_g, t_{ua}, t_{gSym} and t_{uaSym}. □

Lemma 4. *In any optimal alignment, no two segments S_i and S_j (resp. T_i and T_j) having local alignments of type t_g or t_{gSym} can be connected by an arc.*

Proof. Let A be an alignment containing an arc connecting a base of S_i to a base of S_j, and whose local alignments are both of type t_g or t_{gSym}. Let A' be an alignment obtained from A where one of these segments, say S_i, has a local alignment of type t_{ua} or respectively t_{uaSym}. We will show that $cost(A') - cost(A) < 0$. Let k_1 (resp. k_2) denote the number of bases of S_i in A which induce an arc-removing (resp. arc-altering) and which in A' do not induce this arc-removing (resp. arc-altering) but an arc-altering (resp. arc-breaking). Thus, we have $cost(A') - cost(A) = 4w_d - 2w_d + k_1(w_b - w_a) + k_2(w_a - w_r) = 2w_d + 2w_a - 2w_r + k_1(w_b - w_a) + (k_2 - 2)(w_a - w_r)$ where $k_1 + k_2 = 6$, $k_1 \geq 0$ and

$k_2 \geq 2$. According to the conditions imposed by inequalities (1) and (2), $0 < w_d$ and $w_b < w_a < w_r$, therefore we have $cost(A') - cost(A) < 0$. The proof is similar considering T_i and T_j. □

Definition 3. *A canonical alignment A' of (S, P) and (T, Q) containing only local alignments of types t_g, t_{ua}, t_{gSym} and t_{uaSym} and in which no arc connects two segments whose local alignments are of type t_g or t_{gSym} (i.e. respecting the conditions of Lemmas 3 and 4), is called t_g-stable.*

Lemma 5. *The cost of a t_g-stable canonical alignment A' is $cost(A') = 3n(w_b + \frac{4w_d}{3}) - p(6w_b + 2w_d - 6w_a)$ where p is the number of segments whose alignment is of type t_g or t_{gSym}.*

Proof. As mentioned previously, on the whole (S, P) and (T, Q) contain $3n$ arcs. If p is the number of local alignments of type t_g or t_{gSym} in A', then there exists $6p$ arcs connecting a base belonging to a local alignment of type t_g or t_{gSym} to a base belonging to a local alignment of type t_{ua} or t_{uaSym}, and thus $3n - 6p$ arcs between pairs of bases belonging to local alignments of type t_{ua} or t_{uaSym}. We compute the cost of any arc joining two local alignments of types t_{ua} and t_{ua} (resp. t_{ua} and t_g) or symmetric by adding to w_b (resp. w_a) a supplementary cost computed for each incident base and resulting from the equitable distribution of costs w_d between the six arcs involved in each concerned local alignment. These costs w_d deal with the free bases, inside each local alignment.

The cost of an arc between two local alignments of type t_{ua} or t_{uaSym} is computed as follows : $4w_d$ must be distributed on the six arcs involved. Thus for each base incident to such an arc, a supplementary cost of $\frac{4w_d}{6} = \frac{2w_d}{3}$ must be taken into account. The cost of any arc involved in a t_{ua}-t_{ua} junction (or symmetric) is then $w_b + \frac{2w_d}{3} + \frac{2w_d}{3} = w_b + \frac{4w_d}{3}$. For a local alignment of type t_g (or symmetric), we must distribute $2w_d$ on the six arcs involved, which leads to a supplementary cost of $\frac{2w_d}{6} = \frac{w_d}{3}$ for any base incident to such an arc. Thus the cost of any arc involved in a t_{ua}-t_g junction (or symmetric) is $w_a + \frac{w_d}{3} + \frac{2w_d}{3} = w_a + w_d$. We obtain $cost(A') = (3n - 6p)(w_b + \frac{4w_d}{3}) + 6p(w_d + w_a) = 3n(w_b + \frac{4w_d}{3}) - p(6w_b + \frac{24w_d}{3} - 6w_d - 6w_a) = 3n(w_b + \frac{4w_d}{3}) - p(6w_b + 2w_d - 6w_a)$. □

Lemmas 1 to 5 provide us with all the necessary intermediate results to show that the reduction from MIS-3P to EDIT(NESTED, NESTED) is valid.

Lemma 6. *A cubic planar bridgeless connected graph G has an independent set V' such that $|V'| \geq k$ if and only if the edit distance between the sequences (S, P), (T, Q) obtained from G by a UA-construction is at most $\ell = 3n(w_b + \frac{4w_d}{3}) - k(6w_b + 2w_d - 6w_a)$.*

Proof. ($\Rightarrow$) Let $V' \subseteq V$ be an independent set of G such that $|V'| \geq k$. Let A be the canonical alignment of (S, P) and (T, Q) such that (i) $\forall\ v_i \in V'$, the local alignment of S_i and T_i is of type t_g or t_{gSym} and (ii) $\forall\ v_j \in V - V'$, the local alignment of S_j and T_j is of type t_{ua} or t_{uaSym}. Thus, by definition, the alignment is t_g-*stable*. By Lemma 5, $cost(A) = 3n(w_b + \frac{4w_d}{3}) - |V'|(6w_b + 2w_d - 6w_a)$. Since $|V'| \geq k$ by hypothesis, we have $cost(A) \leq 3n(w_b + \frac{4w_d}{3}) - k(6w_b + 2w_d - 6w_a) = \ell$.

($\Leftarrow$) Suppose there exists an edit script between the sequences (S, P), (T, Q), for which the corresponding alignment A' satisfies $cost(A') \leq \ell$. Now let A_{OPT} be an optimal alignment of (S, P) and (T, Q). Let V' be the set of vertices v of G for which, in A_{OPT}, local alignments of the corresponding segments are of type t_g or t_{gSym}. Since we know by Lemma 4 that no arc connects segments of type t_g or t_{gSym} in A_{OPT}, we conclude that V' is an independent set of G. Moreover, by Lemma 5, we have $cost(A_{OPT}) = 3n(w_b + \frac{4w_d}{3}) - |V'|(6w_b + 2w_d - 6w_a)$ and since $cost(A_{OPT}) \leq cost(A') \leq \ell$ with $\ell = 3n(w_b + \frac{4w_d}{3}) - k(6w_b + 2w_d - 6w_a)$, we conclude that $k \leq |V'|$. Lemma 6 is proved. □

4 Conclusion

In this paper, we have proved that the problem EDIT(NESTED,NESTED) defined in [6] is **NP**-complete. This is done using a non trivial reduction from MIS-3P, via a 2-page s-embedding. Though the **NP**-completeness of the problem was already known due to the fact that the LAPCS problem for nested structures was proved to be **NP**-complete [5,3], we have extended this result to a larger and non-intersecting class of instances, for which the set of costs is biologically more relevant. Though the result we give in this paper is in some sense negative, we point out that EDIT(NESTED,NESTED) has a polynomial approximation algorithm of ratio $\beta = \max\{\frac{2w_a}{w_b+w_r}, \frac{w_b+w_r}{2w_a}\}$ [6]. However, this approximation ratio depends on the respective values of the parameters w_a, w_b and w_r. An interesting question is whether there exists a polynomial algorithm for EDIT(NESTED,NESTED) with *constant* approximation ratio.

References

1. Bernhart, F., Kainen, P.C.: The book thickness of a graph. Journal of Combinatorial Theory, Series B 27(3), 320–331 (1979)
2. Biedl, T., Kant, G., Kaufmann, M.: On triangulating planar graphs under the four-connectivity constraint. Algorithmica 19, 427–446 (1997)
3. Blin, G., Touzet, H.: How to compare arc-annotated sequences: the alignment hierarchy. In: Crestani, F., Ferragina, P., Sanderson, M. (eds.) SPIRE 2006. LNCS, vol. 4209, pp. 291–303. Springer, Heidelberg (2006)
4. Evans, P.A.: Algorithms and Complexity for Annotated Sequence Analysis. PhD thesis, University of Victoria (1999)
5. Lin, G.H., Chen, Z.Z., Jiang, T., Wen, J.: The longest common subsequence problem for sequences with nested arc annotations. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 444–455. Springer, Heidelberg (2001)
6. Lin, G.H., Ma, B., Zhang, K.: Edit distance between two RNA structures. In: RECOMB'01. Proceedings of the 5th International Conference on Computational Biology, pp. 211–220. ACM Press, New York (2001)
7. Sankoff, D., Kruskal, B.: Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison. Addison-Wesley, Reading (1983)

On the On-Line Weighted k-Taxi Problem

Weimin Ma[1,2] and Ke Wang[1]

[1] School of Economics and Management
Beijing University of Aeronautics and Astronautics, Beijing, 100083, P.R. China
mawm@buaa.edu.cn, wangke@sem.buaa.edu.cn
[2] School of Economics and Management, Xi'an Technological University
Xi'an, Shaanxi Province, 710032, P.R. China

Abstract. Based on some results of k-server problem and k-taxi problem, the on-line weighted k-taxi problem is originally proposed and studied by our team. In the weighted k-taxi problem, the cost of every taxi on the same distance is different whereas that is same in the traditional k-taxi problem. In this paper, the formulation of the on-line weighted k-taxi problem is presented first. Following that, some preliminary results which used to get the main theorems are given. Thirdly, some on-line algorithms are designed to address the problem and the competitive analysis are given in detail. Furthermore, the lower bound of competitive ratio for some special cases is obtained. Finally, some conclusions are made and some future research directions are pointed out.

1 Introduction

The k-server problem, originally introduced by Manasse, McGeoch and Sleator[1], is one of the three basic on-line problems which had been studied extensively in the late eighties and early nineties of last century, and the two others are paging and metrical task systems.

In the k-server problem, we are given k mobile servers in a metric space M. When a request is made by a point, one of the servers must be moved to the point to satisfy this request immediately. The cost of all servers equals to the distance of all servers' moving. Our goal is to minimize the total cost. An algorithm A which decides a server to satisfy the request at each step, is said to be *on-line* if its decisions are made without the information about future requests. In general, due to the absence of information about the future, an on-line algorithm can't serve the request in an optimal fashion. However, competitive analysis approach, first applied to analyze on-line problem by Sleator and Tarjian[2], suggests us a way of dealing with this problem. The idea behind the competitive analysis is to evaluate the performance of an on-line algorithm by comparing to that of the optimal off-line algorithm.

In the weighted k-server problem, servers have different costs, i.e. each server has been assigned some non-negative weight. The weighted case of the k-server problem turns out to be much more difficult. In paper [3], it was showed that the competitive ratio is at least $k^{\Omega(k)}$ in any space with at least $k+1$ points.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 152–162, 2007.

For $k = 2$, Koutsoupias and Taylor[4] proved that no online algorithm can be better than 10.12-competitive, even if the underlying metric space is the line. For the case $k = 2$ in uniform space, Chrobak and Sgall[5] showed that the Work Function Algorithm is 5-competitive, and no better ratio is possible if the server weights can have arbitrary positive values. They also gave a 5-competitive randomized memoryless algorithm, as well as a matching lower bound. Concerning the weighted 2-server problem on the uniform space, Epstein, Imreh, and Stee[6] gave results for varying cost ratios between the servers. For ratios below 2.2, they presented an optimal algorithm which is trackless. They also presented a general lower bound for trackless algorithms depending on the cost ratio, proving that the algorithm is the optimal trackless algorithm up to a constant factor for any cost ratio.

The k-taxi problem is a generalization of the famous k-server problem. Some important results of the problem have been proposed [7-12]. In the k-taxi problem, each request contains two points, one of which is the (start) point making the request and the other is the destination point. The background of the k-taxi problem is described as follows. There are k taxies on a traffic net to supply service. When a service request that taking a passenger at one point to another point occurs, a taxi must be moved to serve the request immediately, but the information about future requests is unknown. It is assumed that all the taxies are scheduled by a control center.

In this paper, the discussion focuses on the weighted case of the k-taxi problem. It is shown that the weighted k-taxi problem turns out to be more difficult than the unweighted case. In the weighted k-taxi problem, taxies have different costs, i.e. each taxi (say, t_j) has been assigned some non-negative weight (say, μ_j). When the taxi t_j moves, the cost equals to μ_j times of the distance of its moving. The model of this problem is formulated in section 2. Then the rest of paper is organized as follows: in Section 3, some preliminary results concerning the problem are presented. In section 4, several on-line algorithms are designed to address the problem. And then the lower bound of competitive ratio is discussed in section 5. Finally, in section 6, we conclude the paper and discuss the future research directions.

2 The Model

In the weighted k-taxi problem we have k taxies that reside and move in a metric space M to supply service. Each taxi $t_j (1 \leq j \leq k)$ has a non-negative weight μ_j. When the taxi t_j moves, the cost equals to μ_j times of the distance of its moving. Without loss of generality, it is assumed that

$$0 < \mu_1 \leq \mu_2 \leq \mu_3 \leq \cdots \leq \mu_k \leq 1 \tag{1}$$

Thus, t_1 and t_k denote the cheapest and the most expensive taxi, respectively.

For any points x and y in the given metric space, $d(x, y)$ denotes their distance, and the distance is symmetric, i.e., for all x, y, $d(x, y) = d(y, x)$. A service request $r = (a, b)$, (a, b are different points in the given metric space), implies that there

is a passenger at point a must be taken to b by taxi. A service request sequence R consists of some service request in turn, namely $R = (r_1, r_2, \cdots, r_m)$, where $r_i = (a_i, b_i)$. All discussions are based on an essential assumption: when a new service request occurs, k taxies are all free. The weighted k-taxi problem is to decide which taxi should be moved when a new service request occurs on the basis that we have no information about future possible requests. The goal of optimization is to minimize the total cost of all taxies.

For a known sequence $R = (r_1, r_2, \cdots, r_m)$, let $C_{\mathrm{OPT}}(R)$ be the total cost after finishing it with optimal off-line algorithm. An optimal off-line algorithm knows the entire input sequence in advance and can process it optimally. If an algorithm ON can schedule the taxies without information of the sequence after r_i for every new service request r_i, we call ON an on-line algorithm. An on-line algorithm ON is said to be a competitive algorithm, if there are constants α and β for any possible R satisfying:

$$C_{\mathrm{ON}}(R) \leq \alpha \cdot C_{\mathrm{OPT}}(R) + \beta \tag{2}$$

where $C_{\mathrm{ON}}(R)$ is the total cost of satisfying sequence R with algorithm ON. α is the competitive ratio. ON is also called a α-competitive algorithm.

3 Some Preliminary Results

In this section we propose some preliminary results as follows. For the given metric space M, let $d_{max} = \max d(x, y)$, $d_{min} = \min d(x, y)$, where x and y are two different points in M, and let $\lambda = \frac{d_{max}}{d_{min}}$. Then we have

Lemma 1. *For the present request $r = (a, b)$, only considering the cost of finish r, if $\frac{\mu_2}{\mu_1} > \lambda$ holds and there is no taxi at a, moving t_1 to a first and then moving t_1 from a to b to finish r is optimal.*

Proof. Without loss of generality, we assume that taxi t_1 is located at c, and taxi $t_i (i \neq 1)$ is at d, where neither c nor d is a. Let $C_1(r)$ and $C_i(r)$ denote the cost of finishing r by t_1 and t_i respectively. Then we obtain

$$\frac{C_i(r)}{C_1(r)} = \frac{\mu_i \cdot [d(d, a) + d(a, b)]}{\mu_1 \cdot [d(c, a) + d(a, b)]} \geq \frac{\mu_i}{\mu_1 \cdot \lambda} \tag{3}$$

As defined above, for any i $(i \neq 1)$, we have $\mu_i \geq \mu_2$. If $\frac{\mu_2}{\mu_1} > \lambda$ holds, we have

$$\frac{C_i(r)}{C_1(r)} \geq \frac{\mu_i}{\mu_1 \cdot \lambda} \geq \frac{\mu_2}{\mu_1 \cdot \lambda} > 1 \tag{4}$$

That is to say for any i $(i \neq 1)$, the cost of t_i to finish r is more than that of t_1. □

Lemma 2. *For the present request $r = (a, b)$, only considering the cost of finish r, if $\frac{\mu_2}{\mu_1} > 1 + \lambda$ holds, moving t_1 from a to b to finish r (moving t_1 to a first, if t_1 is not at a) is optimal, no matter whether there are taxies at a or not.*

Proof. For the case that t_1 is located at a, lemma 2 obviously stands up, because t_1 is the cheapest one of the k taxies. Next, we discuss the case that t_1 is not located at a. Without loss of generality, it is also assumed that taxi t_1 is located at $c(c \neq a)$, and taxi $t_j(j \neq 1)$ is at a. If $\frac{\mu_2}{\mu_1} > 1 + \lambda$, then $\frac{\mu_2}{\mu_1} > \lambda$ holds. From lemma 1, we can obtain that the cost of moving any other taxi which is not located at a is more than $C_1(r)$. Therefore, we any need to compare $C_1(r)$ with $C_j(r)$.

$$\begin{aligned}\frac{C_j(r)}{C_1(r)} &= \frac{\mu_j \cdot d(a,b)}{\mu_1 \cdot [d(c,a) + d(a,b)]} = \frac{\mu_j}{\mu_1 \cdot [\frac{d(c,a)}{d(a,b)} + 1]} \\ &\geq \frac{\mu_j}{\mu_1 \cdot (1+\lambda)} \geq \frac{\mu_2}{\mu_1 \cdot (1+\lambda)} > 1 \end{aligned} \tag{5}$$

The proof is completed. □

Furthermore, if $\frac{\mu_2}{\mu_1} > 1+\lambda$ holds, from lemma 2 we know that no matter whether a has a taxi or not, moving the taxi t_1 to serve the request is the optimal algorithm. Therefore, we can move t_1 to complete all the request in this case and the cost is minimum. All the other $k-1$ taxies are idle in the whole game. It is equal to the case $k = 1$ in fact, i.e., there is any one taxi in the problem. So we have the following theorem.

Theorem 1. *If $\frac{\mu_2}{\mu_1} > 1 + \lambda$ holds, there is a 1-competitive algorithm for the weighted k-taxi problem.*

As discussed above, it is shown that the weighted case of the k-taxi problem turns out to be more difficult than the unweighted case. In the weighed k-taxi problem, for the present request $r = (a, b)$, the cost of moving a taxi which is not located at a to finish r may be less than that of moving the one at a. But it cannot occur in the unweighted k-taxi problem. However, if the weights are nearly equal, it also could not occur in the weighted case.

Lemma 3. *For the present request $r = (a, b)$, only considering the cost of finish r, if $\mu_1 > \frac{\lambda}{1+\lambda}$ holds and there are taxies located at a, moving the cheapest one at a to finish r is optimal.*

Proof. Without loss of generality, it is assumed that taxi t_i and t_j are located at $c(c \neq a)$ and a, respectively. If $\frac{C_i(r)}{C_j(r)} = \frac{\mu_i \cdot [d(c,a)+d(a,b)]}{\mu_j \cdot d(a,b)} > 1$ for any $i \neq j$ holds, the lemma is proved. The analysis is present as follows.

$$\begin{aligned}\frac{C_i(r)}{C_j(r)} &= \frac{\mu_i \cdot [d(c,a) + d(a,b)]}{\mu_j \cdot d(a,b)} = \frac{\mu_i}{\mu_j} \cdot \left[\frac{d(c,a)}{d(a,b)} + 1\right] \\ &\geq \frac{\mu_i}{\mu_j} \cdot \left(\frac{1}{\lambda} + 1\right) \geq \mu_1 \cdot \frac{1+\lambda}{\lambda} \end{aligned} \tag{6}$$

If $\mu_1 > \frac{\lambda}{1+\lambda}$ holds, then we have $\frac{C_i(r)}{C_j(r)} > 1$. □

4 Competitive Algorithms

In this section, several competitive algorithms and their competitive ratios are presented. Let n denote the number of points in the given metric space M. Subsection 4.1 and 4.2 formulate Greedy Algorithm(GA for short) and Partial Greedy Algorithm(PGA for short), respectively. The two algorithms are compared in subsection 4.3.

4.1 Greedy Algorithm

Firstly, we employ a Greedy Algorithm to address the problem as follows.

Greedy Algorithm. *For the present request $r_i = (a_i, b_i)$,*

(1) If there is only one taxi t_j at a_i, then GA moves t_j from a_i to b_i to complete the request.
(2) If there are more than one taxies at a_i, then GA moves the cheapest one at a_i to b_i to complete the request.
(3) If there is no taxi at a_i, then GA moves the taxi t_1 to a_i first, and then moves t_1 from a_i to b_i to complete the request.

Theorem 2. *For the on-line weighted k-taxi problem,if $\frac{1}{\mu_1} > 1 + \lambda$, GA is a $\frac{1}{\mu_1}$-competitive algorithm; if $\frac{1}{\mu_1} \leq 1 + \lambda$, GA is a $(1 + \lambda)$-competitive algorithm. I.e., the competitive ratio of GA is $\max(\frac{1}{\mu_1}, 1 + \lambda)$.*

Proof. For case (1), the cost of GA for the r_i is $\mu_j \cdot d(a_i, b_i)$, and $\mu_j \cdot d(a_i, b_i) \leq d(a_i, b_i)$. For case (2), GA moves the cheapest one at a_i, so the cost is also not more than $d(a_i, b_i)$. For case (3), the cost is at most $\mu_1 \cdot [d_{max} + d(a_i, b_i)]$. I.e.,

$$C_{\mathrm{GA}}(r_i) \leq \begin{cases} d(a_i, b_i) & \text{for the case (1) and (2);} \\ \mu_1 \cdot [d_{max} + d(a_i, b_i)] & \text{for the case (3).} \end{cases} \tag{7}$$

Then we have

$$\begin{aligned} C_{\mathrm{GA}}(R) &= \sum_{i=1}^{m} C_{\mathrm{GA}}(r_i) \leq \sum_{i=1}^{m} \max(d(a_i, b_i), \mu_1 \cdot [d_{max} + d(a_i, b_i)]) \\ &= \sum_{i=1}^{m} d(a_i, b_i) \cdot \max\left(1, \mu_1 \cdot \left[\frac{d_{max}}{d(a_i, b_i)} + 1\right]\right) \\ &\leq \max(1, \mu_1 \cdot (1 + \lambda)) \cdot \sum_{i=1}^{m} d(a_i, b_i) \end{aligned} \tag{8}$$

It is obvious to see that the cost of finishing R is at least $\mu_1 \cdot \sum_{i=1}^{m} d(a_i, b_i)$ for any algorithm in the given metric space M. That is because $\sum_{i=1}^{m} d(a_i, b_i)$ is the total distance all taxies have to move for finishing R and the weight of the cheapest taxi t_1 is μ_1. Then we have

$$C_{\mathrm{OPT}}(R) \geq \mu_1 \cdot \sum_{i=1}^{m} d(a_i, b_i) \tag{9}$$

Form (8) and (9), we can get

$$C_{\mathrm{GA}}(R) \leq \begin{cases} \frac{1}{\mu_1} \cdot C_{\mathrm{OPT}}(R) & \text{if } \frac{1}{\mu_1} > 1+\lambda; \\ (1+\lambda) \cdot C_{\mathrm{OPT}}(R) & \text{if } \frac{1}{\mu_1} \leq 1+\lambda. \end{cases} \tag{10}$$

The proof is completed. □

4.2 Partial Greedy Algorithm

In this subsection, for the case $1 < k < n-1$, we employ a Partial Greedy Algorithm[13] (which is so called because the Greedy Algorithm is used for some of the cases in this problem) to address the problem. In the paper [13], the PGA moves the nearest truck to serve the request when the point which made the request has no truck. Similar approach also can be seen in paper [8]. In the weighted k-taxi problem, PGA moves the cheapest taxi to serve the request when the point which made the request has no taxi. More details are formulated as follows.

It is assumed that there is at most one taxi located at a point before the first request arrives. Otherwise, we can precondition the taxi locations such that each point has at most one taxi. Furthermore, the cost of this precondition is at most a constant $(k-1)\cdot d_{max}$, and it has no influence on the competitive radio.

Partial Greedy Algorithm. *For the present request $r_i = (a_i, b_i)$,*

(1) If there is a taxi t_j at a_i and no taxi at b_i, then PGA moves t_j from a_i to b_i to complete the request. At present no point has more than one taxi.
(2) If there is no taxi at a_i and there is a taxi t_j at b_i, then PGA moves t_j to a_i first, and then moves t_j from a_i to b_i to complete the request. At present no point has more than one taxi.
(3) If there is a taxi t_x at a_i and also there is a taxi t_y at b_i, and if $x < y (\mu_x \leq \mu_y)$, PGA moves t_x from a_i to b_i to complete the request first and then turn back to a_i, else PGA moves t_y to a_i first and then moves t_y from a_i to b_i to complete the request. At present no point has more than one taxi.
(4) If there is no taxi at a_i and b_i, then PGA moves the taxi t_1 to a_i first and then moves t_1 from a_i to b_i to complete the request. And again no point has more than one taxi.

Theorem 3. *For the case $1 < k < n-1$ of the weighted k-taxi problem, if $\frac{2}{\mu_1} > 1+\lambda$, PGA is a $\frac{2}{\mu_1}$-competitive algorithm; if $\frac{2}{\mu_1} \leq 1+\lambda$, PGA is a $(1+\lambda)$-competitive algorithm. I.e., the competitive ratio of PGA is* $\max(\frac{2}{\mu_1}, 1+\lambda)$.

Proof. The discussion is similar to the description of GA. For case (1), the cost of PGA for the r_i is $\mu_j \cdot d(a_i, b_i)$, and $\mu_j \cdot d(a_i, b_i) \leq d(a_i, b_i)$. For case (2), the cost is $2\mu_j \cdot d(a_i, b_i)$, and $2\mu_j \cdot d(a_i, b_i) \leq 2d(a_i, b_i)$. For case (3), the cost

is $2\min(\mu_x, \mu_y) \cdot d(a_i, b_i)$, and it is also not more than $2d(a_i, b_i)$. For case (4), the cost is at most $\mu_1 \cdot [d_{max} + d(a_i, b_i)]$. I.e,

$$C_{\mathrm{PGA}}(r_i) \leq \begin{cases} d(a_i, b_i) & \text{for the case (1);} \\ 2d(a_i, b_i) & \text{for the case (2) and (3);} \\ \mu_1 \cdot [d_{max} + d(a_i, b_i)] & \text{for the case (4).} \end{cases} \tag{11}$$

Then we have

$$\begin{aligned} C_{\mathrm{PGA}}(R) &= \sum_{i=1}^{m} C_{\mathrm{PGA}}(r_i) + \beta \leq \sum_{i=1}^{m} \max(2d(a_i, b_i), \mu_1 \cdot [d_{max} + d(a_i, b_i)]) + \beta \\ &\leq \max(2, \mu_1 \cdot (1 + \lambda)) \cdot \sum_{i=1}^{m} d(a_i, b_i) + \beta \end{aligned} \tag{12}$$

where β is the cost of preconditioning the taxies such that each point has at most one taxi. From (12) and (9),we have

$$C_{\mathrm{PGA}}(R) \leq \begin{cases} \frac{2}{\mu_1} \cdot C_{\mathrm{OPT}}(R) + \beta & \text{if } \frac{2}{\mu_1} > 1 + \lambda; \\ (1 + \lambda) \cdot C_{\mathrm{OPT}}(R) + \beta & \text{if } \frac{2}{\mu_1} \leq 1 + \lambda. \end{cases} \tag{13}$$

The proof is completed. □

For the case $k \geq n - 1$, we can employ Partial Greedy Algorithm to address the problem as follows. Only move the cheapest $n - 1$ taxis (namely, t_j, where $1 \leq j \leq n - 1$) to complete the service requests, and ignore the others. For every request, use the PGA as defined above to schedule the taxies. It is assumed that the cheapest $n - 1$ taxies are respectively located at $n - 1$ different points in the metric space M before the first request arrives, and the rest of taxies are located arbitrary. Otherwise, we can precondition the taxi locations and the cost of this precondition is at most a constant $(n - 2) \cdot d_{max}$.

As assumed above, the cheapest $n - 1$ taxies are respectively located at $n - 1$ different points. Therefore, the case (4) in the Partial Greedy Algorithm will never occur. For the any other three cases, the cost of finishing r_i could not be more than $2\mu_{n-1} \cdot d(a_i, b_i)$. In fact, the rest taxies (which are not cheaper than t_{n-1}, if exist) are not moved in the whole game. It is equal to the case $k = n - 1$ actually. Then with (9), we can get the following theorem.

Theorem 4. *For the case $k \geq n - 1$ of the weighted k-taxi problem, PGA is a $\frac{2\mu_{n-1}}{\mu_1}$-competitive algorithm.*

4.3 Comparison of the Two Algorithms

In subsections 4.1 and 4.2, we gave two algorithms GA and PGA respectively. Then the two algorithms are compared as follows.

Theorem 5. *For the case $1 < k < n - 1$ of the weighted k-taxi problem, if $\frac{2}{\mu_1} > 1 + \lambda$, then the competitive ratio of GA is less than that of PGA; if $\frac{2}{\mu_1} \leq 1 + \lambda$, then GA and PGA have the same competitive ratio.*

Proof. The criterion, with which one can judge which on-line algorithm is better than the other, is the competitive ratio concerning relevant on-line algorithm. For the case $1 < k < n-1$, respectively the competitive ratios of algorithms GA and PGA are

$$C_{\mathrm{GA}} = \begin{cases} \frac{1}{\mu_1} & \text{if } \frac{1}{\mu_1} > 1+\lambda; \\ 1+\lambda & \text{if } \frac{1}{\mu_1} \le 1+\lambda; \end{cases} \tag{14}$$

and

$$C_{\mathrm{PGA}} = \begin{cases} \frac{2}{\mu_1} & \text{if } \frac{2}{\mu_1} > 1+\lambda; \\ 1+\lambda & \text{if } \frac{2}{\mu_1} \le 1+\lambda. \end{cases} \tag{15}$$

So we have

if $1+\lambda < \frac{1}{\mu_1}$, then $C_{\mathrm{GA}} = \frac{1}{\mu_1}$, $C_{\mathrm{PGA}} = \frac{2}{\mu_1}$, $C_{\mathrm{GA}} < C_{\mathrm{PGA}}$;
if $\frac{1}{\mu_1} \le 1+\lambda < \frac{2}{\mu_1}$, then $C_{\mathrm{GA}} = 1+\lambda$, $C_{\mathrm{PGA}} = \frac{2}{\mu_1}$, $C_{\mathrm{GA}} < C_{\mathrm{PGA}}$;
if $1+\lambda \ge \frac{2}{\mu_1}$, then $C_{\mathrm{GA}} = 1+\lambda$, $C_{\mathrm{PGA}} = 1+\lambda$, $C_{\mathrm{GA}} = C_{\mathrm{PGA}}$.

It is easy to see that if $\frac{2}{\mu_1} > 1+\lambda$ holds, we get $C_{\mathrm{GA}} < C_{\mathrm{PGA}}$. Therefore, as far as competitive ratio is concerned, algorithm GA is better than PGA. If $\frac{2}{\mu_1} \le 1+\lambda$ holds, the two algorithms have the same competitive ratio. □

Concerning the case $k \ge n-1$, the competitive ratio of GA may be less than that of PGA, and also it may be more than that of PGA. Take a special case for example as follows. If $\mu_1 = \mu_2 = \cdots = \mu_{n-1}$, then the competitive ratio of PGA is $C_{\mathrm{PGA}} = \frac{2\mu_{n-1}}{\mu_1} = 2$. And that of GA is $C_{\mathrm{GA}} = \max(\frac{1}{\mu_1}, 1+\lambda) \ge 2 = C_{\mathrm{PGA}}$. Thus, we can see that when $\frac{\mu_{n-1}}{\mu_1}$ is varying below a certain bound, C_{PGA} are also less than C_{GA}; and for some other cases, C_{PGA} may be more than C_{GA}.

However, at the aspect of quality of service, PGA is much better than GA usually. PGA locates the k taxies at k distinct points in the whole game. When request occurs on one of the k points, the request can be satisfied immediately. In fact, PGA is also a coverage strategy. It is to say that the k taxies always cover k points. The idea behind GA is that for the present request, only concern about minimizing the cost of finishing the present request. It is a nature method to handle the problem. In the process of the game, many taxies may be all located at one point. Therefore, the number of points at which the request can be satisfied immediately may lees than k.

5 A Lower Bound

In this section, we give a lower bound of competitive ratio for the weighted k-taxi problem in a symmetric metric space. In the paper [13], the authors compared an on-line algorithm with k trucks to an off-line one with $h \le k$ trucks and got the lower bound as $\frac{k\cdot(\theta+1)}{k\cdot(\theta+2)-2h+2}$, where θ is the ratio between the cost of a truck with goods and that of one without goods on the same distance. Let $\theta = 1$ and $h = k$, then they got the lower bound of the k-taxi problem as $\frac{2k}{k+2}$.

For the weighted case of k-taxi problem, we take in the similar approach and only compare an on-line algorithm with k taxies to an off-line one with k taxies. At first we analyze the unweighted case of the k-taxi problem, then the case in which the weights are nearly equal is studied and the following theorem is obtained.

Theorem 6. *If $\mu_1 > \frac{\lambda}{1+\lambda}$ holds, there is no c-competitive algorithm for the weighed k-taxi problem in a given metric space M with at least $k+2$ points, where $c < \frac{2k \cdot \mu_1}{k+2}$.*

Proof. Without loss of generality, assume A is an on-line algorithm and that the k taxies start out at k different points. Let H (of size $k+2$) be a subset of the given metric space M, induced by the k initial positions of A's taxies and two other points. Define R, A's nemesis sequence on H, such that $R(i)$ and $R(i-1)$ are the two unique points in H not covered by A and a request $ri = d(R(i), R(i-1))$ occurs at time i, where all $i \geq 1$. That is to say at each step R requests the point just vacated by A. Then we have

$$\begin{aligned} C_{\mathrm{A}}(R) &= \sum_{i=1}^{m} C_{\mathrm{A}}(r_i) = \sum_{i=1}^{m} [d(R(i+1), R(i)) + d(R(i), R(i-1))] \\ &= 2 \sum_{i=1}^{m-1} d(R(i+1), R(i)) + d(R(m+1), R(m)) + d(R(1), R(0)) \end{aligned} \tag{16}$$

Let S be any k-element subset of H containing $R(1)$ but not $R(0)$. We can define an off-line algorithm $A(S)$ as follows: the taxies finally occupy the points in set S. To process a request $r_i = d(R(i), R(i-1))$, the following rule is applied: If S contains $R(i)$, move the taxi at $R(i)$ to $R(i-1)$ to complete the request, and update the S to reflect this change. Otherwise move the taxi at $R(i-2)$ to $R(i)$ first and then to $R(i-1)$ to complete the request, and update S to reflect this change. Then the expected cost of the off-line algorithm is (more details please refer to paper[13])

$$\begin{aligned} C_{\mathrm{EXP}}(R) &= \sum_{i=1}^{m} d(R(i), R(i-1)) + \frac{1}{k} \cdot \sum_{i=2}^{m} d(R(i-2), R(i)) \\ &\leq \frac{k+2}{k} \cdot \sum_{i=1}^{m-1} d(R(i+1), R(i)) + \frac{k+1}{k} \cdot d(R(1), R(0)) \\ &\quad - \frac{1}{k} \cdot d(R(m), R(m-1)) \end{aligned} \tag{17}$$

When $\mu_1 > \frac{\lambda}{1+\lambda}$ holds, the weights are nearly equal. Form lemma 3, we know the weighted case is similar to the unweighted case. Because every taxi has a weight which is not more than 1, (17) is still holds for the weighted case and $C_{\mathrm{A}}(R)' \geq \mu_1 \cdot C_{\mathrm{A}}(R)$, where $C_{\mathrm{A}}(R)'$ denotes the cost of algorithm A for the

weighted case. After some mathematical manipulation (e.g., let $m \to \infty$), we obtain

$$\frac{C_{\mathrm{A}}(R)'}{C_{\mathrm{EXP}}(R)} \geq \mu_1 \cdot \frac{C_{\mathrm{A}}(R)}{C_{\mathrm{EXP}}(R)} \geq \frac{2k \cdot \mu_1}{k+2} \tag{18}$$

Finally, there must be some initial set whose performance is often no worse than the average of the costs. Let S be this set, and $A(S)$ be the algorithm starting from this set. □

Taking theorem 1 and theorem 6 together, we obtain the following theorem.

Theorem 7. *For the weighed k-taxi problem in a given metric space M with at least $k+2$ points, if $\mu_1 < \frac{\mu_2}{1+\lambda}$ holds, there exists a 1-competitive algorithm; if $\mu_1 > \frac{\lambda}{1+\lambda}$ holds, there is no c-competitive algorithm for $c < \frac{2k \cdot \mu_1}{k+2}$.*

6 Conclusion

In this paper, we consider the weighted case of k-taxi problem. Some on-line algorithms are designed to address the problem and the competitive analysis are given in detail. Furthermore, the lower bound of competitive ratio for some special cases is also discussed in this paper.

However, there are still many theoretical problems that need to be studied further. For example, although we have get a lower bound of competitive ratio for the case $\mu_1 > \frac{\lambda}{1+\lambda}$, and showed that if $\mu_1 < \frac{\mu_2}{1+\lambda}$ holds, there exists a 1-competitive algorithm, the lower bound of other cases and the optimal lower bound are still open.

Furthermore, concerning that the request cannot be complete instantaneously and the process of satisfying request must last a period of time, an essential assumption in this paper that when a new service request occurs k taxies are all free cannot come into existence in reality. Without this assumption or taking time into account, the problem will be more complex. It needs further investigation.

Acknowledgements. The work was partly supported by the National Natural Science Foundation of China (70671004, 70401006, 70521001), Beijing Natural Science Foundation (9073018), Program for New Century Excellent Talents in University (NCET-06-0172) and A Foundation for the Author of National Excellent Doctoral Dissertation of PR China.

References

1. Manasse, M.S., McGeoch, L.A., Sleator, D.D.: Competitive algorithms for server problems. Journal of Algorithms (11), 208–230 (1990)
2. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communication of the ACM 28, 202–208 (1985)

3. Fiat, A., Ricklin, M.: Competitive algorithms for the weighted server ploblem. Theoretical Computer Science 130, 85–99 (1994)
4. Koutsoupias, E., Taylor, D.: The CNN problem and other k-server variants. In: Reichel, H., Tison, S. (eds.) STACS 2000. LNCS, vol. 1770, pp. 581–592. Springer, Heidelberg (2000)
5. Chrobak, M., Sgall, J.: The weighted 2-server problem. Theoretical Computer Science 324, 289–312 (2004)
6. Epstein, L., Imreh, C., van Stee, R.: More on weighted servers or FIFO is better than LRU. In: Diks, K., Rytter, W. (eds.) MFCS 2002. LNCS, vol. 2420, pp. 257–268. Springer, Heidelberg (2002)
7. Xu, Y.F., Wang, K.L.: On-line k-taxi problem and competitive algorithm. Journal of Xi'an Jiaotong University (1), 56–61 (in Chinese) (1997)
8. Xu, Y.F., Wang, K.L., Ding, J.H.: On-line k-taxi scheduling on a constrained graph and its competitive algorithm. Journal of System Engineering (12), 361–365 (in Chinese) (1999)
9. Xu, Y.F., Wang, K.L., Zhu, B.: On the k-taxi problem. Information 2(4) (1999)
10. Xin, C.L., Ma, W.M.: Scheduling for On-line Taxi Problem On a Real Line and Competitive Algorithms. In: ICMLC 2004. Proceedings of the Third International Conference on Machine Learning and Cybernetics, pp. 3078–3084 (2004)
11. Ma, W.M., Wang, K.: On-line taxi problem on the benefit-cost graphs. In: ICMLC 2006. Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, pp. 900–905 (2006)
12. Ma, W.M., Wang, K.: On the On-Line k-Truck Problem with Benefit Maximization. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288, pp. 660–669. Springer, Heidelberg (2006)
13. Ma, W.M., Xu, Y.F., You, J., Liu, J., Wang, K.L.: On the k-truck scheduling problem. International Journal of Foundations of Computer Science 15(1), 127–141 (2004)

Model Futility and Dynamic Boundaries with Application in Banking Default Risk Modeling

Xiao Jun Shi[1,2]

[1] School of Economics and Management, Beijing University of Aeronautics and Astronautics, Beijing, PRC, 100083
[2] Institute of Finance and Banking, Chinese Academy of Social Science, Beijing, PRC, 100732
Xiao Jun Shi, sxjstein@yahoo.com.cn

Abstract. This paper presents a novel concept of model futility to capture the dynamic feature of modeling risk. Three key components, monitoring statistic, possibility of being futile and futility boundaries are specified. We apply this approach in banking default risk modeling monitoring to solve the optimal pairing ratio problem. Its effectiveness and efficiency are demonstrated by comparison with testing approach which is currently prevailing in banking models validation.

Keywords: Model Risk, Futility, Banking.

1 Introduction

Credit risk modeling is an exciting area for credit professionals and statisticians. Especially as The New Basel Accord will be enforced globally after 2006, new analytical models to measure, manage, and control credit risk are created and promoted more frequently. One of the fundamental problems of the applications of these models is how to confirm the efficacy of them? Nowadays, statistical testing is the most applied method of model validation in banking risk modeling. But most used testing methods such as significance testing, good-of-fitness testing, or more recently used ROC methods borrowed from radar image analysis, among others (for an excellent exhibition of these techniques, refer to [1]), are all static methods. They cannot capture the dynamic feature of model inefficacy, or model risk. Say it in other way, such methods cannot answer questions as "when will a particular model be futile?" This paper presents a novel concept of *model futility* to capture the dynamic feature by resembling the trajectory of model risk statistic as a stochastic process. By this approach, banks can conduct a continuous monitoring of the efficacy of models. We call such an approach of controlling model risk as a *clinical* approach. The critical problems we aim to tackle are how to present a suitable stochastic modeling of the model's inefficacy, and what are the low boundaries of model futility. The following sections are arranged as: section 2 presents typical evidences of dynamic feature of model risk in banking; section 3 resembles model risk process as a Brownian motion; section 4 is the most important part of this paper, in this section, model futility is defined, monitoring statistic is designed, and a direct method to compute futility

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 163–174, 2007.

boundaries is presented; section 5 gives a more convenient algorithm to determine low boundaries based on conditional power; in section 6, we apply the model futility method in Logistic default risk modeling to solve the problem of optimal pairing ratio; and section 7 concludes.

2 Dynamic Feature of Model Risk in Banking

The mostly studied type of model risk in banking is overfitting errors ([2]). Broadly speaking, there are three kinds of overfitting, that is, the so-called curse of dimensionality resulting from estimating a model with highly flexible function form; testing of many possible predictors and choose the one that works best on a specific sample; fitting a feature of one's data collection mechanism rather than an underlying economic relationship. All of these three kinds of overfitting are dynamic in nature which means they are related to the dynamic composition of the model development sample. Just as [3] states: "A good credit model must perform well over time … A model's performance should be verified and validated on a continuous basis. (pp188~189) ". Now that model risk in banking is dynamic, we naturally require a dynamic monitoring approach of the performances of the models. Specifically, we require such an approach that can incorporates: (1) the degree of model risk measurement, (2) the degree of variability of the model risk measurement over time, (3) the likelihood of important excursions in the trajectory over time, (4) the memoryless property.

3 Resemblance of Model Risk Trajectory as a Brownian Motion

Fortunately, we have already had such a tool which can satisfy the requirements stated above. The well-known Brownian motion is such a tool. Review of Brownian motion reveals that the properties of this type of movement closely align with model risk process in banking. The dependence of the location of a Brownian element on its previous position matches with assessment of the behavior of a model risk measurement that is built on accumulating data over time. The fact that Brownian motion follows a normal distribution matches nicely with normality assumption in model errors researches. In addition, some systematic factors such as Basel II driven data accumulation in banking system will universally improve the quality of the credit risk models. And the drift term in Brownian motion can perfectly capture such systematic effects over the course of credit modeling evolution.

The most relevant properties of Brownian motion can be applied in model risk monitoring is *the conditionality property*, one can consult any of the modern probability and stochastic course books (e.g. [4], among others) for the details of these two properties. Here we only present the results.

We define a random variable of model risk measurement over time obeying standard Brownian motion as $B(t)$. Using the *conditionality properties* of Brownian motion we can calculate the probability of model risk greater than a predefined critical value b at time t_2 if we know the level of model risk is a at current time t_1 as following:

$$P\{B(t_2) > b \mid B(t_1) = a\} = 1 - \Phi\{\frac{b-a}{\sqrt{t_2 - t_1}}\} \tag{1}$$

4 Model Futility

The rationale behind the determination of model futility is approximating certain model risk measure as a Brownian motion and comparing the observed trajectory of the model risk with a predefined Brownian motion under the null hypothesis of futility or the alternative hypothesis of efficacy.

There are three key components in the determination of model futility. The first one is the proper measure of the model risk d and converting its dynamics into a Brownian motion. The second one is how possible it is if the model risk dynamic behaves as a futile Brownian motion with elapse of information time[1] t. The last but not the least one is the boundary values b of model futility. Now we show how to specify these three components in a very basic situation.

4.1 Research Design Settings

A model risk investigator is interested in the efficacy of certain loan default probability p prediction model over time in a bank. In the original design, the investigator should examine N consumers over a time span of T.The investigator is interested in conducting an interim review when the model prediction results are available for only $N_t = \sum_{i=1}^{t} n_i$, $t = 1,\ldots,T$, where n_i means the number of loans examined at time of i.

4.2 Monitoring Statistic

Let x_i be default status of the i^{th} loan. Let $x_i = 1$ if the model prediction of the default status is right, that is, if the model prediction of default possibility of a consumer is greater than 0.5 and the consumer really default in one year, or the model result is less than 0.5 and the consumer does not default in one year. On the other hand, let $x_i = 0$. By the end of the study, the best estimator of model prediction efficacy is:

$$\bar{X} = \sum_{i=1}^{N} x_i \Big/ N \tag{2}$$

For the null hypothesis (or the alternative hypothesis) of model futility, we can predefine a suitable value for the mean of x_i denoted as p, its variance is $p(1-p)$. By central limit theorem, if we can write that:

$$\sum_{i=1}^{N_i} x_i \sim N(N_i p, N_i p(1-p)) \tag{3}$$

[1] Information time is quite different from calendar time. In the context of model risk, information time usually refers to the portion of observed sample of the total number of the sample. But the calendar time refers the portion of the time of the existence of the current model of its whole life span.

By (3), it is easy to know:

$$\frac{\sum_{i=1}^{N_i} x_i}{\sqrt{N}\sqrt{p(1-p)}} \sim N(\frac{N_i p}{\sqrt{N}\sqrt{p(1-p)}}, \frac{N_i}{N}) \tag{4}$$

Variance in (4) is exactly the information time IT. Note that, at the beginning of the investigation $IT = 0$ and at the conclusion of the study $IT = 1$. This gradual accrual of information over time is exactly what information time is design to measure.

If the null hypothesis is true, the realized model efficacy is equal to p, and then we can see that

$$ms = \frac{\sum_{i=1}^{N_i} x_i - N_i p}{\sqrt{N}\sqrt{p(1-p)}} \tag{5}$$

is normally distributed with mean of 0 and variance of N_i/N and follows standard Brownian motion. This is the monitoring statistic that we are looking for.

4.3 Possibility of Being Futile

When we have obtained monitoring statistic following Brownian motion, the computation of possibility of being futile is quite straight. We first define the futile process as a standard Brownian motion with no drift term. At information time of $IT = N_i/N$, we observe that the value of monitoring statistic is m_i. Applying equation (1) directly, we know the possibility of the observed trajectory following the futile process is

$$P\{B(t) > m_i\} = 1 - \Phi\{\frac{m_i}{\sqrt{IT}}\} \tag{6}$$

4.4 Dynamic Boundaries

The concept of boundary is central in the dynamic determination of model futility. It is a demarcation divides the space of monitoring statistic into futility region and continuation region. It can take the shape of level line, sloping lines or curves. It is a prospectively declared decision rule which can solve the problem of accumulating type I error accompanying multiple hypothesis testing on the same sample ([5]). Inspired by the workings of [6], [7], and [8] we propose a sloping shaped boundary in banking model risk monitoring as in fig.1. It has decreasing boundary values as information time passing which reflects the fact that with information time elapsing more information will accumulate and the boundaries to determine futility should be more accurately set but not like at the beginning of the process the boundaries can be only loosely set with small fraction of sample information used.

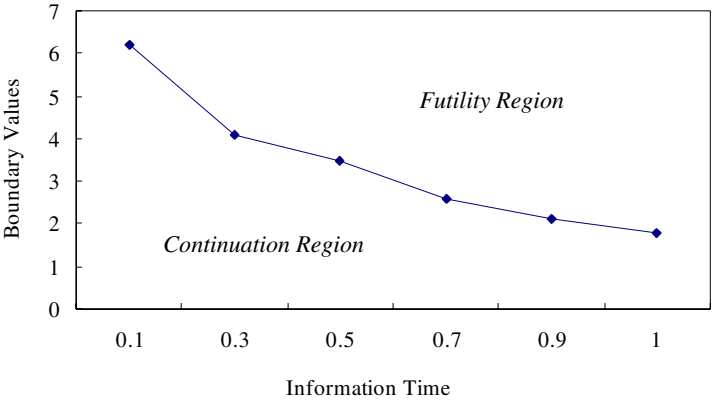


Fig. 1. Sloping Model Futility Boundary. This is only an illustrating figure.

The critical problem now is how to distribute a selected overall type I error α across the interim monitoring points at different information time IT_i as $\alpha_i, i = 1,2,\ldots,M$. Here we applied the well-known *Alpha Spending Function* method in clinical literature which was pioneered by [6], [7] and nicely elaborated by [8].

The specific steps to determine boundaries are as followings:

Step 1: Select an overall type I error α for the monitoring process. Set M points of investigation at information time $IT_i, i = 1,\ldots,M$.

Step 2: Using *alpha spending function*[2] to distribute α across investigation points and set the value of interim monitoring alpha levels $\alpha_1,\ldots,\alpha_M$.

Step 3: Set the boundary values for the i^{th} monitoring point with the constraint of distributed type I error level α_i .

Step 4: Judging whether the monitoring statistic exceeds the boundary value at i^{th} monitoring point, if it exceeds, then the current model is futile.

For a simple case with 3 monitoring points, we present the details of step 3. Boundary values are denoted as b_1, b_2, b_3 . As we have known $\alpha_1, \alpha_2, \alpha_3$, we can obtain the boundary values by:

$$\begin{aligned} &P\{B(t_1) \geq b_1\} = \alpha_1 \\ &P\{B(t_1) < b_1 \cap B(t_2) \geq b_2\} = \alpha_2 \\ &P\{B(t_1) < b_1 \cap B(t_2) < b_2 \cap B(1) \geq b_3\} = \alpha_3 \end{aligned} \tag{7}$$

[2] For space limited, the details of alpha spending function are omitted, readers who are interested can consult [7] and [8] for details.

It is easy to solve b_1 as

$$b_1 = z_{1-\alpha_1} \cdot \sqrt{t_1} \tag{8}$$

where $z_{1-\alpha_1}$ means $(1-\alpha_1)$ percentage points of a standard normal distribution.

However, the second line of expression (7) introduces new complexities. It requires the joint distribution of $B(t_1)$ and $B(t_2)$ to determine boundary value b_2.

Lemma 1. *The correlation coefficients between* $B(t_1)$ *and* $B(t_2)$ *is* $\rho = \sqrt{t_1/t_2}$, *and the joint distribution of* $B(t_1)$ *and* $B(t_2)$ *is*

$$f_{B_{t_1},B_{t_2}}(B_{t_1},B_{t_2}) = \frac{1}{2\pi\sqrt{t_1(t_2-t_1)}} e^{-(t_2B_{t_1}^2 - 2t_1B_{t_1}B_{t_2} + t_1B_{t_2}^2)/2t_1(t_2-t_1)} \tag{9}$$

For space limited, showings of lemmas in this paper are all omitted; but they can be available from the corresponding author by request.

As we know the joint distribution of $B(t_1)$ and $B(t_2)$, the boundary value b_2 can be obtained by solving

$$\int_{-\infty}^{b_1}\int_{b_2}^{\infty} \frac{1}{2\pi\sqrt{t_1(t_2-t_1)}} e^{-(t_2B_{t_1}^2 - 2t_1B_{t_1}B_{t_2} + t_1B_{t_2}^2)/2t_1(t_2-t_1)} dB_{t_1} dB_{t_2} = \alpha_2 \tag{10}$$

Similar procedure can be followed to solve b_3.

5 Conditional Power and Low Boundaries

The procedure above is quite complex and difficult to realize in practical application. If we look at the boundaries problem in a forward way, the solution will be straight and a closed-formed solution can be obtained.

Suppose that we know the value of monitoring statistic at information time IT_1 is s_1. We want to know the possibility that monitoring statistic at information time IT_2 is greater than a predefined value s_2 for $0 \le IT_1 \le IT_2 \le 1$, or

$$P\{ms(IT_2) \ge s_2 \,|\, ms(IT_1) = s_1\} \tag{11}$$

We convert it to an event involving the monitoring statistic following Brownian motion and apply the knowledge of Brownian motion conditional on the past. We can get:

$$P\{ms(IT_2) \ge s_2 \,|\, ms(IT_1) = s_1\} = 1 - \Phi\{\frac{\sqrt{IT_2}\cdot s_2 - \sqrt{IT_1}\cdot s_1}{\sqrt{IT_2 - IT_1}}\} \tag{12}$$

The proof is as following:

$$
\begin{aligned}
& P\{ms(IT_2) \geq s_2 \,|\, ms(IT_1) = s_1\} \\
&= P\{\sqrt{IT_2} \cdot ms(IT_2) \geq \sqrt{IT_2} \cdot s_2 \,\Big|\, \sqrt{IT_1} \cdot ms(IT_1) = \sqrt{IT_1} \cdot s_1\} \\
&= P\{B(IT_2) \geq \sqrt{IT_2} \cdot s_2 \,|\, B(IT_1) = \sqrt{IT_1} \cdot s_1\} \\
&= P\{B(IT_2 - IT_1) \geq \sqrt{IT_2} \cdot s_2 - \sqrt{IT_1} \cdot s_1\} \\
&= P\{N(0, IT_2 - IT_1) \geq \frac{\sqrt{IT_2} \cdot s_2 - \sqrt{IT_1} \cdot s_1}{\sqrt{IT_2 - IT_1}}\} \\
&= 1 - \Phi\{\frac{\sqrt{IT_2} \cdot s_2 - \sqrt{IT_1} \cdot s_1}{\sqrt{IT_2 - IT_1}}\}
\end{aligned}
$$

This is known as conditional power computation under the null hypothesis of standard Brownian motion representing futile process, or $C_p(H_0)$.

By conditional power computation method, we can easily solve the problem of boundary value. In this framework, we can convert the boundary value problem into the following one: what is the value of b at information time $IT_i, i = 1, \ldots, M$ such that the probability that the test statistic falls in the futility region at the conclusion of the study is at least some value γ (i.e. $\gamma = 95\%$). That is

$$P\{ms(1) \geq z_{1-\alpha/2} \,|\, ms(IT_i) = b_i\} = \gamma \tag{13}$$

where α is the significance level of the testing at the conclusion of the investigation process. $z_{1-\alpha/2}$ is the critical value of the critical region at the conclusion of the investigation process.

Lemma 2 gives the closed-formed solution of boundary value b.

Lemma 2. *In conditional power framework, given* α, γ *and* IT_i, *we can obtain the boundary values* b_i *for* i^{th} *interim investigation by*

$$b_i = \frac{z_{1-\alpha/2} - z_{1-\gamma}\sqrt{1 - IT_i}}{\sqrt{IT_i}} \tag{14}$$

6 Application in Banking Default Risk Modeling Monitoring

Logistic modeling of default risk has been widely accepted in banking industry. But there is a fundamental question left unanswered until now, that is, what is the optimal pairing ratio of the healthy companies to the default companies when designing the modeling sample? In an information perspective or a Bayesian view, pairing ratio will definitely affect modeling efficacy. In this paper, we try to apply the model futility and boundary values method proposed above to solve this fundamental problem.

6.1 Rationale

In order to find the optimal pairing ratio, we first design several typical scenarios of pairing ratio of healthy companies to default ones in an increasing order. We take these scenarios as different *interim investigation points* in model futility framework. In this way, we change the optimal pairing ratio problem into an *optimal stopping* problem of Logistic default risk modeling. That is, when the monitoring statistic first falls into the continuation region, we set the optimal pairing ratio by this point.

6.2 The Sample

It is a well-known fact that in China there is no usable default (or bankruptcy) data for many reasons. We commonly use ST (Specially Treated) companies in stock market as proxy of default ones. By the definition of ST, this kind of practice is quite acceptable. We collected the financial data of all the listed A share companies in Shanghai Stock Exchanges until 2004. All the data come from StockStar (www.stockstar.com.cn) which is a designated website by China Securities Regulation Commission for listed companies' information disclosure. During the period from January 1, 2003 to August 9, 2004, there are totally 52 ST companies, 20 in 2003, 32 in 2004. We use all the 20 ST companies in 2003 and randomly selected 19 ST companies in 2004 to build the sample of proxy default companies for model development. And the remaining 13 ST companies in 2004 were concluded in test sample. In order to avoid over-estimation problem argued in Ohlson (1980), we should use financial data two years before to development the model. That is, we should use data in 2001 to predict what will happen in 2003, and data in 2002 to predict the situation in 2004. Finally, there are totally 1080 healthy companies included in development sample.

6.3 Interim Points

Now we have totally 39 proxy default companies in development sample. We design 5 typical pairing ratios, that is, 1:1.05, 1:2, 1:3, 1:5, and the last one including all the collected healthy companies. As we know there are totally 1080 healthy companies, it is easy to calculate information time of interim points corresponded to these 5 typical pairing ratios. Table 1 gives the results:

Table 1. Information times of different interim ponits

Pairig Ratios	Number of healthy companies	Information time
1:1.05	41	0.04
1:2	78	0.07
1:3	117	0.11
1:5	195	0.18
All healthy companies	1080	1.00

6.4 Hosmer-Lemeshow Testing Results

The details of financial indicators selection, Logistic model parameters estimation and tests of good-of-fitness are not presented here for it is not the main purpose of this paper. Here we only present the *Hosmer-Lemeshow* testing results in table 2.

Table 2. Presents Hosmer-Lemeshow testing results at different information time. The first row of the table presents information times (IT) of different interim points corresponding to 5 typical pairing ratios. In the second row, 0 represents healthy company, 1 represents default company. In the third row, OB means observed numbers of certain type of companies, PR means Logistic model predicted frequencies of certain type of companies, TA means total number of companies. The last two rows present values of χ^2 statistic and p-values.

IT	0.04					0.07					0.11				
	0		1			0		1			0		1		
	OB	PR	OB	PR	TA	OB	PR	OB	PR	TA	OB	PR	OB	PR	TA
1	7	7.494	1	.506	8	11	11.782	1	0.218	12	16	16.000	0	.000	16
2	7	6.858	1	1.142	8	12	11.329	0	0.671	12	16	15.988	0	.012	16
3	7	6.288	1	1.712	8	10	10.635	2	1.365	12	16	15.934	0	.066	16
4	5	5.233	3	2.767	8	12	10.086	0	1.914	12	15	15.782	1	.218	16
5	8	4.678	0	3.322	8	11	9.626	1	2.374	12	15	15.483	1	.517	16
6	3	4.172	5	3.828	8	8	8.854	4	3.146	12	16	14.835	0	1.165	16
7	3	3.070	5	4.930	8	8	7.870	4	4.130	12	14	13.267	2	2.733	16
8	0	2.142	8	5.858	8	4	5.490	8	6.510	12	8	8.755	8	7.245	16
9	0	.910	8	7.090	8	1	2.166	11	9.834	12	1	.956	15	15.044	16
10	1	.154	7	7.846	8	1	.162	8	8.838	9	0	.000	12	12.000	12
χ^2		22.475					13.421					5.030			
Sig.		0.004					0.098					0.754			

IT	0.18					1				
	0		1			0		1		
	OB	PR	OB	PR	TA	OB	PR	OB	PR	TA
1	23	23.000	0	0.000	23	112	111.905	0	0.095	112
2	23	22.994	0	0.006	23	112	111.619	0	0.381	112
3	23	22.969	0	0.031	23	111	111.314	1	0.686	112
4	22	22.912	1	0.088	23	112	110.985	0	1.015	112
5	22	22.814	1	0.186	23	111	110.659	1	1.341	112
6	23	22.507	0	0.493	23	109	110.217	3	1.783	112
7	23	21.925	0	1.075	23	111	109.634	1	2.366	112
8	23	20.879	0	2.121	23	109	108.784	3	3.216	112
9	13	14.618	10	8.382	23	108	106.942	4	5.058	112
10	0	0.382	27	26.618	27	85	87.941	26	23.059	111
χ^2	17.998					4.104				
Sig.	.021					.848				

6.5 Boundaries and Values of Monitoring Statistic

We set α at 0.05, γ at 0.95. By (14) in lemma 2, it is easy to calculate boundaries for 5 interim points. In order to calculate values of monitoring statistic, we should first select a proper level of p. By experience, we set p at 0.01. For every interim point, differences between observed and predicted values of two types of companies are calculated, and then sum up the absolute values of these differences, we get

$\sum_{i=1}^{N_i} x_i$ in (6). With p and $\sum_{i=1}^{N_i} x_i$ known, it is easy to obtain values of monitoring statistic for 5 interim points. Table 3 and Fig. 2. present boundaries and values of monitoring statistic. It is quite straight to judge from Fig. 2. that the optimal pairing ratio is 1:3, because only in this case monitoring statistic falls into the continuation region.

Table 3. Boundaries and monitoring statistic values of 5 interim points

Information times	0.04	0.07	0.11	0.18	1.00
Boundaries	9.74	7.07	5.77	4.48	1.96
Monitoring Statistic	12.03	11.58	4.23	7.92	4.34

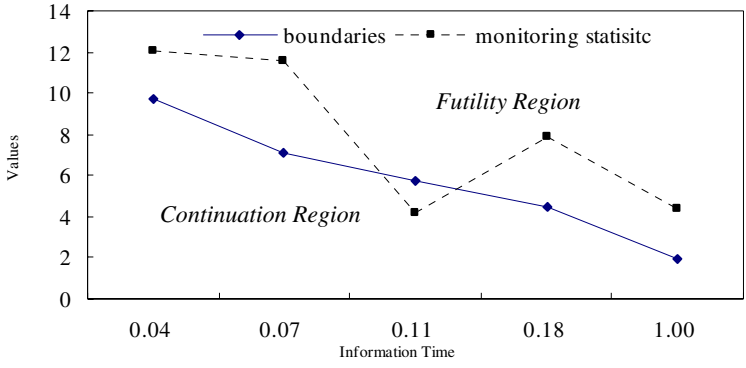


Fig. 2. Presents low boundaries and values of monitoring statistic. It shows that boundaries decrease with information time increasing because of more information arriving. But there is volatility in the trajectory of monitoring statistic at different information times.

6.6 Comparison with Testing Approach

If we want to solve this optimal pairing ratio problem by testing approach, we should compare expected losses in all these 5 pairing situations. So we should first estimate probabilities and loss function of error I and error II. We set three cutoff points for every pairing ratio, that is, 0.5, 0.647 and 0.341. And loss of error I is 1, 20, and 38 times loss of error II when cutoff points are 0.5, 0.647 and 0.341 respectively. Table 4 presents error rates and losses in different situations. Fig. 3 presents the curves of total error, error I and expected loss in different situations.

In order to decide which pairing ratio is optimal in this framework, we take elimination and minimization method. First, eliminate all situations with any kind of error rate more than 0.35. Second, we follow expected loss minimizing principal to find the optimal pairing ratio. Finally, we pick up 1:3 as the optimal pairing ratio.

This answer is the same as the one we've got in model futility boundaries framework.

Comparing with the method we propose in this paper, the testing approach is clearly more tedious and subjective. Additionally, there are suppositions which demand past experiences and judgments such as loss function specification if we apply testing approach. Although futility boundaries approach also require subjective specification of α and γ, but it is much easier to decide, because these two parameters represent the accuracy level of the model, they should naturally be set by the investigator. In one word, futility boundaries approach captures the dynamic feature of the model risk; and it is more efficiency than the testing approach.

Table 4. Presents error rates, loss specification and expected losses in 5*3 situations

Pairing Ratio	Cutoff Points	Total Error	Error I	Error II	Loss of Error I	Loss of Error II	Expected Loss
1:1	0.5	0.154	0.154	0.154	1	1	0.308
	0.647	0.078	0.308	0	20	1	3.077
	0.341	0.269	0.077	0.462	38	1	3.385
1:2	0.5	0.154	0.308	0	1	1	0.308
	0.647	0.231	0.462	0	20	1	9.24
	0.341	0.162	0.154	0.231	38	1	6.083
1:3	0.5	0.269	0.154	0.385	1	1	0.539
	0.647	0.115	0.154	0.077	20	1	3.157
	0.341	0.231	0.077	0.462	38	1	3.346
1:5	0.5	0.192	0.154	0.231	1	1	0.386
	0.647	0.154	0.154	0.154	20	1	3.234
	0.341	0.231	0.077	0.385	38	1	3.311
All healthy Companies	0.5	0.346	0.692	0	1	1	0.692
	0.647	0.346	0.692	0	20	1	13.846
	0.341	0.269	0.077	0.462	38	1	20.462

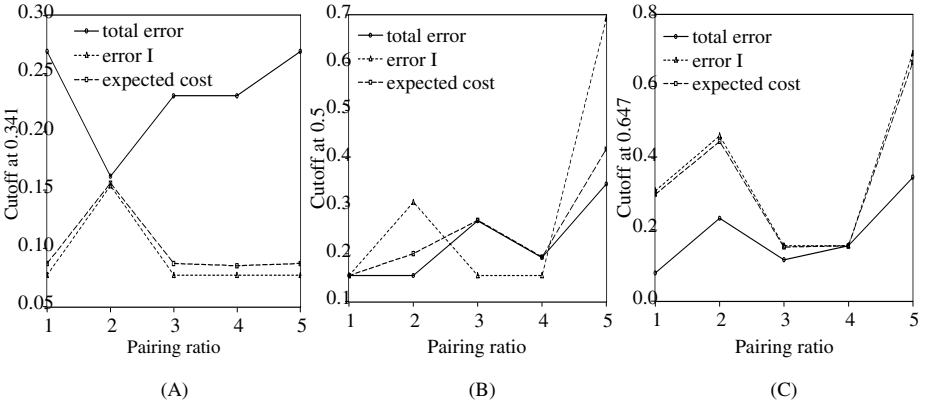


Fig. 3. Presents total error rate, possibilities of error I and expected costs in different situations. (A), (B), and (C) present the situations of cutoff point at 0.341, 0.5 and 0.647 respectively. On the horizontal axis, 1 to 5 represent 1:1.05, 1:2, 1:3, 1:5, and all healthy companies scenarios respectively.

7 Conclusion and Remarks

In this paper we present a novel concept of model futility to capture the dynamic features of model risk. We also specify the approaches to compute possibility of being futile and the futility boundaries in conditional power framework. By application in Logistic default risk modeling monitoring in banking, effectiveness and efficiency of this method can be proved. In the future research, there are two directions demanding attention. The first one is how to combine the prior knowledge of the distribution of monitoring statistic with the method we have presented here in Bayesian way. The second one is more complicated, if the futile process is not a standard Brownian motion but with jumps or other characteristics, then how to modify the method we are now presenting?

Acknowledgement. Financial supports from National Natural Science Foundation of China (Grant NO. 70502005), BOSHIDIAN Foundation from Ministry of Education of China (Grant NO. 2005006004), National Social Science Foundation of China (Grant NO. 05CJL003) are acknowledged.

References

1. van Deventer, D., Imai, K.: Credit Risk Models and The Basel Accords. John Wiley & Sons, Asia (2003)
2. Dwyer, D.W.: Examples of Overfitting Encountered When Building Private Firm Default Prediction Models, April 12, 2005. Moody's KMV Working Paper (2005)
3. Caouette, J.B., Altman, E.I., Narayanan, P.: Managing Credit Risk: The Next Great Financial Challenge. John Wiley & Sons, New York (1998)
4. Ross, S.M.: Stochastic Processes, 2nd edn. John Wiley & Sons, New York (1983)
5. Moyé, L.A.: Statistical Monitoring of Clinical Trials: Fundamentals for Investigators. Springer, Heidelberg (2005)
6. O'Brien, P.C., Fleming, T.R.: A multiple testing precdure for clinlical trials. Biometrics 35, 549–556 (1979)
7. Lan, K.K., DeMets, D.L.: Discrete Sequential Boundaries for Clinical Trials. Biometrics 70, 659–663 (1983)
8. Fleming, T.R., Green, S.J., Harrington, D.P.: Considerations for Monitoring and Evaluating Treatment Effects in Clinical Trials. Controlled Clinical Trials 5, 55–56 (1994)
9. Ohlson.: Financial Ratios and the Probabilistic Prediction of Bankruptcy. Accounting Research 18, 109–131 (1980)

On the Minimum Risk-Sum Path Problem

Xujin Chen, Jie Hu, and Xiaodong Hu*

Institute of Applied Mathematics
Chinese Academy of Sciences
P.O. Box 2734, Beijing 100080, China
{xchen, hujie, xdhu}@amss.ac.cn

Abstract. This paper presents efficient algorithms for the *minimum risk-sum path problem* which arises in a variety of applications. Given a source-destination node pair in a network $G = (V, E)$, where each link e in G can be traveled using time x_e in a prespecified interval $[l_e, u_e]$ while taking risk $\frac{u_e - x_e}{u_e - l_e}$, the *minimum risk-sum path problem* is to find a path in G from source to destination, together with an assignment of travel times along each link on the path, so that the total travel time of the path is no more than a given constant and the risk sum over the links on the path is minimized. In this paper, we solve the minimum risk-sum path problem optimally in $O(|V|^3|E|)$ time.

1 Introduction

Path planning under uncertainty has been extensively studied in computer science and operation research due to its applications in diverse areas such as communication routing and transportation engineering (see, e.g., [11,15,14]). The omnipresent uncertainty of travel times on network links could result from network congestions, hardware failures, traffic jams or accidents, temporary construction projects, weather conditions. The client/traveler, because of the lack of updated on-line information on network links (cables, roads, etc.), will have to consider possible ranges of the travel times on network links in making the appropriate route choice in a risk averse manner, where route choice decisions often involve tradeoffs between travel times and risks to be taken.

Continuing the efforts devoted to path planning under uncertainty, we study in the paper the minimum risk-sum path problem for which few existing models in literature accurately represent the aim of decision maker. The *minimum risk-sum path problem* consists in finding a path of minimum risk sum under a total travel time constraint. Many practical situations fall within the framework of this problem. For example, a traveler wishes to travel within time K from a place (*source*) to another place (*destination*) in a transportation network. On a road e, the traveler could ask the driver to go through the road using time x_e (between the least possible travel time l_e and the most possible travel time u_e),

* Supported in part by the NSF of China under Grant No. 70221001, 60373012 and 10531070.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 175–185, 2007.

while taking an amount $r(x_e)$ of risk (e.g., over-speed risk). It is considered no risk, i.e., $r(x_e) = 0$, if $x_e = u_e$; a (full) risk, i.e., $r(x_e) = 1$, if $x_e = l_e$; and $0 \leq r(x_e) \leq 1$ for $x_e \in [l_e, u_e]$ in general. It is desirable to find a route P from source to destination and an assignment of travel times along each road on P so that the traveler takes risks $\sum_{e \text{ on } P} r(x_e)$ as few as possible, while assuring the arrival at destination using time $\sum_{e \text{ on } P} x_e \leq K$. Interestingly, the risk $r(x_e)$ can also be understood as a premium paid by the traveler to speed up the journey and assure the travel time x_e (on e) in mind.

Related work. Previous studies on path planning under uncertainty include robustness optimization [11] and stochastic applications [4].

The robustness approach to decision making in environments of data uncertainty also structures the uncertainty by taking the arc lengths as intervals defined by known lower and upper bounds. Under the most popular robustness criterion – *robust deviation* (*minimax regret*) *criterion*, the *robust shortest path problem* consists in finding among all the paths from source to destination the one, that over all the possible realizations of arc lengths, minimizes the maximum deviation of the path length from the length of the shortest path in the corresponding realizations, where a *realization* means an arc length taking a value in its interval. The NP-hardness of the robust shortest path problem [17] explains the reason of the lack of efficient algorithms for the problem [10,14]. Kouvelis, Yu and Gu [11,16] studied the *robust shortest path problem with a discrete scenario set* in which each *scenario* represents a possible realization of the arc lengths. They proved that the problem is NP-hard (strongly NP-hard) with a bounded (unbounded) number of scenarios.

The probability tools provide another way of dealing with uncertainty in data. In the stochastic settings, the probability distributions are estimated/suggested for modeling stochastic arc travel times. Fan et al [1] proposed an adaptive heuristic for paths that maximize the probability of arriving on time [2]. Along a different line, lots of researchers [5,3,13] gave approximations and heuristics for expected shortest paths in stochastic networks with dynamic arc length distributions. Besides the requirement on knowledge of probability distribution functions, it is computationally intractable to solve the problems especially with nonlinear objective functions and discrete decision variables.

Our results. In addition to different objectives, a major and apparent difference between the minimum risk-sum path problem and the previous robustness and stochastic models resides in the requirement in the minimum risk-sum path problem of not only a path connecting source and destination, BUT ALSO an assignment of travel time along each link on the path. In this paper we bridge the gap of lacking accurate mathematical model for the minimum risk-sum path problem. Based on the close connection discovered between the minimum risk-sum path problem and the constrained shortest path problem (see [9] or Section 3.1 for its definition), we employ the dynamic programming procedures [7] for the latter and design $O(mn)$ time approximation algorithm and $O(m^2n^2)$ time exact algorithm for the former, where m (resp. n) is the number of links (resp. nodes)

in the network. The polynomial time solvability established for the minimum risk-sum path problem exhibits its essential difference from the aforementioned NP-hard problems studied in path planning under uncertainty.

Organization of the paper. In section 2, we present mathematical model for the minimum risk-sum path problem, and a *nice property* of some optimal solutions to the problem. In Section 3, we first give polynomial time constructions between the minimum risk-sum path problem and the constrained shortest path problem, then we use the construction to design efficient approximation and exact algorithms for the minimum risk-sum path problem that output solutions of the nice property. In section 4, we eliminate the technical assumptions made in section 2 and extend our results to more general settings. In Section 5, we conclude this paper with remarks on future research.

2 Model

We model the network as a loopless[1] directed graph $G = (V, E)$ with vertex set $V = V(G)$ of size $n = |V|$ and arc set $E = E(G)$ of size $m = |E|$, where vertex (resp. arc) corresponds to network node (resp. link). Each arc $e \in E$ is associated with an interval $[l_e, u_e]$ indicating the lower bound $l_e \in \mathbb{R}_+$ and upper bound $u_e \in \mathbb{R}_+$ of the travel time along e (from its tail to its head). For ease of description, we assume l_e, u_e are nonnegative integers and $l_e < u_e$, for all $e \in E$. (These assumptions will be eliminated latter in Section 4.) Thus we consider $l = (l_e : e \in E)^T$ and $u = (u_e : e \in E)^T$ as integral vectors in $\mathbb{Z}_+^E$. Recall that when traveling arc e using time x_e, the risk incurred is 1 if $x_e = l_e$ and 0 if $x_e = u_e$; it is nature to quantify the risk as $\frac{u_e - x_e}{u_e - l_e}$ for any $x_e \in [l_e, u_e]$. In other words, traversal at the lower bound incurs one unit of risk, and this falls off linearly to zero risk for a traversal at the upper bound. Given *source* $s \in V$ and *destination* $d \in V$, let $\mathcal{P}$ denote the set of (directed) paths in G from s to d. In the *Minimum Risk-sum Path Problem*, we are given a time bound $K \in \mathbb{R}_+$ which is a constant no less than $\min_{P \in \mathcal{P}} \sum_{e \in E(P)} l_e$, and our aim is to find a pair (P, x) such that

- $P \in \mathcal{P}$;
- $x \in \mathbb{R}_+^{E(P)}$, and $x_e \in [l_e, u_e]$ for all $e \in E(P)$; and
- $\sum_{e \in E(P)} x_e \leq K$.

We call such a pair (P, x) a *solution* to the minimum risk-sum path problem (on G) of *risk-sum* $\tau(P, x) = \sum_{e \in E(P)} \frac{u_e - x_e}{u_e - l_e}$. If, in addition, $\tau(P, x) = \min\{\tau(P', x') : (P', x')$ is a solution to the minimum risk-sum path problem on $G\}$, then we say that (P, x) is *optimal*. For technical reasons, $K \in \mathbb{Z}_+$ will be assumed till the end of Section 3.

The following lemma states a nice property of some optimal solution to the minimum risk-sum path problem that will play an important role in our algorithm design. Intuitively speaking, the property ensures that all arcs e, with at

[1] A loop in a directed graph is an arc with its tail and head being the same vertex.

most one exception, on a minimum risk-sum path can be traversed at, l_e or u_e, either of the extremes of their feasible interval.

Lemma 1. *There exists an optimal solution (P^*, x^*) to the minimum risk-sum path problem in which P^* has an arc $f \in E(P^*)$ such that $x^*_f \in [l_f, u_f]$, and $x^*_e \in \{l_e, u_e\}$ for all $e \in E(P^*) \setminus \{f\}$.*

Proof. Let (P^*, x^*) be an optimal solution to the minimum risk-sum path problem on G with the set $S_{(P^*,x^*)} := \{g : g \in E(P^*) \text{ and } x^*_g \in (l_g, u_g)\}$ containing arcs as few as possible. We prove that $|S_{(P^*,x^*)}| \leq 1$, and therefore such an optimal solution (P^*, x^*) satisfies the lemma.

Assume the contrary that two different arcs $e, f \in S_{(P^*,x^*)}$, satifies $u_e - l_e \leq u_f - l_f$. Take $\delta := \min\{u_e - x^*_e, x^*_f - l_f\}$ and define $x' \in \mathbb{R}_+^{E(P^*)}$ by $x'_e := x^*_e + \delta$, $x'_f := x^*_f - \delta$, and $x'_g := x^*_g$ for all $g \in E(P^*) \setminus \{e, f\}$. Then

$$x'_e = u_e \text{ or } x'_f = l_f,$$

$$\sum_{e \in E(P^*)} x'_e = \sum_{e \in E(P^*)} x^*_e \leq K,$$

$$\sum_{e \in E(P^*)} \frac{u_e - x'_e}{u_e - l_e} = \left(\sum_{e \in E(P^*)} \frac{u_e - x^*_e}{u_e - l_e} \right) - \frac{\delta}{u_e - l_e} + \frac{\delta}{u_f - l_f} \leq \sum_{e \in E(P^*)} \frac{u_e - x^*_e}{u_e - l_e}.$$

It follows from the definitions of δ and x' that (P^*, x') is an optimal solution to the minimum risk-sum path problem such that $S_{(P^*,x')}$ is a proper subset of $S_{(P^*,x^*)}$ with e or $f \in S_{(P^*,x^*)} - S_{(P^*,x')}$, contradicting the minimality of $S_{(P^*,x^*)}$. □

For briefness, we refer every optimal solution of the property in Lemma 1 as to a *simple optimal solution* to the minimum risk-sum path problem.

3 Algorithms

In this section, we first present a fast approximation algorithm which produces solutions of high quality to the minimum risk-sum path problem; then we extend this approximation to be an exact algorithm which solves the problem in $O(m^2n^2)$ time.

3.1 A Fast Approximation Algorithm

In a directed graph with real function π defined on its arc set, an arc with tail a and head b is written as (a, b), and a (directed) path P from vertex a to vertex b is called an *a-b path*, for which $\pi(P)$ represents the summation $\sum_{e \in E(P)} \pi(e)$. (The conceptions and notations extend to undirected graphs in a straightforward way.)

The constrained shortest path problem. Our algorithms rely on the establishment of a close connection between the minimum risk-sum path problem and the constrained shortest path problem [9], a special case of the extensively studied bicriteria network design problem [6,8,12]. Given a directed graph H in which every arc e is associated with two nonnegative integers: its *length* $c(e)$ and its *traversal time* $t(e)$, the *Constrained Shortest Path Problem* is to determine a shortest s-d path in H from the prespecified source vertex s to destination vertex d, under the constraint that the traversal time of the path does not exceed prespecified integer T. Considering c and t integral functions on the arc set $E(H) = E'$ of H, one may write the constrained shortest path problem on H (with parameters c, t, T) as the following combinatorial optimization problem (CSP_T):

$$\begin{aligned} OPT \;=\; & \min c(Q) \\ & \text{s.t. } t(Q) \le T \\ & \quad Q \text{ is an } s\text{-}d \text{ path in } H \end{aligned}$$

Dynamic programming technique provides us with exact algorithms (Algorithm A and Algorithm B in [7]) for the constrained shortest path problem, which are polynomial for the cases with arc lengths or traversal time bounded.

Theorem 1. *[7] The constrained shortest path problem (CSP_T) can be solved optimally*

(i) in $O(T|E'|)$ time by Algorithm A, whose output includes optimal solutions to ($CSP_{T'}$) on H with parameters c, t, T' for all $T' = 0, 1, \ldots, T$;
(ii) in $O(OPT|E'|)$ time by Algorithm B.

Polynomial time constructions. For the minimum risk-sum path problem on $G = (V, E)$, we construct in $O(m)$ time a directed graph $H = (V', E')$, length function $c \in \{0, 1\}^{E'}$, and traversal time function $t \in \mathbb{Z}_+^{E'}$ such that

- H has the same vertex set as G: $V' = V$;
- every arc $e = (a, b)$ in G corresponds to *two* arcs e_l, e_u in H both with tail a and head b: $E' = \{e_l, e_u : e \in E\}$;
- $c(e_l) = 1$, $c(e_u) = 0$, $t(e_l) = l_e$, $t(e_u) = u_e$, for every $e \in E$.

Given an a-b path P in G, and $x \in \mathbb{R}_+^{E(P)}$ with $x_e \in \{l_e, u_e\}$ for all $e \in E(P)$, we use $\textsc{Path}(P, x)$ to denote the unique a-b path Q in H such that for every $e \in E$,

- $e_l \in E(Q)$ if and only if $e \in E(P)$ and $x_e = l_e$;
- $e_u \in E(Q)$ if and only if $e \in E(P)$ and $x_e = u_e$.

Conversely, given an a-b path Q in H, we use $\textsc{Pair}(Q)$ to denote the unique pair (P, x) such that P is an a-b path in G, and $x \in \mathbb{R}_+^{E(P)}$ for which the above two necessary and sufficient conditions hold, i.e., $\textsc{Path}(P, x) = Q$. So

$$Q = \textsc{Path}(P, x) \text{ if and only if } (P, x) = \textsc{Pair}(Q).$$

Moreover the construction from (P, x) to $Q = \textsc{Path}(P, x)$ (resp. from Q to $(P, x) = \textsc{Pair}(Q)$) can be completed in $O(n)$ time such that the following holds.

Lemma 2. *If* $Q = \textsc{Path}(P, x)$ *, or equivalently* $(P, x) = \textsc{Pair}(Q)$*, then* $c(Q) = \sum_{e \in E(P)} \frac{u_e - x_e}{u_e - f_e}$ *and* $t(Q) = \sum_{e \in E(P)} x_e$*.*

$O(mn)$ time approximation. For convenience, we define the *Discrete Minimum Risk-sum Path problem* (on G), which has the same description as the minimum risk-sum path problem except that the requirement $x_e \in [l_e, u_e]$ over there is replaced with $x_e \in \{l_e, u_e\}$. Clearly, a solution to the discrete minimum risk-sum path problem is also a solution to the minimum risk-sum path problem, though the reverse is not necessarily true.

Algorithm 1. *Input:* directed graph $G = (V, E)$; $l, u \in \mathbb{Z}_+^E$; $K \in \mathbb{Z}_+$. *Output:* an optimal solution (P, x) to the discrete minimum risk-sum path problem.

1. Construct $H = (V, E'), c \in \{0, 1\}^{E'}, t \in \mathbb{Z}_+^{E'}$ based on G, l, u.
2. $T \leftarrow K$
3. Apply Algorithm B to find an optimal path Q to (CSP_T) on H with parameters c, t, T
4. Output $(P, x) \leftarrow \textsc{Pair}(Q)$

Theorem 2. *In $O(mn)$ time, Algorithm 1 outputs an optimal solution (P, x) to the discrete minimum risk-sum path problem, which is an approximate solution to the minimum risk-sum problem with risk-sum $\tau(P, x)$ at most one more than the optimal risk-sum for the minimum risk-sum path problem.*

Proof. From our constructions and Lemma 2, it is not hard to see that Algorithm 1 solves the discrete minimum risk-sum path problem optimally. The time complexity $O(mn)$ of Algorithm 1 is guaranteed by Theorem 1(ii), since $|E'| = 2m$, and $|V'| = n$, $c \in \{0, 1\}^{E'}$ imply $OPT \leq n$ for the (CSP_T) on H under consideration.

To verify the quality of the solution (P, x) output by Algorithm 1 from a viewpoint of the minimum risk-sum path problem, let us consider a simple optimal solution (P^*, x^*), as stated in Lemma 1, for which we have some $f \in E(P^*)$ and $x_e \in \{l_e, u_e\}$ for all $e \in E(P^*) \setminus \{f\}$. A solution (P^*, x') to the discrete risk-sum path problem on G can be defined by $x'_f := l_f$ and $x'_e = x_e$ for all $e \in E(P^*) \setminus \{f\}$. From the optimality of (P, x) to the discrete minimum risk-sum path problem, we deduce that $\tau(P, x) \leq \tau(P^*, x') \leq \tau(P^*, x^*) + 1$, proving the theorem. □

3.2 An Efficient Exact Algorithm

Next we focus on the minimum risk-sum path problem and on finding a simple optimal solution to it. The basic idea behind our algorithm is testing all m arcs

to find an arc f such that f and some simple optimal solution (P^*, x^*) satisfy Lemma 1.

As is customary, for vector/function $x \in \mathbb{R}_+^J$ and set $J' \subseteq J$, the restriction of x on J' is written as $x|_{J'}$; for arc (edge) e in a graph G', the deletion of e from G' results in a graph $G' \backslash e$. In the following pseudo-code description, the set Sol holds feasible solutions to the minimum risk-sum path problem found by the algorithm.

Algorithm 2. *Input: directed graph* $G = (V, E)$; $l, u \in \mathbb{Z}_+^E$; $K \in \mathbb{Z}_+$. *Output:* an optimal solution (P, x) to the minimum risk-sum problem on G.

1. $Sol \leftarrow \emptyset$
2. **while** G contains an arc e that does not belong to any s-d path in G **do**
3. $\quad G \leftarrow G \backslash e$
4. **end-while**
5. **for** every $f = (a, b) \in E$ **do**
6. $\quad$ Construct $H = (V, E'), c \in \{0, 1\}^{E'}, t \in \mathbb{Z}_+^{E'}$ based on G, l, u
7. $\quad \mathcal{Q}_1 \leftarrow$ the set of s-a paths in H, $\mathcal{Q}_2 \leftarrow$ the set of b-d paths in H
8. $\quad$ **for** $i = 1, 2$ **do**
9. $\quad\quad$ Apply Algorithm A to the constrained shortest path problem: $\min t(Q)$ s.t. $c(Q) \leq n$ and $Q \in \mathcal{Q}_i$, and find, for every $n_i = 0, \ldots, n$, an optimal solution (path) Q'_{n_i} to (CSP_{n_i}): $\min t(Q)$ s.t. $c(Q) \leq n_i$ and $Q \in \mathcal{Q}_i$
10. $\quad$ **end-for**
11. $\quad$ **for** every pair of integers n_1, n_2 with $0 \leq n_1, n_2 \leq n$ **do**
12. $\quad\quad$ **if** Q'_{n_1} and Q'_{n_2} have some common vertex
13. $\quad\quad\quad$ **then** $Q' \leftarrow$ an s-d path in $Q'_{n_1} \cup Q'_{n_2}$, $(P', x') \leftarrow \textsc{Pair}(Q')$
14. $\quad\quad\quad$ **else** $(P'_i, x'_i) \leftarrow \textsc{Pair}(Q'_{n_i})$, $i = 1, 2$
15. $\quad\quad\quad\quad P' \leftarrow P'_1 \cup \{f\} \cup P'_2$
16. $\quad\quad\quad\quad x'|_{E(P_i)} \leftarrow x'_i$, $i = 1, 2$
17. $\quad\quad\quad\quad x'_f \leftarrow \max\{l_f, \min\{u_f, |K - t(Q'_1) - t(Q'_2)|\}\}$
18. $\quad\quad$ **if** $\sum_{e \in E(P')} x'_e \leq K$ **then** $Sol \leftarrow Sol \cup \{(P', x')\}$
19. $\quad$ **end-for**
20. **end-for**
21. Take $(P, x) \in Sol$ with minimum $\tau(P, x)$
22. Output (P, x)

Before proceeding, let us make some elementary observations. The while-loop (Step 2 – 4) in Algorithm 2 and the constructions of H, c, t guarantee that, for every $f = (a, b)$ considered in Steps 5 – 20, H contains an s-a path Q_1 and a b-d path Q_2 with $c(Q_1) = c(Q_2) = 0$, implying the validity of Step 9. Moreover, by Theorem 1(i), the running time of Step 9 is $O(mn)$.

Suppose Q is an a-b path in a graph (directed or undirected), and a' is a vertex on Q, we use $Q[a, a']$ (resp. $Q[a', b]$) to denote the subpath of Q from a to a' (resp. from a' to b).

Lemma 3. *Let Q_1 be an s-a path and Q_2 be a b-d path in a graph having some common vertex. Then $Q_1 \cup Q_2$ contains an s-d path.*

Proof. Let a' be the common vertex of Q_1 and Q_2 first encountered when Q_1 is traversed from s to a. It is easily checked that $Q_1[s, a']$ and $Q_2[a', d]$ have exactly one common vertex, i.e., a', and that $Q_1[s, a'] \cup Q_2[a', d]$ is an s-d path contained in $Q_1 \cup Q_2$. □

From the last lemma, we see that Steps 12 – 17 of Algorithm 2 are valid, and produce in $O(n)$ time an s-d path P' in G, and real vector $x' \in \mathbb{R}_+^{E(P')}$ with $x'_e \in [l_e, u_e]$ for all $e \in E(P')$. In turn, it follows from Step 18 that any member in Sol is a feasible solution to the minimum risk-sum path problem on G.

Theorem 3. *Algorithm 2 solves the minimum risk-sum path problem optimally in $O(mn^3)$ time.*

Proof. Let (P^*, x^*) be a simple optimal solution to the minimum risk-sum path problem guaranteed by Lemma 1. By the above observations, it suffices to show that the final Sol contains an element (P', x') with $\tau(P', x') \le \tau(P^*, x^*)$. To this end,

(1) let $f = (a, b) \in E(P^*)$ be such that $x^*_e \in \{l_e, u_e\}$ for all $e \in E(P^*) \setminus \{f\}$.

Recall from Lemma 2 that $Q_1^* = \textsc{Path}(P^*[s, a], x^*|_{E(P^*[s,a])})$ is an s-a path in H, and $Q_2^* = \textsc{Path}(P^*[b, d], x^*|_{E(P^*[b,d])})$ is a b-d path in H such that

(2) $c(Q_1^*) = \sum_{e \in E(P^*[s,a])} \frac{u_e - x^*_e}{u_e - l_e}$, $t(Q_1^*) = \sum_{e \in E(P^*[s,a])} x^*_e$; and
$c(Q_2^*) = \sum_{e \in E(P^*[b,d])} \frac{u_e - x^*_e}{u_e - l_e}$, $t(Q_2^*) = \sum_{e \in E(P^*[b,d])} x^*_e$.

Notice from $c \in \{0, 1\}^{E'}$ that $c(Q_i^*) \le n$ for $i = 1, 2$. In the implementation of Algorithm 2, at some time, Steps 5 – 20 consider the $f = (a, b) \in E(P^*)$ as in (1), and Steps 11 – 19 consider $n_i = c(Q_i^*)$, $i = 1, 2$. Since Q_i^* is a feasible solution to the constrained shortest path problem (CSP_{n_i}), $i = 1, 2$, defined in Step 9, we have $t(Q'_{n_i}) \le t(Q_i^*)$ for $i = 1, 2$. It follows from (2) that

(3) $t(Q'_{n_1}) + t(Q'_{n_2}) \le t(Q_1^*) + t(Q_2^*) = \sum_{e \in E(P^*) \setminus \{f\}} x^*_e \le K - x^*_f$, in particular, $K - t(Q'_{n_1}) - t(Q'_{n_2}) \ge x^*_f \ge l_f$; and $c(Q'_{n_1}) + c(Q'_{n_2}) \le n_1 + n_2 = c(Q_1^*) + c(Q_2^*) = \sum_{e \in E(P^*) \setminus \{f\}} \frac{u_e - x^*_e}{u_e - l_e}$.

Further to the observation made before the theorem, the combination of Lemma 2 and (3) shows that

(4) the (P', x') produced by Steps 12 – 17 is an optimal solution to the minimum risk-sum path problem.

If Q'_{n_1} and Q'_{n_2} have some common vertex, then $\sum_{e \in E(P')} x'_e = t(Q') \le t(Q'_{n_1}) + t(Q'_{n_2}) \le K$ and $\tau(P', x') = c(Q') \le c(Q'_{n_1}) + c(Q'_{n_2}) \le \sum_{e \in E(P^*) \setminus \{f\}} \frac{u_e - x_e}{u_e - l_e} \le \tau(P^*, x^*)$; else $x^*_f \le x'_f = \min\{K - t(Q'_{n_1}) - t(Q'_{n_2}), u_f\} \le K - t(Q'_{n_1}) - t(Q'_{n_2})$,

$\sum_{e\in E(P')} x'_e = (\sum_{e\in E(P'_1)} x'_{1e}) + x'_f + (\sum_{e\in E(P'_2)} x'_{2e}) = t(Q'_{n_1}) + x'_f + t(Q'_{n_2}) \le K$, and $\tau(P', x') = c(Q'_{n_1}) + c(Q'_{n_2}) + \frac{u_f - x'_f}{u_f - l_f} \le \left(\sum_{e\in E(P^*)\setminus\{f\}} \frac{u_e - x^*_e}{u_e - l_e}\right) + \frac{u_f - x^*_f}{u_f - l_f} = \tau(P^*, x^*)$. Thus (4) holds.

By (4), we see that Step 18 puts (P', x') into Sol. The theorem is proved. □

4 Extension

In this section, we extend our results to more general settings of the minimum risk-sum path problem.

Firstly, to deal with the case in which $l_e = u_e$ for some e, we make the notational convention that $\frac{0}{0} = 0$ (i.e., $x_e = u_e = l_e \Leftrightarrow \frac{u_e - x_e}{u_e - l_e} = \frac{0}{0} = 0$) in the expression of the risk-sum of a solution to the minimum risk-sum path problem.

Secondly, the bounds $K \in \mathbb{R}_+$, l, $u \in \mathbb{R}^E_+$ may not be integral. It is easy to reduce the problem to an integral case by multiplying K and all l_e, u_e $(e \in E)$ with an appropriate integer M, and it is routine to check that the multiplication has effect on neither the time complexity of our algorithms nor the quality of the solutions output.

Finally, the underlying graph G may be undirected. The standard technique is replacing each edge e in G with two opposite arcs with the same ends as e, and reducing the problem to the directed case. In fact, the more flexibility of undirected graphs (than that of directed graphs) provides us the following Algorithm 3 faster than Algorithm 2, which runs in $O(m^2 n)$ time.

Algorithm 3. *Input:* undirected graph $G = (V, E)$; $l, u \in \mathbb{Z}^E_+$; $K \in \mathbb{Z}_+$. *Output:* an optimal solution (P, x) to the minimum risk-sum path problem on G.

1. $Sol \leftarrow \emptyset$
2. **for** every $f \in E$ **do**
3. Construct[2] $H = (V, E'), c \in \{0,1\}^{E'}, t \in \mathbb{Z}^{E'}_+$ based on G, l, u
4. Obtain undirected graph H_0 from H by contracting f_l, f_u into a vertex[3]
5. Apply Algorithm A to the constrained shortest path problem on H_0 with parameters $t|_{E'\setminus\{f_l,f_u\}}, c|_{E'\setminus\{f_l,f_u\}}, n$, and find for every $n_0 = 0, \ldots, n$ an optimal solution (path) Q'_0 to (CSP_{n_0}):
   $\min t(Q_0)$ s.t. $c(Q_0) \le n_0$ and Q_0 is an s-d path in H_0
6. **for** $n_0 = 0$ **to** n **do**
7. $Q' \leftarrow$ an s-d path in H containing $E(Q'_0)$ and avoiding f_u[4]
8. **if** $f_l \notin E(Q')$ **then** $(P', x') \leftarrow \textsc{Pair}(Q')$
9. **else** let Q_1 and Q_2 be the two paths whose union is $Q'\setminus f_l$

[2] The construction for undirected graphs is essentially the same as that for directed graphs with "edge" or "edges" in place of "arc" or "arcs".

[3] This vertex in H_0 is named s (resp d) if s (resp. d) is an end of f. Possibly, $s = d$ in H_0 while $s \ne d$ in H.

[4] Note that $E(Q') - E(Q'_0) \in \{\emptyset, \{f_l\}\}$, and $E(Q') = E(Q'_0)$ if and only if $f_l \notin E(Q')$.

10. $(P'_i, x'_i) \leftarrow \textsc{Pair}(Q_i)$, $i = 1, 2$
11. $P' \leftarrow P'_1 \cup \{f\} \cup P'_2$
12. $x'|_{E(P_i)} \leftarrow x'_i$, $i = 1, 2$
13. $x'_f \leftarrow \max\{l_f, \min\{u_f, |K - t(Q'_1) - t(Q'_2)|\}\}$
14. **if** $\sum_{e \in E(P')} x'_e \leq K$ **then** $Sol \leftarrow Sol \cup \{(P', x')\}$
15. **end-for**
16. **end-for**
17. Take $(P, x) \in Sol$ with minimum $\tau(P, x)$
18. Output (P, x)

The correctness of Algorithm 3 can be proved in a way similar to that in which Theorem 3 is proved. Again we take an simple optimal solution (P^*, x^*) and consider edge $f \in E(P^*)$ with ends a and b such that $x_e \in \{l_e, u_e\}$ for all $e \in E(P^*) \setminus \{f\}$. It is easy to see that when this f is considered by Algorithm 3 (Step 2 – 16), graph H_0 contains an s-d path Q with $E(Q) = E(\textsc{Path}(P^*, x'')) \setminus \{f_l\}$ (where $x'' \in \mathbb{R}_+^{E(P^*)}$ satisfies $x''_e \in \{l_e, u_e\}$ for all $e \in E(P^*)$ by setting $x''|_{E(P^*)\setminus\{f\}} := x^*|_{E(P^*)\setminus\{f\}}$, $x''_f := l_f$), $c(Q) = \sum_{e \in E(P^*)\setminus\{f\}} \frac{u_e - x^*_e}{u_e - l_e}$ and $t(Q) = \sum_{e \in E(P^*)\setminus\{f\}} x^*_e$; and furthermore, when the internal for-loop (Step 6 – 15) considers $n_0 = c(Q)$, the path Q is a feasible solution to (CSP_{n_0}). Hence we have $c(Q'_0) \leq c(Q)$ and $t(Q'_0) \leq t(Q)$. Using similar argument to that for proving (4) in the proof of Theorem 3, we deduce that (P', x') produced by Step 8 –13 of Algorithm 3 is a feasible solution to the minimum risk-sum problem with risk-sum $\tau(P', x') \leq \tau(P^*, x^*)$, and therefore is optimal. Thus Algorithm 3 solves optimally the minimum risk-sum path problems whose networks can be represented by undirected graphs.

5 Conclusions

In this paper we have proposed efficient algorithms for the minimum risk-sum path problem, which finds practical applications in route planning under uncertainty. The minimum risk-sum path problem bears some similarities to the robust shortest path problem (travel time interval vs. arc length interval) and to the stochastic path planning (risk vs. probability), however, as shown in this paper, has salient and essential differences from problems in these robust or stochastic settings, not only in solution structure (path with assignment vs. path without assignment), but also in complexity status (polynomial solvability vs. NP-hardness).

On the other hand, surprisingly, our simulation shows that for some appropriate time bounds K, the solutions of the minimum risk-sum path problem are quite similar to those of the robust shortest path problem [10]. This suggests future investigation on the reasonable intuitive/theoretical explanation of this phenomenon.

References

1. Fan, Y., Kalaba, R., Moore, J.: Arriving on time. Journal of Optimization Theory and Applications 127(3), 497–513 (2005)
2. Frank, H.: Shortest paths in probabilistic graphs. Operations Research 17, 583–599 (1969)
3. Fu, L., Rilett, L.R.: Expected shortest paths in dynamic and stochastic traffic network. Transportation Research 32, 499–512 (1998)
4. Gao, S., Chabini, I.: Optimal routing policy problems in stochasic time-dependent networks. Transportation Research Part B: Methodological 40, 93–122 (2006)
5. Hall, R.W.: The fastest path through a network with random time-dependent travel times. Transportation Science 20, 182–188 (1986)
6. Handler, G., Zang, I.: A dual algorithm for the constrained shortest path problem. Networks 10, 293–310 (1980)
7. Hassin, R.: Approximation schemes for the restricted shortest path problem. Mathematics of Operations Research 17, 36–42 (1992)
8. Henig, M.: The shortest path problem with two objective functions. European Journal of Opertional Research 25, 281–291 (1985)
9. Joksch, H.K.: The shortest route problem with constraints. Journal of Mathematical Analysis and Applications 14, 191–197 (1966)
10. Karaşan, O.E., Pinar, M.Ç., Yaman, H.: The robust shortest path problem with interval data. Computers & Operations Research (to appear)
11. Kouvelis, P., Yu, G.: Roubust Discrete Optimization and its Applications. Kluwer Academic Publishers, Boston (1997)
12. Marathe, M.V., Ravi, R., Sundaram, R., Ravi, S.S., Rosenkrantz, D.J., Hunt III, H.B.: Bicriteria network design problems. Journal of Algorithms 28, 142–171 (1998)
13. Miller-Hook, E.D., Mahmassani, H.S.: Least expected time paths in stochastic, time-varying transportation networks. Transportation Science 34, 198–215 (2000)
14. Montemanni, R., Gambardella, L.M., Donati, A.V.: A branch and bound algorithm for the robust shortest path problem with interval data. Operation Research Letters 32, 225–232 (2004)
15. Waller, S.T., Ziliaskopoulos, A.K.: On the online shortest path problem with limited arc cost dependencies. Networks 40(4), 216–227 (2002)
16. Yu, G., Yang, J.: On the robust shortest path problem. Computers & Operations Research 25, 457–468 (1998)
17. Zieliński, P.: The computational complexity of the relative robust shortest path problem with interval data. European Journal of Operational Research 158, 570–576 (2004)

Constrained Cycle Covers in Halin Graphs

Yueping Li

Department of Computer Science, Sun Yat-sen University
Guangzhou 510275, P.R. China
leeyueping@gmail.com

Abstract. This paper deals with constrained cycle cover problem in Halin graphs. A (constrained) cycle cover of a graph is a set of cycles such that every (selected) vertex is in at least one cycle. This problem arises in the design fiber-optic telecommunication networks which employ a set of rings covering the network. Besides two types of minimum weight cycle cover problem, we also settle the problem of covering Halin graphs with an optimal 2-edge-connected subgraph. Linear time algorithms are given for all of the problems.

Keywords: network, Halin graph, cycle cover, algorithm.

1 Introduction and Terminology

Telecommunication networks may be represented as a graph where the edges stand for the links between vertices that represent users (servers). An essential factor in the design of such a network is its *survivability*– its robustness to continue communicating in the event that one or more of its links are failed. For this purpose, the network must be built so that there are at least two disjoint routes between any pair of vertices. In this context, given a physical biconnected network, we wish to devise a logical network in the form of a set of cycles covering it. The problem is *cycle cover problem* in the literature [5].

In this paper, we discuss the *constrained cycle cover problem*, which is to find a minimum cost cycle cover for a specified vertex set such that the cycles are edge-disjoint. A cycle is said to cover the vertices it contains. Halin graphs are nontrivial generalization of tree and ring networks [8]. Since the topology of fiber-optic network between large servers (switches) is usually similar to a tree and planar, we focus on the problem in Halin graphs. For the terminology and notation not defined in this paper, reader can refer to [2].

A Halin graph is constructed as follows: start with a tree T in which each nonleaf has degree at least 3. Embed the tree in a plane and then connect all the leaves of T with a cycle C such that the resulting graph $H = T \cup C$ is planar. Suppose T has at least two nonleaves. Let w be a nonleaf of T which is adjacent to only one other nonleaf of T. Then the set of leaves of T adjacent to w, which we denote by $C(w)$, comprises a consecutive subsequence of the cycle C. The subgraph of H induced by $\{w\} \cup C(w)$ is said to be a *fan* and the vertex w is the *centre* of the fan. In Fig.1. the black vertices are the centres of the fans which are indicated by dotted lines.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 186–197, 2007.

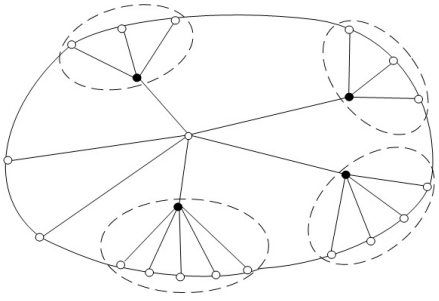


Fig. 1. A Halin graph

A graph G is called a *wheel* if G consists of a cycle every vertex of which is joined to a single common vertex by an edge.

Let F be a fan of Halin graph $H = T \cup C$, let $H \times F$ denote the graph obtained from H by shrinking F to form a new "pseudo-vertex" u_F; that is, $V(H \times F) = \{u_F\} \cup \{V(H) \backslash V(F)\}$ and $E(H \times F)$ are defined as follows:

1. An edge with both ends in F is deleted;
2. An edge with both ends in $H - F$ remains unchanged;
3. An edge with one end-vertex in $H - F$ and the other in F now joins the incident vertex of $H - F$ and the pseudo-vertex u_F.

This contraction was introduced by Cornuejols, Naddef and Pulleyblank [3]. They gave a linear time algorithm for *travelling salesman problem* (TSP) in Halin graphs. It applies "Shrinking Fan" operation recursively until the Halin graph is reduced to a wheel in which TSP can be easily solved. In the light of solving the steiner tree problem in Halin graphs by Winter [8], we also employ this mechanism but the calculation when shrinking a fan and the strategy to handle the final reduced wheel are quite different.

Denote the closure of H under the operation of "Shrinking Fan" by H^*. For a fan F of a Halin graph H, the three edges connecting $V(F)$ to $V(H-F)$ compose a 3-edge cutset of H, denoted by $EC_3(F)$. When F' is a fan in $H' \in H^*$, let $EC_{ex}(F')$ be the corresponding 3-edge cutset in H with respect to $EC_3(F')$. Let F_{ex} be the subgraph of H which F is fully restored to;that is, F_{ex} is the component of $H - EC_{ex}(F)$ in which F lies.

2 Preliminary Results

Lemma 1. *A Halin graph is minimally 3-edge-connected.*

Lemma 2 (Cornuejols[3]). *A Halin graph $H = T \cup C$ which is not a wheel has at least two fans.*

Lemma 3 (Cornuejols[3]). *If F is a fan in a Halin graph H, then $H \times F$ is a Halin graph.*

3 Our Results

Theorem 1. *Let H be a Halin graph and F be a fan in H. For every cycle C^* of H, $|E(C^*) \cap EC_3(F)| = 2$ or $E(C^*) \cap EC_3(F) = \emptyset$.*

Proof. Immediate.

Corollary 1. *Let H be a Halin graph and F' be a fan in $H' \in H^*$. For every cycle C^* of H, we have $|E(C^*) \cap EC_{ex}(F')| = 2$ or $E(C^*) \cap EC_{ex}(F') = \emptyset$. Furthermore, when $E(C^*) \cap EC_{ex}(F') = \emptyset$, if $E(C^*) \cap E(F'_{ex}) \neq \emptyset$, then $E(C^*) \subseteq E(F'_{ex})$.*

Theorem 2. *Let $H = T \cup C$ be a Halin graph. If C^* and C^{**} in H are edge-disjoint cycles which are connected by vertices, then $|V(C^*) \cap V(C^{**})| = 1$ and the intersecttion vertex belongs to $V(T) \setminus V(C)$.*

Proof. First, we show $|V(C^*) \cap V(C^{**})| \leq 1$. Suppose it does not hold and $|V(C^*) \cap V(C^{**})| \supseteq \{u_1, u_2\}$ for $u_1, u_2 \in V(H)$. Then choose an edge e of any path from u_1 to u_2. Since Halin graphs are minimally 3-edge-connected by Lemma 1, there are edges f, g such that $\{e, f, g\}$ is a 3-edge cutset which separates u_1, u_2 into different components. Since C^* covers u_1 and u_2, it contains two edges of $\{e, f, g\}$. So does C^{**}. Thus, $E(C^*) \cap E(C^{**}) \neq \emptyset$, a contradiction!

Secondly, since for each vertex of C, its degree is 3, it cannot be the intersection of two edge-disjoint cycles. Hence, $\{V(C^*) \cap V(C^{**})\} \subseteq \{V(T) \setminus V(C)\}$.

4 Covering Vertices with One Cycle

When one user (server) wants to communicate with selected users (servers), one good strategy is to build a logic cycle covering them. It not only allows data transport, but also uses of fast automatic protection in case of failure. This type of request can be viewed as covering selected vertices with one cycle. If all the vertices are chosen to be covered, the problem is reduced to TSP.

Let H be a Halin graph and $\alpha(u, v)$ be the cost of the edge $e = (u, v)$, $u, v \in V(H)$. For $e \in E(H)$, we abbreviate it by α_e. And let SV be the set of selected vertices and C^* be an optimal covering cycle. We examine the relation between C^* and a fan F in H. Let u_c be the centre of F and let u_1, u_2, ..., u_r for $r \geqslant 2$ be the vertices of F, which belong to C (in anti-clockwise order). Let $EC_3(F) = \{j, k, l\}$. See Fig. 2.

If the centre $u_c \in SV$, the situation is denoted by Case 1;the other is denoted by Case 2. Since only the choice of $EC_3(F)$ can affect the decision outside fan F for a cycle cover, we could enumerate all the combinations of them. And by Theorem 1, there are four choices. We define the functions as follows:

- $C_{kj} \equiv$ the minimum cost of $E(C^*) \cap E(F)$ when k, j are chosen;
- $C_{kl} \equiv$ the minimum cost of $E(C^*) \cap E(F)$ when k, l are chosen;
- $C_{jl} \equiv$ the minimum cost of $E(C^*) \cap E(F)$ when j, l are chosen;
- $C_{\emptyset} \equiv$ the minimum cost of C^* when none of k, j, l is chosen.

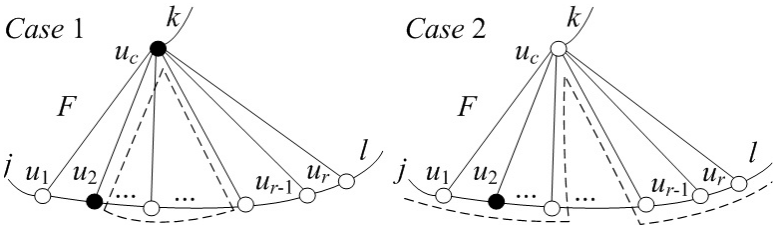


Fig. 2. One-cycle covers in fans

In addition, the structure which makes the minimum is also recorded in the function, if more than one obtains the minimum, select one arbitrarily. For original vertex u in H, let $C_{kj}(u) = C_{kl}(u) = C_{jl}(u) = 0$ and $C_{\emptyset}(u) = +\infty$. For abbreviation, we define $D_{a,b} \equiv \sum_{i=a}^{b} C_{jl}(u_i) + \sum_{i=a}^{b-1} \alpha(u_i, u_{i+1})$ for $1 \le a < b \le r$. And assume that $D_{a,b} = 0$ for $b \le a$ and $\alpha(u_0, u_1) = \alpha(u_r, u_{r+1}) = 0$. Let $m = \max\{\, i \,|\, u_i \in SV \cup \{u_1\}\}$ and $m' = \min\{\, i \,|\, u_i \in SV \cup \{u_r\}\}$.

When shrinking a fan, we need to calculate the functions. The formulas are presented as follows:

– Case 1: $u_c \in SV$.

$$C_{kj}(F) = \min\{D_{1,i-1} + \alpha(u_{i-1}, u_i) + C_{kj}(u_i) + \alpha(u_i, u_c) \,|\, m \le i \le r\} \tag{1a}$$

$$C_{kl}(F) = \min\left\{D_{i+1,r} + \alpha(u_i, u_{i+1}) + C_{kl}(u_i) + \alpha(u_i, u_c) \,|\, 1 \le i \le m'\right\} \tag{1b}$$

$$C^*_{jl}(F) = \min_{1 \le i < r} \left\{ \begin{array}{l} D_{1,i-1} + \alpha(u_{i-1}, u_i) + C_{kj}(u_i) + \alpha(u_i, u_c) \\ +\alpha(u_c, u_j) + C_{kl}(u_j) + \alpha(u_j, u_{j+1}) + D_{j+1,r} \\ \mid i \le j \le \min\{\, k \,|\, u_k \in SV \cup \{u_r\} \text{ for } i < k \le r\} \end{array} \right\} \tag{1c}$$

$$C_{jl}(F) = C^*_{jl}(F) \tag{1d}$$

$$C^*_{\emptyset}(F) = \min_{\substack{1 \le i \le m' \\ m \le j \le r}} \left\{ \begin{array}{l} \alpha(u_c, u_i) + C_{kl}(u_i) + \alpha(u_i, u_{i+1}) + D_{i+1,j-1} \\ +\alpha(u_{j-1}, u_j) + C_{kj}(u_j) + \alpha(u_c, u_j) \end{array} \right\} \tag{1e}$$

$$C^{**}_{\emptyset}(F) = \min\{C_{\emptyset}(u_i) \,|\, 1 \le i \le r\} \tag{1f}$$

$$C_{\emptyset}(F) = \min \begin{cases} C^*_{\emptyset}(F) & \text{if } SV \subseteq V(F), \\ C^{**}_{\emptyset}(F) & \text{if } SV = \{u_i\} \text{ where } 1 \le i \le r, \\ +\infty & \text{otherwise.} \end{cases} \tag{1g}$$

– Case 2: $u_c \notin SV$.

The subformulas of $C_{kj}(F)$, $C_{kl}(F)$, $C^*_{jl}(F)$, $C^*_{\emptyset}(F)$, $C^{**}_{\emptyset}(F)$ and $C_{\emptyset}(F)$ are the same as Case 1. But $C_{jl}(F)$ is defined as : $C_{jl}(F) = \min\{C^*_{jl}, D_{1,r}\}$.

Besides the computations above, if F contains any vertex in SV, then delete $V(F) \cap SV$ in SV and put u_F into SV. Then, one "Shrinking Fan" operation is done.

It can be easily concluded that the subformulas (1c) and (1e) could be calculated in $O(|r^2|)$ time. However, we give better implementations in $O(|r|)$ time.

We discuss subformula (1e), first. In this subformula, the whole covering cycle is in the fan F. Its sequence is always in the form u_c, u_i, u_{i+1}, ..., u_{j-1}, u_j, u_c where $1 \leq i \leq m'$ and $m \leq j \leq r$. It can be viewed as using exactly a "triangle" to cover the vertices $SV \cap V(F)$. An example is shown in dotted line in Case 1, Fig. 2.

To search an optimal covering triangle, we employ the dynamic programming strategy. It is necessary to introduce the substructure definition of *pseudo-fan*: a pseudo-fan $PF_{a,b}$ of F is the induced subgraph $F[x_c, x_a, x_{a+1}, \ldots, x_b]$ where $1 \leq a \leq b \leq r$.

Similarly, we use one function of pseudo-fan PF to store the minimum cost of the part of C^* which is in PF. Initially, we assign the values to the functions of $PF_{1,1}$ according to different types of triangle covers, which will be presented later. For a pseudo-fan $PF_{1,i}$ where $i \geq 2$, we assume that $SV \cap V(PF_{1,i})$ are already covered by C^*. We enumerate the choices of the edges (u_c, u_{i-1}), (u_{i-1}, u_i) and (u_c, u_i). The function $PF_{1,i}$ is defined to store the minimum cost for all the situations. We use a unique value to stand for each choice as follows:

- $PF_{1,i}(0)$: when neither of (u_{i-1}, u_i) and (u_c, u_i) is in C^*;
- $PF_{1,i}(2)$: when (u_{i-1}, u_i) is in C^* but (u_c, u_{i-1}) and (u_c, u_i) are not;
- $PF_{1,i}(1)$: when (u_c, u_i) is in C^* but (u_{i-1}, u_i) is not . It can be concluded that (u_i, u_{i+1}) (if exists) must be in C^*;
- $PF_{1,i}(-1)$: when (u_{i-1}, u_i) and (u_c, u_i) are in C^*. It can be concluded that (u_i, u_{i+1}) (if exists) cannot be in C^*;

Figure 3 displays the configurations of each choice. The wavy lines stand for the edges in C^*.

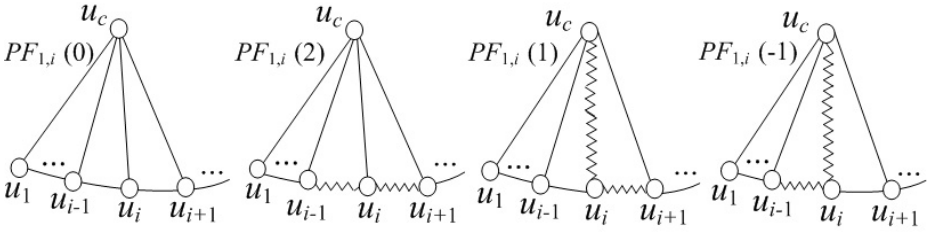


Fig. 3. Configurations

To solve subformula (1e), we define the function $PF_{1,1}$ in formula (2). Moveover, the deduction from $PF_{1,i}$ to $PF_{1,i+1}$ where $1 \leq i \leq r$ is given in formula (3).

$$\begin{aligned} PF_{1,1}(0) &= \begin{cases} 0 & \text{if } u_1 \notin SV, \\ +\infty & \text{otherwise.} \end{cases} & PF_{1,1}(-1) &= +\infty \\ PF_{1,1}(1) &= \alpha(u_1, u_c) + C_{kl}(u_1) + \alpha(u_1, u_2) & PF_{1,1}(2) &= +\infty \end{aligned} \tag{2}$$

$$
\begin{aligned}
PF_{1,i}(0) &= \begin{cases} PF_{1,i-1}(0) & \text{if } u_i \notin SV, \\ +\infty & \text{otherwise.} \end{cases} \\
PF^*_{1,i}(0) &= \begin{cases} \min\{PF^*_{1,i-1}(0), PF_{1,i-1}(-1)\} & \text{if } u_i \notin SV, \\ +\infty & \text{otherwise.} \end{cases} \\
PF_{1,i}(-1) &= \min\{PF_{1,i-1}(1), PF_{1,i-1}(2)\} + C_{kj}(u_i) + \alpha(u_i, u_c) \\
PF_{1,i}(1) &= PF_{1,i}(0) + C_{kl}(u_i) + \alpha(u_c, u_i) + \alpha(u_i, u_{i+1}) \\
PF_{1,i}(2) &= \min\{PF_{1,i-1}(1), PF_{1,i-1}(2)\} + C_{jl}(u_i) + \alpha(u_i, u_{i+1})
\end{aligned} \tag{3}
$$

The principle of formulas (2) and (3) is that one $PF_{1,i}(1)$ structure must be matched to a $PF_{1,j}(-1)$ structure where $1 \leq i < j \leq r$. Once the values of $PF_{1,r}$ are obtained, the value of $\min\{PF^*_{1,r}(0), PF_{1,r}(-1)\}$ is the solution to subformula $(1e)$.

Secondly, we focus on subformula $(1c)$. The sequence of $E(C^*) \cap E(F)$ is always in the form $u_1, u_2, \ldots, u_i, u_c, u_j, u_{j+1}, \ldots, u_r$ where $1 \leq i < j \leq r$ and $\{u_{i+1}, \ldots, u_{j-1}\} \cap SV = \emptyset$. It can be viewed as using two disjoint triangles to cover the vertices $\{u_1, u_r\} \cup \{SV \cap V(F)\}$ where the edges (u_c, u_1) and (u_c, u_r) are eliminated. An example is shown in dotted line in Case 2, Fig. 2. The formulas are presented as follows:

$$
\begin{aligned}
&PF_{1,1}(0) = +\infty \quad PF_{1,1}(-1) = \alpha(u_1, u_c) + C_{kj}(u_1) + \alpha(u_1, u_2) \\
&PF_{1,1}(1) = +\infty \quad PF_{1,1}(2) = C_{jl}(u_1) + \alpha(u_1, u_2) \\
&PF^*_{1,1}(2) = +\infty
\end{aligned} \tag{4}
$$

$$
\begin{aligned}
PF^*_{1,i}(0) &= \begin{cases} \min\{PF^*_{1,i-1}(0), PF_{1,i-1}(-1)\} & \text{if } u_i \notin SV, \\ +\infty & \text{otherwise.} \end{cases} \\
PF_{1,i}(-1) &= PF_{1,i-1}(2) + C_{kj}(u_i) + \alpha(u_i, u_c) \\
PF^*_{1,i}(1) &= PF^*_{1,i}(0) + C_{kl}(u_i) + \alpha(u_c, u_i) + \alpha(u_i, u_{i+1}) \\
PF_{1,i}(2) &= PF_{1,i-1}(2) + C_{jl}(u_i) + \alpha(u_i, u_{i+1}) \\
PF^*_{1,i}(2) &= \min\{PF^*_{1,i-1}(1), PF^*_{1,i-1}(2)\} + C_{jl}(u_i) + \alpha(u_i, u_{i+1})
\end{aligned} \tag{5}
$$

The value of $C^*_{jl}(F)$ equals $PF_{1,r}(-1) - \alpha(u_1, u_c) - \alpha(u_r, u_c)$.

If a subformula uses a function whose value is $+\infty$, that is, it indicates no such structure of cycle cover in the pseudo-fan, then the value of the subformula is set to be $+\infty$, immediately. For example, we calculate the function $PF^*_{1,2}(2)$ in formula (5), if $PF^*_{1,1}(2) = +\infty$, then $PF^*_{1,2}(2)$ is set to be $+\infty$.

5 Main Procedure of the Algorithm

Input: Given a Halin graph $H = T \cup C$ and the set SV as selected vertices.
Output: The cycle covering SV with minimum cost.

1) Choose a non-leaf vertex of T, denoted by v_{root},
   such that v_{root} is adjacent to a leaf of T, denoted by v_{leaf}.
2) Perform a postorder scan of T, for each fan F has been found do
3) If the centre of $F \neq v_{root}$ then

begin
shrink F to u_F;
calculate the values to the function of F and store into u_F ;
record the structure of F corresponding to each value.
end

Let H_w be the wheel we finally get and k be the edge joining v_{root} and v_{leaf}.

Let j, l be the two edges in C adjacent to v_{leaf} such that the direction of j, v_{leaf}, l is clockwise.

4) Let fan $F_w \equiv H_w - v_{leaf}$. Calculate the values to the function of F_w.
5) Store the structure of covering cycle C^* in F_w corresponding to each value.
6) Shrink the fan F_w to the pseudo-vertex v_w.
7) $Cost := \min\{C_{kj}(v_w)+\alpha_k+\alpha_j, C_{kl}(v_w)+\alpha_k+\alpha_l, C_{jl}(v_w)+\alpha_j+\alpha_l, C_{\emptyset}(v_w)\}$.
Let E^* be the edges in $\{k, j, l\}$ the choice of which makes the maximum
//If more than one get the maximum, choose one arbitrarily.
8) Mark E^* belonging to the covering cycle C^*.

9) While (there is pseudo-vertex in H) do
10) begin
Let v_F be a pseudo-vertex such that v_F is shrunk from the fan F.
11) According to the information stored in v_F,
we can obtain the corresponding part of C^* in F.
12) Restore v_F to the fan F .
13) Mark the edges of C^* in F.
14) end
15) All the marked edges compose the cycle cover C^* for output.

In the algorithm, the section 1) needs $O(1)$ time. The section 3) is the operation of shrinking fans in H. If a fan F contains $r+1$ vertices, then it is verified that the time of the shrinking operation is $O(r)$. Moreover, shrinking F reduces the number of vertices of the graph by r. Thus, the total time of the shrinking operation is $O(|V|)$. The time of the postorder scan without shrinking is bounded by $O(|V|)$. The sections 4) $\sim$ 8) need $O(1)$ time. The time of sections 9) $\sim$ 14) is the same as the section 3). Therefore, the total time for this algorithm is $O(|V|)$.

6 Covering Vertices with Disjoint Cycles

One variation of covering vertices using one cycle is covering them with edge-disjoint cycles which are connected by vertices. The logic network consists of covering cycles is more robust: if there are more than one links failed and they lie in different cycle, then each node of the network is still reachable.

By Theorem 2, the edge-disjoint cycles intersect in the inner of T. For a fan F of H^*, if it contains parts of two edge-disjoint vertex-connected cycles C^* and C^{**}, then the common vertex is the centre of F.

If $E(C^*) \cup E(C^{**}) \subseteq E(F)$, it could be viewed as using *disjoint* triangles to cover the selected vertices.

If $E(C^*) \cup E(C^{**}) \nsubseteq E(F)$, there is exactly one cycle contains two edges of $EC_3(F)$. Furthermore, the edges of other covering cycles must be in F. Then, suppose $E(C^*) \nsubseteq E(F)$, the edges j, k, l and the vertices $u_c, u_1, \ldots, u_r$ are defined the same as Section 4.

- If C^* uses $\{j, k\}$, the sequence of $E(C^*) \cap E(F)$ is always in the form u_1, u_2, $\ldots$, u_i, u_c where $1 \leq i \leq r$. It can be viewed as one triangle which contains u_1 where the edge (u_c, u_1) is eliminated.
- If C^* uses $\{k, l\}$, the situation is symmetric as above.
- If C^* uses $\{j, l\}$, the sequence is in the form u_1, u_2, $\ldots$, u_i, u_c, u_j, u_{j+1}, $\ldots$, u_r where $1 \leq i < j \leq r$. It can be viewed as two disjoint triangles which contain u_1, u_r respectively where the edges (u_c, u_1) and (u_c, u_r) are eliminated.

We also use the functions C_{jk}, C_{jl}, C_{kl}, $C_\emptyset$ and the formulas are similar with Section 4. The difference is that the calculation to the cost of covering cycles is using arbitrary number (not just one) disjoint triangles to cover $SV \cap V(F)$. The cost can be computed by the following formulas:

$$\begin{aligned} &PF_{1,1}(0) = \begin{cases} 0 & \text{if } u_1 \notin SV, \\ +\infty & \text{otherwise.} \end{cases} && PF_{1,1}(-1) = \alpha(u_1, u_c) + C_{kj}(u_1) \\ &PF_{1,1}(1) = \alpha(u_1, u_c) + C_{kl}(u_1) + \alpha(u_1, u_2) && PF_{1,1}(2) = +\infty \end{aligned} \tag{6}$$

$$\begin{aligned} PF_{1,i}(0) &= \begin{cases} \min\{PF_{1,i-1}(0), PF_{1,i-1}(-1)\} & \text{if } u_i \notin SV, \\ +\infty & \text{otherwise.} \end{cases} \\ PF_{1,i}(-1) &= \min\{PF_{1,i-1}(2), PF_{1,i-1}(1)\} + C_{kj}(u_i) + \alpha(u_i, u_c) \\ PF_{1,i}(1) &= PF_{1,i}(0) + C_{kl}(u_i) + \alpha(u_c, u_i) + \alpha(u_i, u_{i+1}) \\ PF_{1,i}(2) &= \min\{PF_{1,i-1}(1), PF_{1,i-1}(2)\} + C_{jl}(u_i) + \alpha(u_i, u_{i+1}) \end{aligned} \tag{7}$$

Formula (6) is the initialization of calculating $C_{kj}(F)$. For other functions, it needs to be changed. The value of $C_{kj}(F)$ is $\min\{PF_{1,r}(0), PF_{1,r}(-1)\} - \alpha(u_1, u_c)$ where SV is set to $\{u_1\} \cup \{SV \cap V(F)\}$, temporally. The calculation of C_{kl} is symmetric. The value of C_{jl} is $PF_{1,r}(-1) - \alpha(u_1, u_c) - \alpha(u_r, u_c)$ where SV is set to $\{u_1, u_r\} \cup \{SV \cap V(F)\}$. The value of $C_\emptyset$ is $\min\{PF_{1,r}(0), PF_{1,r}(-1)\}$ where SV is unchanged.

The main procedure of algorithm is the same as Section 5 except that the store of the covering cycles' structure should be included in the calculation of each $PF_{1,i}$ in every fan. It takes more $O(V)$ space. The previous one just stores the structure into the function of the fan. For instance, $C_{jk}(F)$ records the choice of i which obtains minimum.

7 Covering Vertices with an Optimal 2-Edge-Connected Subgraph

The functions and definitions mentioned in Section 4 are still used in this Section while each function value records the cost of the part in a fan with respect to the optimal 2-edge-connected subgraph. In addition, we use $C_{jkl}(F)$ to store the

minimum cost when j, k, l are in the subgraph. For an arbitrary fan F in H, the cases are simple and similar with previous ones.

- Case 1: none of j, k, l is used.
  This case is solved in Section 5.
- Case 2: j, k are used.
  This situation can be viewed as using arbitrary number edge-disjoint triangles to cover $\{u_1\} \cup \{SV \cap V(F)\}$. Just need to discount the edge (u_1, u_c) when calculating the cost.
- Case 3: k, l are used. Symmetrically.
- Case 4: j, l are used.
  Case 4.1: If there is an edge in $E(F) \cap E(C)$ not selected into the subgraph. This situation can be viewed as using arbitrary number edge-disjoint triangles to cover $\{u_1, u_r\} \cup \{SV \cap V(F)\}$. Just need to discount the edges (u_1, u_c) and (u_r, u_c) when calculating the cost.

  Case 4.2: There is no such edge in Case 4.1. Subformula (1c) is changed to be $C^*_{jl}(F) = \min_{1 \le i < j \le r}\{\alpha(u_c, u_i) + \alpha(u_c, u_j)\} + D_{1,r}$.
- Case 5: j, k, l are used.
  Case 5.1: If there is an edge in $E(F) \cap E(C)$ not selected into the subgraph, this case is the same as Case 4.1.

  Case 5.2: There is no such edge in Case 5.1. Choose the edge (u_c, u_i) of which the cost is minimum in $\{(u_c, u_1), \ldots, (u_c, u_r)\}$. If there is other edge (u_c, u_j) where $j \neq i$ selected, it can be removed. Hence, $C_{jkl} = D_{1,r} + \min_{1 \le i \le r}\{\alpha(u_c, u_i)\}$.

We proceed to handle the fans of H^*. For an arbitrary fan F in H^*, the cases are quite different from previous ones.

- Case 1: none of j, k, l is used.
  Case 1.1: If $SV = \{u_i\}$ for some $i \in \{1, \ldots, r\}$, then $C_\emptyset(F) = C_\emptyset(u_i)$.

  Case 1.2: Otherwise, $C_\emptyset(F)$ equals the cost of using arbitrary number triangles to cover $SV \cap V(F)$. The difference is that two adjacent triangles can have one common edge in $\{(u_c, u_1), \ldots, (u_c, u_r)\}$. We do not calculate the subformula (1e), (1f) and (1g).

  Instead, this type of coverage can be solved by means of the following formulas. The definitions of $PF_{1,1}(0), PF_{1,1}(1)$ and $PF_{1,1}(-1)$ are the same as formula (6). And we set $C_\emptyset(F) = \min\{PF_{1,r}(0), PF_{1,r}(-1)\}$.

$$\begin{aligned} PF_{1,i}(0) &= \begin{cases} \min\{PF_{1,i-1}(0), PF_{1,i-1}(-1)\} & \text{if } u_i \notin SV, \\ +\infty & \text{otherwise.} \end{cases} \\ PF_{1,i}(-1) &= PF_{1,i-1}(1) + C_{kj}(u_i) + \alpha(u_i, u_c) \\ PF_{1,i}(1) &= \min \begin{cases} PF_{1,i}(0) + C_{kl}(u_i) + \alpha(u_c, u_i) + \alpha(u_i, u_{i+1}) \\ PF_{1,i-1}(1) + \min\{C_{jl}(u_i), C_{jkl}(u_i) + \alpha(u_c, u_i)\} + \alpha(u_i, u_{i+1}) \end{cases} \end{aligned} \tag{8}$$

– Case 2: j, k are used.
We define $PF_{1,1}(0) = NULL$, $PF_{1,1}(1) = \min\{C_{jl}(u_1), C_{jlk}(u_1)\} + \alpha(u_1, u_2)$ and $PF_{1,1}(-1) = C_{kj}(u_1) + \alpha(u_1, u_2)$. Use formula (8) to calculate $PF_{1,i}$.
At last, $C_{jk}(F) = \min\{PF_{1,r}(0), PF_{1,r}(-1)\}$.

– Case 3: k, l are used. Symmetrically.

– Case 4: j, l are used.
Case 4.1: If u_c is not selected, subformula is set to be $C_{jl}(F) = D_{1,r}$.
Case 4.2: Otherwise, since the edge k is not selected, we should guarantee there are at least two edges incident with u_c. New technique is introduced as follows:

Let E_k be the edge set incident with u_c. Let function $PF_{1,i}$ store corresponding values when $E(PF_{1,i}) \cap E_k = \emptyset$. $PF^*_{1,i}$ and $PF^{**}_{1,i}$ are used when $|E(PF_{1,i}) \cap E_k| = 1$ and $|E(PF_{1,i}) \cap E_k| \geq 2$, respectively. Thus, $C_{jl}(F) = PF^{**}_{1,r}(1)$ according to the requirement. The formulas applied this technique are given below:

$$
\begin{aligned}
&PF_{1,i}(0) = +\infty, \forall i \in \{1..r\}, && PF_{1,i}(-1) = +\infty, \forall i \in \{1..r\},\\
&PF_{1,1}(1) = C_{jl}(u_1) + \alpha(u_1, u_2),\\
&PF^*_{1,1}(0) = +\infty, && PF^*_{1,1}(-1) = C_{kj}(u_1) + \alpha(u_c, u_1),\\
&PF^*_{1,1}(1) = \alpha(u_1, u_c) + \min\{C_{jkl}(u_1), C_{kl}(u_1)\} + \alpha(u_1, u_2),\\
&PF^{**}_{1,1}(0) = +\infty, && PF^{**}_{1,1}(-1) = +\infty,\\
&PF^{**}_{1,1}(1) = +\infty,
\end{aligned}
$$

$$
\begin{aligned}
PF^*_{1,i}(0) &= \begin{cases} \min\{PF^*_{1,i-1}(0), PF^*_{1,i-1}(-1)\} & \text{if } u_i \notin SV,\\ +\infty & \text{otherwise.} \end{cases}\\
PF^{**}_{1,i}(0) &= \begin{cases} \min\{PF^{**}_{1,i-1}(0), PF^{**}_{1,i-1}(-1)\} & \text{if } u_i \notin SV,\\ +\infty & \text{otherwise.} \end{cases}\\
PF_{1,i}(1) &= PF_{1,i-1}(1) + C_{jl}(u_i) + \alpha(u_i, u_{i+1})\\
PF^*_{1,i}(1) &= \min\left\{ \begin{array}{l} PF^*_{1,i}(1) + C_{jl}(u_i) + \alpha(u_i, u_{i+1}),\\ PF_{1,i-1}(1) + C_{jkl}(u_i) + \alpha(u_i, u_c) + \alpha(u_i, u_{i+1}) \end{array} \right\}\\
PF^{**}_{1,i}(1) &= \min\left\{ \begin{array}{l} PF^{**}_{1,i-1}(1) + \alpha(u_i, u_{i+1}) + \min\{C_{jl}(u_i), C_{jkl}(u_i) + \alpha(u_i, u_c)\},\\ PF^*_{1,i-1}(1) + C_{jkl}(u_i) + \alpha(u_i, u_c) + \alpha(u_i, u_{i+1}),\\ \min\{PF^*_{1,i-1}(0), PF^*_{1,i-1}(-1), PF^{**}_{1,i-1}(0), PF^{**}_{1,i-1}(-1)\}\\ \quad + C_{kl}(u_i) + \alpha(u_i, u_c) + \alpha(u_i, u_{i+1}) \end{array} \right\}\\
PF^*_{1,i}(-1) &= PF_{1,i-1}(1) + C_{kj}(u_i) + \alpha(u_i, u_c)\\
PF^{**}_{1,i}(-1) &= \min\{PF^*_{1,i-1}(1), PF^{**}_{1,i-1}(1)\} + C_{kj}(u_i) + \alpha(u_i, u_c)
\end{aligned}
\tag{9}
$$

– Case 5: j, k, l are used.
Use formula (9) to obtain $PF^*_{1,r}$ and $PF^{**}_{1,r}$. We have $C_{jkl}(F) = \min\{PF^*_{1,r}(1), PF^{**}_{1,r}(1)\}$ in this case. Note that if $C_{jkl}(F) = PF^{**}_{1,r}(1)$, it indicates that the edge k is removable with respect to fan F. If this edge is also removable in the fan of H^* which u_F lies in, the cost $C_{jl}(F) < C_{jkl}(F)$ will be smaller. So the optimal subgraph does not contain such removable edge k.

8 Example

An example for our algorithms is given in Figure 1. The arrows show the main procedure of shrinking fans. The graph marked with ① illustrates an optimal one-cycle cover which is indicated by dotted lines. The graph marked with ② shows an optimal cover with edge-disjoint cycles. And the one marked with ③ displays an optimal cover with a 2-edge-connected subgraph. The values to the functions of pseudo-vertices are presented in Table 1.

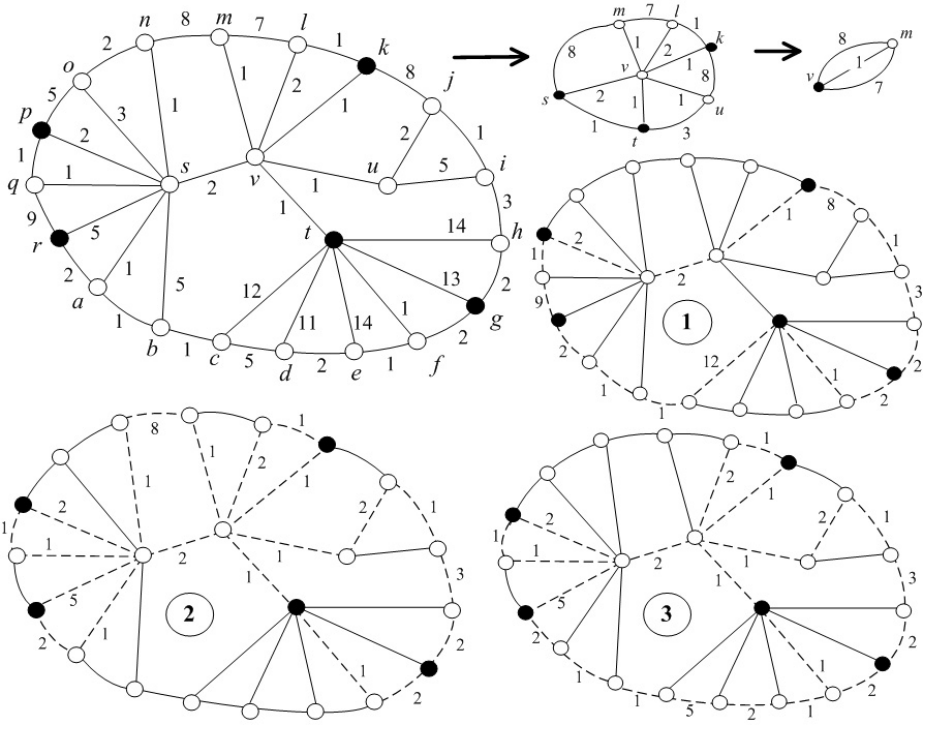


Fig. 4. Example

Table 1. Values to the functions of pseudo-vertices

	Covering with One Cycle				Covering with Cycles				Covering with subgraph				
	$C_\emptyset$	C_{kj}	C_{kl}	C_{jl}	$C_\emptyset$	C_{kj}	C_{jl}	C_{jl}	$C_\emptyset$	C_{kj}	C_{kl}	C_{jl}	C_{jkl}
s	15	20	15	16	12	13	12	13	12	13	12	13	13
t	16	23	5	17	16	23	5	17	16	23	5	17	13
u	0	3	2	1	0	3	2	1	0	3	2	1	3
v	48	47	48	43	43	32	40	30	40	32	38	30	30
Cost	<u>48</u>	56	56	58	43	<u>41</u>	48	45	<u>40</u>	41	46	45	46

9 Conclusions

We settle the problem of searching an optimal constrained cycle covers in Halin Graphs. We study two types of cycle covers and 2-edge-connected subgraph cover. The method is more applicable. For example, since the Synchronous Optical NETwork (SONET) standard limits the number of nodes on a ring [9]. We can add one more variable, which indicates the length of cycle, in the functions' definitions. If the parameter D is the maximum number of nodes allowed in a cycle, then each formula turns into D ones. The formula indicates length n is calculated from the ones standing for length $n-1$. The space and time complexity will grow to $O(D|V|)$.

Another extension is that our algorithm can be easily changed to solve the optimal one-path cover in Halin graphs. In practice, we usually establish the ring using two connections in both directions within one link. Hence, one path covers selected vertices is sufficient. Li, Lou and Lu [6] solved the optimal Hamiltonian paths in Halin graphs. Hamiltonian path can be viewed as one path covering all the vertices. We could solve the constrained problem which covers required vertices by means of the amended version of our algorithm.

The "Shrinking Fan" method works since Halin graphs have small treewidth, which was stated in [1]. Our algorithm might be considerable to tackle the constrained cycle cover problem in small treewidth graphs.

References

[1] Arnborg, S., Lagergren, J.: Easy problems for tree-decomposable graphs. Journal of Algorithms 12, 308–349 (1991)
[2] Bondy, J.A., Murty, U.S.R.: Graph theory with application. Macmillan, London (1976)
[3] Cornuejols, G., Naddef, D., Pulleyblank, W.: Halin graphs and the traveling salesman problem. Mathematical Programming 26, 287–294 (1983)
[4] Cornuejols, G., Naddef, D., Pulleyblank, W.: The traveling salesman problem in graphs with 3-edge cutsets. Journal of ACM 32, 383–410 (1985)
[5] Hochbaum, D.S., Olinick, E.V.: The bounded cycle-cover problem. INFORMS Journal on Computing 13, 104–119 (2001)
[6] Li, Y.P., Lou, D.J., Lu, Y.T.: Algorithms for the optimal hamiltonian path in Halin graphs. Ars Combinatoria (accepted)
[7] Pesant, G., Soriano, P.: An optimal strategy for the constrainted cycle cover problem. Annuals of Mathematics and Aritificial Intelligence 34, 313–325 (2002)
[8] Winter, P.: Steiner problem in halin graphs. Discrete Apply. Math. 17, 281–294 (1987)
[9] Wu, T.H.: Fiber network service survivability. Artech House, Norwood, MA (1992)

Optimal Semi-online Algorithms for Scheduling with Machine Activation Cost*

Shuguang Han[1,3], Yiwei Jiang[1,2,**], and Jueliang Hu[1]

[1] Faculty of Science, Zhejiang Sci-Tech University, Hangzhou 310018, China
[2] Key Laboratory of Advanced Textile Materials and Manufacturing Technology (Zhejiang Sci-Tech University), Ministry of Education, Hangzhou 310018, China
mathjyw@yahoo.com.cn
[3] Department of Mathematics, Zhejiang University, Hangzhou 310027, China

Abstract. We investigate the following two semi-on-line scheduling problems with machine activation cost. We are given two potential identical machines to non-preemptively process a sequence of independent jobs. Machines need to be activated before starting to process, and each machine activated incurs a fixed machine activation cost. No machines are initially activated, and when a job is revealed the algorithm has the option to activate new machines. The objective is to minimize the sum of the makespan and activation cost of machines. For the first semi-on-line problem with known the sum size of all jobs P in advance , we present an semi-on-line algorithm which is optimal for every $P > 0$. For the second problem with known the largest size of all jobs L in advance, we present an optimal semi-on-line algorithm for every $L > 0$.

1 Introduction

Problem statement. We consider the following semi-on-line scheduling problems. We are given a sequence J of independent jobs with positive processing times (sizes) $p_1, p_2, \ldots, p_n$, which must be non-preemptively scheduled on identical machines. We identify jobs with their sizes here. Jobs arrive one by one (*on-line over list*) and are to be scheduled irrevocably on machines as soon as they are given. We are given m potential machines which need to be activated before starting to process and the activation cost cannot be neglected. Initially there are no machines activated. As the machines are identical, by normalizing all job sizes and machine activation cost, we assume that the activation cost for each machine is 1 without loss of generality. The goal is to minimize the sum of the makespan and the total machine activation cost. In this paper, we deal with two semi-on-line problems on two potential machines (i.e., $m = 2$). One is to be known the total size of all jobs in advance, denoted by P, and the other is to be known the size of the largest job in advance, denoted by L. Using a

* Research supported by Natural Science Foundation of China(10671177), Natural Science Foundation of Zhejiang Province (Y605316), and Natural Science Foundation of Education Department of Zhejiang Province (20060578).

** Corresponding author.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 198–208, 2007.

three field notation, we denote these two problems by $P2|sum|C_{max} + m'$ and $P2|max|C_{max} + m'$, respectively.

Note that our semi-on-line problems are quite different from the classical semi-online parallel machine scheduling problems [3,6,9,11], where we typically have a fixed number m of machines, the scheduler makes no decision regarding the number of machines that are used to process jobs, and the provided machines can be activated without any cost. We still use *competitive ratio* to measure the performance of our semi-online algorithm. For a job sequence $\mathcal{J}$ and an algorithm A, let $c^A(\mathcal{J})$ (or shortly c^A) denote the makespan produced by A and let $c^*(\mathcal{J})$ (or shortly c^*) denote the optimal makespan in an off-line version. Then the competitive ratio of A is defined as the smallest number C such that for any $\mathcal{J}$, $c^A(\mathcal{J}) \leq Cc^*(\mathcal{J})$. An on-line problem has a *lower bound* ρ if no on-line deterministic algorithm has a competitive ratio smaller than ρ. An on-line algorithm is called *optimal* if its competitive ratio matches the lower bound.

Previous work. Imreh and Noga [7] first considered the problem with $m = +\infty$, that is, there are sufficient large number of identical machines to be activated. They presented an online $\frac{1+\sqrt{5}}{2} \approx 1.618$-competitive algorithm A_ρ while the lower bound was 4/3. Dósa and He [2] made an improvement by presenting an algorithm with a competitive ratio of $\frac{2\sqrt{6}+3}{5} \approx 1.5798$. He and Jiang [8] extended to consider preemptive online algorithms for *List Model* problem. In addition, Panwalker and Liman [10] proposed another off-line scheduling problem that there are $m = +\infty$ potential identical machines which can be activated, and the objective is to find an optimal schedule, the optimal number of machines, and the respective due dates to minimize the weighted sum of earliness, tardiness, and machine activation cost. It is remarkable that the good performance of the algorithms in [7,2,8] is based on that we are allowed to activate a large number of machines if needed.

But in some application, the number of the potential machines may not be sufficient large. Cao, Chen and Wan [1] considered a parallel machine scheduling problem where only finite machines are provided. The objective is to minimize the sum of the total weighted job tardiness penalties and the total machine activation cost. He, Han and Jiang [5] considered online algorithms for *List Model* problem with finite identical machines. The objective is to minimize the sum of the makespan and the total machine activation cost. Han, Jiang and Hu [4] considered such a scheduling problem on two uniform machines.

Our results. We consider two semi-online problems in this paper, i.e. $P2|sum|C_{\max} + m'$ and $P2|max|C_{\max} + m'$. For the first problem, we present an algorithm which is optimal for every $P > 0$. For the second problem, we also present an optimal algorithm for every $L > 0$.

The rest of the paper is organized as follows. Section 2 gives some basic notation. Section 3 presents the lower bound and an optimal algorithm for

$P2|sum|C_{\max}+m'$. Section 4 presents the lower bound and an optimal algorithm for $P2|max|C_{\max}+m'$. Finally, section 5 contains some remarks.

2 Preliminary

To simplify the presentation, the following notation and definitions are required in the remainder of the paper.

Denote by $p_j^{\max}=\max\{p_i|i=1,\cdots,j\}$, and L the largest size of all jobs, then $L=p_n^{\max}$. Denote by $P=\sum_{i=1}^{j}p_n$ and $P_j=\sum_{i=1}^{j}p_i$. Let M_1,M_2 be the two potential machines. We call *moment* j as the time right after the j-th job is scheduled. Let $s_{i,j}$ denote the current load of machine M_i at moment $j\geq 0$ in an algorithm A, $i=1,2$. Let C^A and C^* be the makespan yielded by A and in the optimal solution, respectively. And let m and m^* be the machine activation cost yielded by algorithm A and in the optimal solution, respectively. Then we have $c^A=C^A+m$ and $c^*=C^*+m^*$.

The following Theorem is easy to be obtained.

Theorem 1. *The optimal value for the two semi-on-line problems is at least* $\min\{1+P,2+\max\{\frac{P}{2},L\}\}$.

3 Problem $P2|sum|C_{\max}+m'$

In this section, we assume that the total size of all jobs P is known in advance. We first give the lower bound of the problem, then present an algorithm $H1$ which is optimal for any $P>0$.

3.1 Lower Bound

We present the lower bound of the problem below.

Theorem 2. *The competitive ratio of any semi-on-line algorithm A for problem* $P2|sum|C_{\max}+m'$ *is*

$$C\geq\begin{cases}1, & 0<P\leq 2,\\ \frac{2+2P}{4+P}, & 2<P\leq 3,\\ \frac{12+4P}{12+3P}, & P>3.\end{cases}$$

Proof. The result is obviously true for $0\leq P\leq 2$.

For $P>2$, the first two jobs $p_1=p_2=\frac{P}{6}$ arrive. If algorithm A activates only one machine to process them, then the last two jobs $p_3=p_4=\frac{P}{3}$ arrive. Clearly, by Theorem 1 and $P>2$, we have $c^*=2+\frac{P}{2}$. It is not hard to obtain that $c^A=\min\{1+P,2+\frac{2P}{3}\}$ and thus

$$C\geq\frac{c^A}{c^*}\geq\frac{\min\{1+P,2+\frac{2P}{3}\}}{2+\frac{P}{2}}=\begin{cases}\frac{2+2P}{4+P}, & 2<P\leq 3,\\ \frac{12+4P}{12+3P}, & P>3.\end{cases}$$

On the other hand, if algorithm A activates two machines and schedules the jobs p_1 and p_2 onto different machines, then the last two jobs $p_3 = \frac{P}{6}$ and $p_4 = \frac{P}{2}$ arrive. We also have $c^* = 2 + \frac{P}{2}$. It is clear that $c^A \geq 2 + \frac{2P}{3}$ and thus

$$C \geq \frac{c^A}{c^*} = \frac{2 + \frac{2P}{3}}{2 + \frac{P}{2}} = \frac{12 + 4P}{12 + 3P} \geq \begin{cases} \frac{2+2P}{4+P}, & 2 < P \leq 3, \\ \frac{12+4P}{12+3P}, & P > 3. \end{cases}$$ □

3.2 Optimal Algorithm $H1$

We present an optimal semi-on-line algorithm $H1$ for the problem. The algorithm activates only one machine if $P \leq 3$. If $P > 3$, three cases are considered according to the size of job p_t, where $t = \min\{j | \sum_{i=1}^{j} p_i > \frac{2P}{3}\}$.

Algorithm $H1$

1. If $P \leq 3$, activate machine M_1 to schedule all the jobs, Stop.
2. If $P \geq 3$, activate machine M_1 to schedule $p_1, p_2, \cdots, p_{t-1}$.
   - 2.1. If $P - 1 < p_t \leq P$, schedule p_t and all the remaining jobs onto machine M_1, Stop.
   - 2.2. If $\frac{P}{3} < p_t \leq P - 1$, activate machine M_2 to schedule p_t, and schedule all the remaining jobs onto machine M_1, Stop.
   - 2.3. If $p_t \leq \frac{P}{3}$, activate machine M_2 to schedule p_t and all the remaining jobs, Stop.

Lemma 1. *If* $P \leq 3$, *then* $\frac{c^{H1}}{c^*} \leq \begin{cases} 1, & 0 < P \leq 2, \\ \frac{2+2P}{4+P}, & 2 < P \leq 3. \end{cases}$

Proof. According to the rule of algorithm $H1$, only one machine M_1 is activated by the algorithm $H1$, then $c^{H1} = 1 + P$. If $P \leq 2$, by Theorem 1, we have $c^* \geq \min\{1 + P, 2 + \frac{P}{2}\} = 1 + P = c^{H1}$. If $2 < P \leq 3$, then we have $c^* \geq \min\{1 + P, 2 + \frac{P}{2}\} = 2 + \frac{P}{2}$ and thus $\frac{c^{H1}}{c^*} \leq \frac{1+P}{2+\frac{P}{2}} = \frac{2+2P}{4+P}$. □

Lemma 2. *If* $P \geq 3$, *then* $\frac{c^{H1}}{c^*} \leq \frac{12+4P}{12+3P}$.

Proof. If $P - 1 < p_t \leq P$, by the rule of step 2.1 in $H1$, only one machine is activated and thus $c^{H1} = 1 + P$. Furthermore, $P - 1 < p_t \leq P$ implies that $L = p_t$ due to $P > 3$. Thus we have $c^* \geq \min\{1 + P, 2 + L\} = 1 + P$ according to Theorem 1, that is, $c^{H1} = c^*$.

If $\frac{P}{3} < p_t \leq P - 1$, from step 2.2 of $H1$, only job p_t is scheduled onto machine M_2, and all the others onto machine M_1, then we have $c^{H1} = 2 + \max\{p_t, P - p_t\}$. Moreover, if $\frac{P}{2} \leq p_t \leq P - 1$, then $p_t = L$ and thus $c^* \geq \min\{1 + P, 2 + p_t\} = 2 + p_t = c^{H1}$. On the other hand, if $\frac{P}{3} \leq p_t \leq \frac{P}{2}$, we have $c^{H1} = 2 + \max\{P - p_t, p_t\} = 2 + P - p_t \leq 2 + \frac{2P}{3}$. By Theorem 1, $c^* \geq \min\{1 + P, 2 + \frac{P}{2}\} = 2 + \frac{P}{2}$ due to $P \geq 3$. It follows that $\frac{c^{H1}}{c^*} \leq \frac{2+\frac{2P}{3}}{2+\frac{P}{2}} = \frac{12+4P}{12+3P}$.

Finally, we consider the case of $p_t \le \frac{P}{3}$. By the definition of p_t, we have $\sum_{i=1}^{t-1} p_i < \frac{2P}{3}$ and $\sum_{i=1}^{t} p_i > \frac{2P}{3}$, which, together with $p_t \le \frac{P}{3}$, leads to $\sum_{i=1}^{t-1} p_i > \frac{2P}{3} - p_t > \frac{P}{3}$ and thus $\sum_{i=t}^{n} p_i < \frac{2P}{3}$. From step 2.3, we have $c^{H1} = 2 + \max\{\sum_{i=t}^{n} p_i, \sum_{i=1}^{t-1} p_i\} < 2 + \frac{2P}{3}$. It follows that $\frac{c^{H1}}{c^*} \le \frac{2+\frac{2P}{3}}{2+\frac{P}{2}} = \frac{12+4P}{12+3P}$ due to $c^* \ge \min\{1+P, 2+\frac{P}{2}\} = 2+\frac{P}{2}$. □

Now we give the main Theorem of this section:

Theorem 3. *Algorithm $H1$ has a parameter competitive ratio of*

$$\begin{cases} 1, & 0 < P \le 2, \\ \frac{2+2P}{4+P}, & 2 < P \le 3, \\ \frac{12+4P}{12+3P}, & P > 3, \end{cases}$$

and it is optimal.

Proof. The competitive ratio of algorithm $H1$ is obtained by Lemmas 1 and 2, which can be illustrated as Figure 1. Moreover, the optimality of algorithm $H1$ is a direct consequence of Theorem 2. □

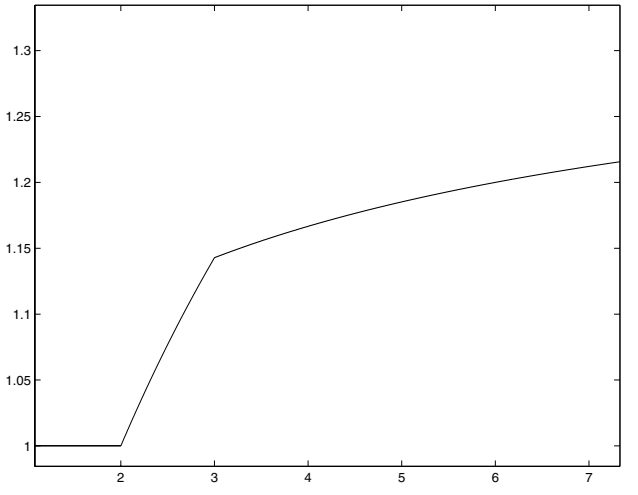


Fig. 1. The competitive ratio of $H1$ for all $P > 0$

4 Problem $P2|max|C_{\max} + m'$

In this section, we assume that the largest size L of all jobs is known in advance. We first give the lower bound of the problem, then present an algorithm $H2$ which is optimal for any $L > 0$.

4.1 Lower Bound

We present the lower bound of the considered problem. The proof is more complicated than that of Theorem 2. We first give the following technical lemmas.

Lemma 3. *If $0 < L \leq 1$, then the competitive ratio of any semi-on-line algorithm for problem $P2|max|C_{\max} + m'$ is at least $\frac{4}{3}$.*

Proof. Assume that an algorithm A exists and has a competitive ratio $C < \frac{4}{3}$.

We show $C \geq \frac{4}{3}$ if $0 < L \leq 1$. The first job $p_1 = L$ arrives, then machine M_1 must be activated to process job p_1. Consider a sequence of jobs with each job $p_i = \varepsilon$ for all $i \geq 2$, where ε is a very small positive number. We can claim that algorithm A must assign the first k jobs (if any) to M_1 to avoid $C \geq \frac{4}{3}$, with k satisfying $P_k = \sum_{i=1}^{k} p_i = 5$. Otherwise, we let $p_l, 1 < l \leq k$, be the first job to be assigned to machine M_2 and no other new jobs arrive after scheduling p_l. Thus we have $P_{l-1} < 5$ and $c^A = 2 + P_{l-1}$.

If $P_{l-1} = \sum_{i=1}^{l-1} p_i < 2$, then the optimal value $c^* = 1 + P_l = 1 + P_{l-1} + \varepsilon$, which follows that

$$C \geq \frac{2 + P_{l-1}}{1 + P_{l-1} + \varepsilon} \geq 1 + \frac{1}{1 + 2 + \varepsilon} \to \frac{4}{3} \quad (\varepsilon \to 0).$$

If $2 \leq P_{l-1} < 5$, then we have $c^* = 2 + \frac{P_l}{2} = 2 + \frac{P_{l-1}+\varepsilon}{2}$. It follows that

$$C \geq \frac{2 + P_{l-1}}{2 + \frac{P_{l-1}+\varepsilon}{2}} \geq \frac{2 + 2}{2 + \frac{2+\varepsilon}{2}} \to \frac{4}{3} \quad (\varepsilon \to 0).$$

Thus algorithm A must assign the first k jobs to the machine M_1 completely. Then no new jobs arrive and thus $c^A \geq 1 + P_k = 6$, while $c^* = 2 + \frac{P_k}{2} = 2 + \frac{5}{2}$. It yields that $C \geq \frac{c^A}{c^*} \geq \frac{6}{2+\frac{5}{2}} = \frac{4}{3}$. The proof is completed. □

Lemma 4. *If $1 < L \leq \frac{1+\sqrt{17}}{2}$, then the competitive ratio of any semi-on-line algorithm for problem $P2|max|C_{\max} + m'$ is at least $\frac{3+L}{2+L}$.*

Proof. The proof is similar as that of Lemma 3 except for the size of P_k. Here let $P_k = \sum_{i=1}^{k} p_i = \begin{cases} \frac{8+2L}{L+1}, & 1 < L \leq 2, \\ 2L, & 2 < L \leq \frac{1+\sqrt{17}}{2}. \end{cases}$. We only prove the result for $1 < L \leq 2$. For the case of $2 < L \leq \frac{1+\sqrt{17}}{2}$, the desired result can be obtained by the same argument.

We claim that algorithm A must activate only one machine to process all the first k jobs to avoid $C \geq \frac{L+3}{L+2}$. Otherwise, we also let $p_l, 1 < l \leq k$, be the first job to be assigned to machine M_2 and no other new jobs arrive after scheduling p_l. Thus we have $P_{l-1} < \frac{8+2L}{L+1}$ and $c^A = 2 + P_{l-1}$. Similarly, we prove the result according to the size of P_{l-1}.

(i) $L \le P_{l-1} < L+1$. Then we have $c^* = 1 + P_l = 1 + P_{l-1} + \varepsilon$ and thus

$$C \ge \frac{2+P_{l-1}}{1+P_{l-1}+\varepsilon} \ge \frac{2+L+1}{1+L+1+\varepsilon} \to \frac{3+L}{2+L} \quad (\varepsilon \to 0).$$

(ii) $L+1 \le P_{l-1} < 2L$. It is easy to obtain that $c^* = 2+L$. It follows that

$$C \ge \frac{2+P_{l-1}}{L+2} = \frac{2+P_l-\varepsilon}{2+L} \ge \frac{2+L+1-\varepsilon}{2+L} \to \frac{3+L}{2+L} \quad (\varepsilon \to 0).$$

(iii) $2L \le P_{l-1} < \frac{8+2L}{L+1}$. Then we have $c^* = 2 + \frac{P_l}{2} = 2 + \frac{P_{l-1}+\varepsilon}{2}$ and thus

$$C \ge \frac{2+P_{l-1}}{2+\frac{P_{l-1}+\varepsilon}{2}} \ge \frac{2+2L}{2+\frac{2L+\varepsilon}{2}} \ge \frac{3+L}{2+L+\frac{\varepsilon}{2}} \to \frac{3+L}{2+L} \quad (\varepsilon \to 0).$$

From above argument, we are sure that the algorithm A must assign all the first k jobs to the machine M_1. Then no new jobs arrive and thus $c^A \ge 1 + P_k$, while $c^* = 2 + L + \frac{P_k}{2}$. Recall that $P_k = \frac{2L+8}{L+1}$. Hence, $C \ge \frac{c^A}{c^*} \ge \frac{1+P_k}{2+\frac{P_k}{2}} = \frac{3+L}{2+L}$. The proof is completed. □

Lemma 5. *If $L > \frac{1+\sqrt{17}}{2}$, then the competitive ratio of any semi-on-line algorithm for problem $P2|max|C_{\max} + m'$ is at least $\frac{4+4L}{4+3L}$.*

Proof. Assume that an algorithm A exists and has a competitive ratio C. The first two jobs $p_1 = p_2 = \frac{L}{2}$ arrive. If algorithm A activates only one machine to process the jobs, then the last two jobs $p_3 = L$ and $p_4 = L$ arrive. It implies that $c^A \ge 2 + 2L$, while $c^* = 2 + \frac{3L}{2}$. Thus $C \ge \frac{c^A}{c^*} \ge \frac{2+2L}{2+\frac{3L}{2}} = \frac{4+4L}{4+3L}$. If A activates two machines and schedules p_1 and p_2 onto different machines. Then the last job $p_3 = L$ arrives, which yields that $c^A = 2 + \frac{3L}{2}$ and $c^* = 2 + L$. It follows that

$$C \ge \frac{c^A}{c^*} = \frac{2+\frac{3L}{2}}{2+L} = \frac{2+2L-\frac{L}{2}}{2+\frac{3L}{2}-\frac{L}{2}} > \frac{2+2L}{2+\frac{3L}{2}} = \frac{4+4L}{4+3L}.$$

The proof is completed. □

Together with Lemmas 3, 4 and 5, we have the following Theorem for the lower bound of the problem $P2|max|C_{max} + m'$.

Theorem 4. *The competitive ratio of any semi-on-line algorithm for problem $P2|max|C_{\max} + m'$ is at least*

$$\begin{cases} \frac{4}{3}, & 0 < L \le 1, \\ \frac{3+L}{2+L}, & 1 < L \le \frac{1+\sqrt{17}}{2}, \\ \frac{4+4L}{4+3L}, & L > \frac{1+\sqrt{17}}{2}. \end{cases}$$

4.2 Optimal Algorithm $H2$

We present an optimal algorithm $H2$ for the problem $P2|max|C_{\max} + m'$. We divide the algorithm into two parts according to the value of L. Part 1 is used to deal with the case of $L \leq 1$ and Part 2 to deal with the case of $L > 1$. The detail can be described below.

Algorithm $H2$
Part 1. (For the case $L \leq 1$)

1. Activate machine M_1 to process job p_1. Let $k = 1$, $m_1 = 1$.
2. **If** no new job arrives, stop. Otherwise, let $k = k + 1$.
3. **If** $m_{k-1} = 1$ and $s_{1,k-1} + p_k \leq 2$, schedule p_k on machine M_1. Let $m_k = m_{k-1}$. Return step 2.
4. **If** $m_{k-1} = 1$ and $s_{1,k-1} + p_k > 2$, activate machine M_2 to process job p_k. Let $m_k = m_{k-1} + 1$. Return step 2.
5. **If** $m_{k-1} = 2$, schedule p_k by LS rule (LS rule means that the job is scheduled on the machine with minimum current load). Return step 2.

Part 2. (For the case $L > 1$)

1. Activate machine M_1 to process job p_1. Let $k = 1$, $m_1 = 1$.
2. **If** no new job arrives, stop. Otherwise, let $k = k + 1$.
3. **If** $m_{k-1} = 1$ and the largest job is revealed.
   3.1. If $s_{1,k-1} + p_k \leq (L + 1)$, schedule p_k on machine M_1. Let $m_k = m_{k-1}$. Return step 2.
   3.2. If $s_{1,k-1} + p_k > L + 1$, activate machine M_2 to process job p_k. Let $m_k = m_{k-1} + 1$. Return step 2.
4. **If** $m_{k-1} = 1$ and the largest job is not yet revealed.
   4.1. If $s_{1,k-1} + p_k \leq 2L$, schedule p_k on machine M_1. Let $m_k = m_{k-1}$. Return step 2.
   4.2. If $s_{1,k-1} + p_k > 2L$, activate machine M_2 to process job p_k. Let $m_k = m_{k-1} + 1$. Return step 2.
5. **If** $m_{k-1} = 2$, schedule p_k by LS rule. Return step 2.

Lemma 6. *If* $L \leq 1$, *then* $\frac{c^{H2}}{c^*} \leq \frac{4}{3}$.

Proof. We only need to consider Part 1 of algorithm $H2$. If $H2$ only activates one machine M_1, then it is clear that $P_n \leq 2$ and thus $c^{H2} = 1 + P_n = c^*$.

We now deal with the case that $H2$ activates both machines. Then we have $P_n > 2$ by the rule of the algorithm. Let p_l be the job that determines the makespan. We can conclude that it is impossible to schedule p_l by step 4. Otherwise, by the rule of step 4, p_l is the first job assigned to the machine M_2. Moreover, by the definition of p_l, we obtain that p_l is the unique job processed on machine M_2, then $c^{H2} = 2 + p_l$. However, step 4 states that $P_l = P_{l-1} + p_l > 2$ and $P_{l-1} < 2$. Note that $p_l \leq L \leq 1$ implies $s_{1,l} = P_{l-1} > s_{2,l} = p_l$. It follows

that $c^{H2} = 2+P_{l-1} > 2+p_l = c^{H2}$, a contradiction. Hence, p_l must be scheduled by step 3 or step 5.

If p_l is scheduled by step 3, then we have $c^{H2} \le 2 + s_{1,l} \le 4$. By Theorem 1, we have $c^* \ge \min\{1 + P_n, 2 + \frac{P_n}{2}\} = 2 + \frac{P_n}{2} \ge 3$ and thus $\frac{c^{H2}}{c^*} \le \frac{4}{3}$.

If p_l is scheduled by step 5, then $P_l > 2$, which yields that $\max\{s_{1,l}, s_{2,l}\} > 1$. Since p_l is scheduled by LS rule, we can conclude that $|s_{1,l} - s_{2,l}| \le p_l \le L$, i.e., $\min\{s_{1,l}, s_{2,l}\} \ge \max\{s_{1,l}, s_{2,l}\} - L$. We know that $c^{H2} = 2 + \max\{s_{1,l}, s_{2,l}\}$ and

$$c^* \ge 2 + \frac{P_l}{2} = 2 + \frac{s_{1,l} + s_{2,l}}{2} = 2 + \frac{\min\{s_{1,l}, s_{2,l}\} + \max\{s_{1,l}, s_{2,l}\}}{2}.$$

It follows that

$$\begin{aligned}
\frac{c^{H2}}{c^*} &\le \frac{2 + \max\{s_{1,l}, s_{2,l}\}}{2 + \frac{\min\{s_{1,l},s_{2,l}\}+\max\{s_{1,l},s_{2,l}\}}{2}} \\
&\le \frac{2 + \max\{s_{1,l}, s_{2,l}\}}{2 + \frac{\max\{s_{1,l},s_{2,l}\}-L+\max\{s_{1,l},s_{2,l}\}}{2}} \\
&= \frac{2 + \max\{s_{1,l}, s_{2,l}\}}{2 + \max\{s_{1,l}, s_{2,l}\} - \frac{L}{2}} \\
&\le \frac{2+1}{2+1-\frac{L}{2}} < \frac{3}{3-\frac{1}{2}} = \frac{6}{5} < \frac{4}{3}.
\end{aligned}$$

□

Lemma 7. *If $L \ge 1$, then $\frac{c^{H2}}{c^*} \le \max\{\frac{L+3}{L+2}, \frac{4+4L}{4+3L}\}$.*

Proof. We consider Part 2 of algorithm $H2$. If $H2$ only activates M_1 to process all the jobs, then we conclude that $P_n \le L+1$ by step 3.1 and have $c^{H2} = 1+P_n \le L+2$. Due to Theorem 1, we obtain that $c^* \ge \min\{L+2, 1+P_n\} = 1+P_n = c^{H2}$.

Now we deal with the case that both machines are activated by $H2$. As the algorithm states that the second machine is activated by step 3.2 or 4.2, and after that, all the remaining jobs are scheduled by LS rule, we can discuss two cases according to the activation of M_2. Let p_l be the job that determines the makespan.

Case 1. M_2 is activated by step 3.2. If p_l is scheduled by step 3, it is not hard to obtain that $C^{H2} = \max\{s_{1,l}, s_{2,l}\} \le L+1$ and thus $c^{H2} = 2+C^{H2} \le 3+L$. From step 3.2 we have $P_l \ge L+1$, then $c^* \ge L+2$ due to Theorem 1. Hence, $\frac{c^{H2}}{c^*} \le \frac{3+L}{2+L}$. If p_l is scheduled by LS rule. Note that after activating the second machine, at least one machine has a load no less than L. Then we have $\max\{s_{1,l-1}, s_{2,l-1}\} > L$, which implies that $\min\{s_{1,l}, s_{2,l}\} > L$ due to the definition and assignment of p_l. We can also conclude that $\min\{s_{1,l}, s_{2,l}\} + L \ge \max\{s_{1,l}, s_{2,l}\}$ due to $p_l \le L$. Since $c^{H2} = 2 + \max\{s_{1,l}, s_{2,l}\}$ and $c^* \ge 2 + \frac{P_l}{2} = 2 + \frac{s_{1,l}+s_{2,l}}{2} = 2 + \frac{\min\{s_{1,l},s_{2,l}\}+\max\{s_{1,l},s_{2,l}\}}{2}$, we obtain

$$\frac{c^{H2}}{c^*} \le \frac{2 + \max\{s_{1,l}, s_{2,l}\}}{2 + \frac{\min\{s_{1,l},s_{2,l}\}+\max\{s_{1,l},s_{2,l}\}}{2}}$$

$$\leq \frac{2+L+\min\{s_{1,n}, s_{2,l}\}}{2+\frac{\min\{s_{1,l},s_{2,l}\}+L+\min\{s_{1,l},s_{2,l}\}}{2}}$$
$$= \frac{2+L+\min\{s_{1,l}, s_{2,l}\}}{2+\min\{s_{1,l}, s_{2,l}\}+\frac{L}{2}} \leq \frac{2+2L}{2+L+\frac{L}{2}} = \frac{4+4L}{4+3L}.$$

Case 2. M_2 is activated by step 4.2. By the rule of step 4, we conclude that when the second machine M_2 is activated to process the first job, no largest jobs are yet revealed. Thus we only need to consider the competitive ratio of $H2$ after assigning the first largest job with size of L. As the remaining jobs(including the largest jobs) will be scheduled by step 5 (LS rule), after assigning the the first largest job, denoted by p_k, it is not hard to obtain that $\min\{s_{1,k}, s_{2,k}\} > L$ and $\min\{s_{1,k}, s_{2,k}\} + L \geq \max\{s_{1,k}, s_{2,k}\}$. Moreover, for any $i \geq k$, p_i is scheduled by LS rule, then $\min\{s_{1,i}, s_{2,i}\} > L$ and $\min\{s_{1,i}, s_{2,i}\} + L \geq \max\{s_{1,i}, s_{2,i}\}$ hold. By the same argument of Case 1, we can obtain the desired competitive ratio.

Hence, for $L > 1$, $\frac{c^{H2}}{c^*} \leq \max\{\frac{L+3}{L+2}, \frac{4+4L}{4+3L}\} = \begin{cases} \frac{3+L}{2+L}, & 1 < L \leq \frac{1+\sqrt{17}}{2}, \\ \frac{4+4L}{4+3L}, & L > \frac{1+\sqrt{17}}{2}. \end{cases}$

□

Theorem 5. *Algorithm $H2$ has a competitive ratio of*

$$\begin{cases} \frac{4}{3}, & 0 < L \leq 1, \\ \frac{3+L}{2+L}, & 1 < L \leq \frac{1+\sqrt{17}}{2}, \\ \frac{4+4L}{4+3L}, & L > \frac{1+\sqrt{17}}{2}, \end{cases}$$

and it is optimal.

Proof. The competitive ratio of algorithm $H2$ is directly from above Lemmas 6 and 7, which can be illustrated as Figure 2. Moreover, the optimality of algorithm $H2$ is a direct consequence of Theorem 4. □

5 Final Remarks

As we know that both the optimal algorithms for the classic semi-on-line problems $P2|sum|C_{\max}$ [9] and $P2|max|C_{\max}$ [6] have a constant competitive ratio of 4/3 for any $P > 0$ and $L > 0$. In this paper we studied two problems $P2|sum|C_{\max}+m'$ and $P2|max|C_{\max}+m'$ and presented two optimal algorithms $H1$ and $H2$ with the parameter competitive ratios for every $P > 0$ and $L > 0$, respectively. Clearly, our problems are more complicated than the classic ones not only from the problems themselves but also the performances of the bounds. It is an interesting thing that the bound of the problem $P2|max|C_{\max}+m'$ first decreases and then increases as L increases, see Figure 2.

As this paper only considered the case of two identical machines, it can be very interesting to extend the results to general m machines case.

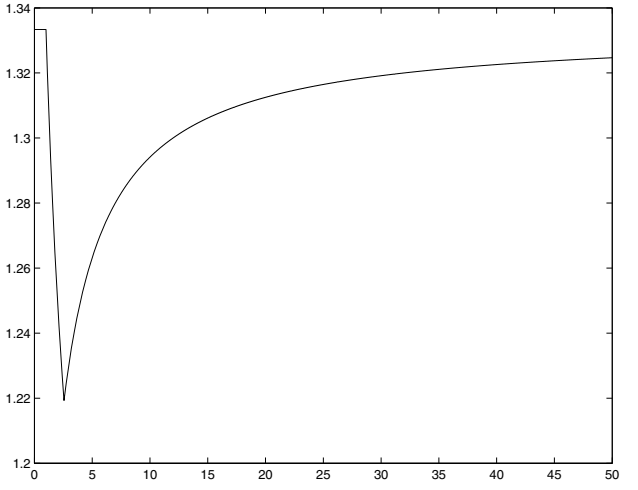


Fig. 2. The competitive ratio of $H2$ for all $L > 0$

References

1. Cao, D., Chen, M., Wan, G.: Parallel machine selection and job scheduling to minimize machine cost and job tardiness. Comput. & Oper. Res. 32, 1995–2012 (2005)
2. Dósa, G., He, Y.: Better on-line algorithms for scheduling with machine cost. SIAM Journal on Computing 33, 1035–1051 (2004)
3. Graham, R.L.: Bounds on muliprocessing finishing anomalies. SIAM Journal on Applied Mathematics 17, 416–429 (1969)
4. Han, S.G., Jiang, Y.W., Hu, J.L.: On-line algorithms for scheduling with machine activation cost on two uniform machines. Journal of Zhejiang University SCIENCE 8(1), 127–133 (2007)
5. He, Y., Han, S.G., Jiang, Y.W.: On-line algorithms for scheduling with machine activation cost. Asia-Pacific Journal of Operations research 24(2), 263–277 (2007)
6. He, Y., Zhang, G.: Semi-on-line scheduling on two identical machines. Computing 62, 179–187 (1999)
7. Imreh, C., Noga, J.: Scheduling with machine cost. In: Hochbaum, D.S., Jansen, K., Rolim, J.D.P., Sinclair, A. (eds.) RANDOM 1999 and APPROX 1999. LNCS, vol. 1671, pp. 168–176. Springer, Heidelberg (1999)
8. Jiang, Y.W., He, Y.: Preemptive on-line algorithms for scheduling with machine cost. Acta Informatica 41, 315–340 (2005)
9. Kellerer, H., Kotov, V., Speranza, M.R., Tuza, Z.: Semi on-line algorithms for the partition problem. Operation Research Letters 21, 235–242 (1997)
10. Panwalker, S., Liman, S.D.: Single operation earliness-tardiness scheduling with machine activation cost. IIE Transactions 34, 509–513 (2002)
11. Tan, Z.Y., He, Y.: Semi-on-line problems on two identical machines with combined partial information. Operation Research Letters 30, 408–414 (2002)

A Fast Asymptotic Approximation Scheme for Bin Packing with Rejection

Wolfgang Bein[1,⋆], José R. Correa[2,⋆⋆], and Xin Han[3,⋆⋆⋆]

[1] Center for the Advanced Study of Algorithms, School of Computer Science, University of Nevada, Las Vegas, NV 89154, USA
bein@cs.unlv.edu
[2] School of Business, Univesidad Adolfo Ibáñez, Santiago, Chile
correa@uai.cl
[3] School of Informatics, Kyoto University, Kyoto 606-8501, Japan
hanxin@kuis.kyoto-u.ac.jp

Abstract. The *bin packing with rejection* problem is the following: Given a list of items with associated sizes and rejection costs, find a packing into unit bins of a subset of the list, such that the number of bins used plus the sum of rejection costs of unpacked items is minimized. In this paper, we first show that bin packing with rejection can be reduced to n multiple knapsack problems. Then, based on techniques for the multiple knapsack problem we give a fast asymptotic polynomial time approximation scheme with time complexity $O(n^{O(\epsilon^{-2})})$. This improves a recent approximation scheme given by Epstein, which has time complexity $O(n^{O((\epsilon^{-4})^{\epsilon^{-1}})})$. Finally, we show that our algorithm can be extended to variable-sized bin packing with rejection and give an asymptotic polynomial time approximation scheme for it.

1 Introduction

In the bin packing problem items of specified size have to be packed into the minimum number of unit bins. This problem, of particular interest in operations research and computer science, has been extensively studied since the Sixties (see [3] for a detailed survey). Interestingly, although bin packing is NP-hard, efficient algorithms for computing solutions which are arbitrarily close to optimal (asymptotically) were obtained early on by Fernandez de la Vega and Luecker [6], and by Karmarkar and Karp [9]. More recently, packing and scheduling problems with rejection penalties, which are natural variants of their classic counterparts, have received attention (see e.g. [1,4,8]). Here one can decide to refuse an item, but in doing so one is charged a certain cost. This setting is of interest since in many practical situations one may choose to leave some items unpacked (or

⋆ Research conducted while visiting Japan as Kyoto University Visiting Professor.
⋆⋆ Supported in part by CONICYT through grants FONDECYT 10600035 and ACT08.
⋆⋆⋆ Supported by NSFC (10231060).

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 209–218, 2007.

unscheduled) at a certain price. For instance, this problem arises naturally in the context of outsourcing tasks (e.g. shipments), in which a third party can take care of processing a task as long as they are paid for it. Other examples are in caching and data storage across a network, where an item can be cached or stored locally in advance, or, instead, may be fetched at cost when needed.

Formally, in the bin packing with rejection problem we are given a list I of items. Each item $i \in I$ has an associated size s_i, and a rejection cost r_i. The goal is to find a partition of I into a set A of "accepted" items and a set R of "rejected" items, together with a packing of A into unit bins, such that the number of bins needed to pack A plus the total price of items in R is minimized. (Clearly the NP-hardness of the bin packing with rejection problem follows from that of bin packing.) This particular problem, was recently studied by He and Dósa [4], who obtain several online and off line results. In particular, they give an algorithm with asymptotic approximation ratio of 3/2. Epstein [5] gave the first asymptotic polynomial time approximation scheme (APTAS) for the bin packing with rejection problem based on classical bin packing techniques.

Throughout this paper we will make use of the notion of *guessing*, see e.g. [2]. One "guesses" a quantity if one can construct a polynomial size set of values, which contains the desired value and a certificate can be constructed in polynomial time. By an abuse of terminology we will also use the term "guessing" for constructing the certificate itself.

Our contribution. Borrowing techniques for the multiple knapsack problem [2,10], we provide an APTAS for bin packing with rejection which turns out to be more efficient than that of Epstein. Furthermore we extend our scheme to variable-sized bin packing with rejection and give an APTAS for this problem as well. Specifically, we show that the bin packing with rejection problem can be reduced to n multiple knapsack problems. Based on the techniques for the multiple knapsack problem, we give a fast asymptotic polynomial time approximation scheme with time complexity $O(n^{O(\epsilon^{-2})})$ thereby improving the current best $O(n^{O((\epsilon^{-4})^{\epsilon^{-1}})})$ scheme [5]. Our techniques directly apply to the variable-sized case.

The outline of this approach is as follows: i) First, we guess a packing cost $cost_p$ and a rejection cost $cost_r$ such that the two values are not much larger than those of an optimal solution, respectively. ii) Then, based on the two guessed values, we guess a sublist L_r of the input list L such that the total profit (rejection cost) in L_r is bounded by $(1 + O(\epsilon)cost_r)$ and all items in $L - L_r$ can be packed into bins with $cost_p$. iii) Finally, we call a bin packing algorithm to pack all the items in $L - L_r$. The key point is that we can guarantee that all the guesses above can be done in polynomial time of n, where n is the size of the input list.

2 An APTAS for the Bin Packing with Rejection

We first show that for solving the problem of bin packing with rejection we only need to solve at most n multiple knapsack problems, where n is the size of the input list. Note that any optimal value $OPT(L)$ can be written as follows:

$$OPT(L) = OPT_p(L) + OPT_r(L),$$

where L is the input list, $OPT_p(L)$ is the cost for packing, i.e., the number of bins used, $OPT_r(L)$ is the total rejection cost for rejection.

Definition 1. *Given a list L, we denote the total rejection cost in it by $p(L)$.*

Lemma 1. *Assume OPT_p is fixed, then the original problem is equivalent to the multiple knapsack problem of packing the input list into OPT_p bins to maximize the total profit of the bins.*

Proof. Let $OPT_p = i$ and $L = L_p \cup L_r$, where L_p, L_r are the packed list and the rejected list, respectively. Then we have

$$\begin{aligned} OPT(L) &= OPT_p(L) + OPT_r(L) \\ &= i + \textstyle\sum_{j \in L_r} r_j \\ &= (i + \textstyle\sum_{j \in L} r_j) - \textstyle\sum_{j \in L_p} r_j \end{aligned}$$

By the above equation, if $\sum_{j \in L_p} r_j$ is maximized, then $OPT(L)$ reaches its minimum. In fact, maximizing $\sum_{j \in L_p} r_j$ is equivalent to selecting a sublist L_p from the input list L and packing them into i bins to maximize the total rejection cost of the bins. Thus, if the knapsack problem of packing L into i bins is solved, then the original problem is solved as well. □

Lemma 1 naturally suggests the following algorithm:

- Guess OPT_p from 1 to n, i.e., the number of bins to be used.
- Consider rejection costs as profits, then call PTAS for the multiple knapsack problem [2,10] to pack items into OPT_p bins and regard the unpacked items as the rejected items.
- Output $\min\{OPT_p + R\}$ over all OPT_p, where R is the total cost of all rejected items.

Unfortunately, this does not yield an APTAS. In the above algorithm, the packing cost is OPT_p, i.e., the number of bins used, is OPT_p. Assume now that the optimal value of maximizing the total profit (rejection cost) packed into bins with OPT_p is X and the total profit packed by the PTAS of [2,10] is $(1-\epsilon)X$. Then the total cost from our algorithm is

$$\epsilon X + OPT(L).$$

The value of ϵX can be large, even larger than $OPT(L)$, thus we can not guarantee that the above algorithm always produces a solution near the optimal cost.

Instead, we focus on the rejected list to obtain our APTAS. More precisely, we guess a rejected list such that its total rejection cost is at most $(1+O(\epsilon))OPT_r(L)$

and such that the remaining items in L can be packed into bins with $OPT_p(L)$. Before we give details we recall the following lemma:

Lemma 2. *[9] For the bin packing problem there is an algorithm which runs in time polynomial in n and $1/\varepsilon$ and finds a solution that uses at most*

$$(1+\epsilon)OPT + O(\epsilon^{-2})$$

bins, where OPT is the optimal number of bins and $\epsilon > 0$ is sufficiently small.

Let OPT_p and OPT_r be the packing cost and the rejection cost in some optimal solution. There are three steps in our algorithm. First, we guess OPT_p and OPT_r. Then, based on these two values, we guess the packed list and the rejected list. Finally, we pack the packed list by a classical bin packing algorithm and reject all other items.

Our algorithm

1. Guess the packing cost $cost_p$ and the rejection cost $cost_r$ such that $cost_p = OPT_p$ and $OPT_r \leq cost_r \leq (1+\epsilon)OPT_r + 1$.
2. Guess a rejected list $L_r^{'}$ such that $cost_r \leq p(L_r^{'}) \leq (1+O(\epsilon))cost_r$, where $p(L_r^{'})$ is the total rejection cost in $L_r^{'}$, and the remaining items $L - L_r^{'}$ can be packed $cost_p$ bins.
3. Call APTAS to pack $L - L_r^{'}$ and reject all items in $L_r^{'}$ output the bins used and $p(L_r^{'})$.

2.1 Guessing the Rejection Cost and the Packing Cost

Since every item can be packed into one bin, the optimal value for n items is at most n and the optimal values for OPT_p and OPT_r are at most n as well. Then there are $n+1$ possibilities for OPT_p. For the value of the rejection cost OPT_r, if $OPT_r <= 1$ holds then we can instead estimate $cost_r = 1$ for the asymptotic approximation. Else, if $(1+\epsilon)^i \leq OPT_r < (1+\epsilon)^{i+1}$ holds then we have $cost_r = (1+\epsilon)^{i+1}$, i.e., we guess the optimal rejection cost as $(1+\epsilon)^{i+1}$, where ϵ is the error bound and $i \geq 0$ is an integer. Since $OPT_r \leq OPT(L) \leq n$, we have

$$i \leq \frac{\ln n}{\ln(1+\epsilon)} \leq \frac{2\ln n}{\epsilon},$$

where the last inequality follows from $\ln(1+\epsilon) \geq \epsilon - \epsilon^2/2$ and $\epsilon \leq 1/2$. Thus we have the following lemma:

Lemma 3. *We can guess $cost_r$ and $cost_p$ in time $O(\epsilon^{-1} n \ln n)$ such that*

$$cost_p = OPT_p, \qquad OPT_r \leq cost_r \leq (1+\epsilon)OPT_r + 1,$$

where $\epsilon > 0$ is the error bound and OPT_p (OPT_r) is the packed (rejection) cost in some optimal solution.

2.2 Guessing the Rejected List and the Packed List

We briefly note the following simple lemma:

Lemma 4. *In a given input list L, for any item $i \in L$, if $r_i > 1$ then in some optimal solution item i should be in the packed list.*

Proof. If an item i with $r_i > 1$ is rejected in some optimal solution then we can reduce the total cost of the optimal solution by packing item i itself into a single bin. □

We describe how to guess the rejected items and the packed items such that the total cost of the rejected and packed items are near the optimal values, based on the rejection cost $cost_r$ and the packed cost $cost_p$ guessed in step 1. In the following, we denote the guessed rejected (packed) list by $L_r^{'}$ ($L_p^{'}$).

How to guess the rejected list and the packed list

2.1 Place every item with rejection cost larger than 1 into packed list $L_p^{'}$ and all items with rejection cost smaller than $1/n$ into the rejected list $L_r^{'}$.

2.2 Consider the remaining items with the rejection cost in $[1/n, 1]$. There are three phases in assigning all the remaining items into $L_p^{'}$ and $L_r^{'}$: i) rounding down each item's rejection cost to obtain $O(\epsilon^{-1} \ln n)$ sublists such that in each sublist all the items have the same rejection cost, ii) then guessing the rejected items in each sublist and placing them into $L_r^{'}$ such that $L_r^{'}$ has its rejection cost in $[cost_r, (1 + O(\epsilon)cost_r)]$, iii) lastly, placing all remaining items into $L_p^{'}$, testing the feasibility of packing $L_p^{'}$ into bins with $cost_p$.

Below are the details of the three phases in step 2.2; we refer to them as "*rounding down*", "*guessing the rejected list*" and "*testing the packed list.*"

Rounding down. Given an item with a rejection cost r, we round r down to $\bar{r}$ for some integer $i \geq 0$ such that

$$\bar{r} = \frac{(1+\epsilon)^i}{n} \leq r < \frac{(1+\epsilon)^{i+1}}{n},$$

where ϵ is the given error bound. Since $r \leq 1$ we have $\frac{(1+\epsilon)^i}{n} \leq 1$ and

$$i \leq \frac{\ln n}{\ln(1+\epsilon)} \leq 2\epsilon^{-1} \ln n.$$

Then we get $h \leq 2\epsilon^{-1} \ln n$ sublists L_i and in L_i all items have the same rejection cost $\frac{(1+\epsilon)^i}{n}$.

Guessing the rejected list. Let U be the rejected items in some optimal solution of $\cup_i L_i$ and let $U_i = L_i \cap U$. Next, we enumerate a polynomial set of candidates for the rejected list such that one of them includes U and the total rejection cost in them is at most $(1 + O(\epsilon))cost_r$. Let $p(U_i)$ be the total rejection cost in U_i. We select items from L_i as follows. We guess the values

$p(U_i)$ approximately for $0 \leq i \leq h$. For each i we guess a value $k_i \in [0..h/\epsilon]$ such that

$$k_i(\epsilon \cdot cost_r/h) \leq p(U_i) \leq (k_i+1)(\epsilon \cdot cost_r/h).$$

Then we guess the rejected items in each sublist L_i. Let a_i be the rejection cost of one item in L_i. For each k_i, for $0 \leq i \leq h$, we order all items in L_i in order of non-decreasing size values. Then if $a_i > \epsilon \cdot cost_r/h$ then pick the largest $\lceil k_i(\epsilon \cdot cost_r/h)/a_i \rceil$ items from L_i and put them into $L_r^{'}$, else pick the largest $\lfloor (k_i+1)(\epsilon \cdot cost_r/h)/a_i \rfloor$ items from L_i and put them into $L_r^{'}$. Later, we show all the candidates for k_i can be bounded by $O(n^{\epsilon^{-2}})$, i.e., there are a polynomial number of rejected lists.

Testing the packed list. Each time, after selecting the rejected item from L_i, we put the remaining items in L_i into the packed list $L_p^{'}$. Then we try to use APTAS [9] to pack the items in $L_p^{'}$ into $(1+\epsilon)cost_p + O(\epsilon^{-2})$ bins. If all the items in $L_p^{'}$ can be packed then return $L_p{}'$ and $L_r{}'$. Else we try another tuple $(k_1, \ldots, k_h)$.

2.3 Analysis

Now we prove that the number of all candidate h-tuples $(k_1, \ldots, k_h)$ can be bounded by $O(n^{O(1/\epsilon^2)})$, which is polynomial in n. To this end, we first quote an important lemma obtained by Chekuri and Khanna [2]. (We include the short proof for self-containedness.)

Lemma 5. *Let f be the number of g-tuples of non-negative integers such that the sum of tuple coordinates is equal to d. Then $f = \binom{d+g-1}{g-1}$. If $d+g \leq \alpha g$ then $f = O(e^{\alpha g})$.*

Proof. The first part of the lemma is elementary counting. If $d+g \leq \alpha g$ then

$$f \leq \binom{\alpha g}{g-1} \leq \frac{(\alpha g)^{g-1}}{(g-1)!}.$$

Using Stirling's formula we can approximate $(g-1)!$ by $\sqrt{2\pi(g-1)}((g-1)/e)^{g-1}$. Thus $f = O((e\alpha)^{g-1}) = O(e^{\alpha g})$. □

A naive way of guessing the values $k_1, \cdots, k_h$ requires $n^{O(\ln n/\epsilon^2)}$ which is exponential. However, not all the values $k_1, \cdots, k_h$ are independent. Indeed, we have that

$$\sum_i (k_i \cdot \frac{\epsilon \cdot cost_r}{h}) \leq \sum_i p(U_i) = p(U) \leq cost_r,$$

where the last inequality follows from the inequality $p(U) \leq OPT_r \leq cost_r$. By the above observation, we have $\sum_i k_i \leq h/\epsilon = O(\ln n/\epsilon^2)$, then we get the following lemma.

Lemma 6. *Let an integer $h \leq 2\epsilon^{-1} \ln n$. Then the number of h-tuples $(k_1, \ldots, k_h)$ such that $\sum_i k_i \leq h/\epsilon$ is $O(n^{O(\epsilon^{-2})})$.*

Proof. We apply the bound from Lemma 5, we have $\alpha = (1 + 1/\epsilon)$ and $g = 2\epsilon^{-1} \ln n$, hence we get an upper bound $e^{2\epsilon^{-1}(1+1/\epsilon)\ln n}$. □

We introduce the notion of a "*small-packed*" solution:

Definition 2 (Small-packed). *Given a solution of the packed list L_p and the rejected list L_r, if there are two items $i \in L_p$ and $j \in L_r$ such that*

$$r_i = r_j, \quad s_i \leq s_j,$$

then we say the solution is small-packed*, otherwise* not small-packed*.*

We can see if a solution is not *small-packed*, then there exist a packed item i and a rejected item j such that $r_i = r_j, \quad s_i > s_j$. Then we exchange the two items i and j, i.e., pack j and reject i, to get another solution without enlarging the total cost. By the same approach, we can prove the following lemma.

Lemma 7. *Any optimal solution can be reduced to a* small-packed *solution without changing the cost.*

To prove that our scheme works we also need the following lemma:

Lemma 8. *Assume all items have rejection costs equal to $\frac{(1+\epsilon)^i}{n}$ for some integer i, where $0 \leq i \leq h = O(\epsilon^{-1} \ln n)$, and $p(U) \leq cost_r \leq (1 + O(\epsilon))p(U) + 1$, where U is a set of the rejected items in some optimal solution with the* small-packed *property. Then there exists a valid tuple $(k_1, \ldots, k_h)$ associated with $L_r^{'}$ such that $U \subseteq L_r^{'}$ and $p(L_r^{'}) \leq (1 + O(\epsilon))p(U) + 1$.*

Proof. U is a set of the rejected items in some optimal solution with the *small-packed* property. Then for all i, we define

$$k_i = \lfloor p(U_i)h/(\epsilon \cdot cost_r) \rfloor.$$

There are two cases based on the value of a_i, which is the rejection cost of one item in L_i. Assume there are x_i items in U_i. Then $a_i \cdot x_i = p(U_i)$.

(i) $a_i > \epsilon \cdot cost_r/h$. We prove our selection from L_i is as the same as $U_i = U \cap L_i$, i.e., $U_i = L_i \cap L_r^{'}$. By the definition $k_i = \lfloor p(U_i)h/(\epsilon \cdot cost_r) \rfloor$, we have

$$\frac{p(U_i)h}{\epsilon \cdot cost_r} - 1 < k_i \leq \frac{p(U_i)h}{\epsilon \cdot cost_r}.$$

Then

$$p(U_i) - \frac{\epsilon \cdot cost_r}{h} < \frac{k_i \cdot \epsilon \cdot cost_r}{h} \leq p(U_i)$$

Since $a_i > \epsilon \cdot cost_r/h$ and $a_i \cdot x_i = p(U_i)$,

$$x_i - 1 < \frac{p(U_i)}{a_i} - \frac{\epsilon \cdot cost_r}{ha_i} < \frac{k_i \cdot \epsilon \cdot cost_r}{ha_i} \leq \frac{p(U_i)}{a_i} = x_i.$$

So we select $\lceil (k_i \cdot \epsilon \cdot cost_r)/(ha_i) \rceil (= x_i)$ items from L_i. Since our selection is from large to small and the optimal solution with the rejected list U is *small-packed*, our selection from L_i is as the same as $U_i = U \cap L_i$, i.e., $U_i = L_i \cap L_r^{'}$.

(ii) $a_i \leq \epsilon \cdot cost_r/h$. Here we prove that $U_i \subseteq (L_i \cap L_r^{'})$ and $p(L_i \cap L_r^{'}) \leq p(U_i) + \frac{\epsilon \cdot cost_r}{h}$. Again, by the definition $k_i = \lfloor p(U_i)h/(\epsilon \cdot cost_r) \rfloor$, we have

$$\frac{p(U_i)h}{\epsilon \cdot cost_r} < k_i + 1 \leq \frac{p(U_i)h}{\epsilon \cdot cost_r} + 1.$$

Then

$$p(U_i) < \frac{(k_i+1) \cdot \epsilon \cdot cost_r}{h} \leq p(U_i) + \frac{\epsilon \cdot cost_r}{h}$$

So we select $\lfloor ((k_i+1)\epsilon \cdot cost_r)/(ha_i) \rfloor (\geq x_i)$ items from L_i. Since our selection is from large to small and the optimal solution with the rejected list U is *small-packed*, $U_i \subseteq L_i \cap L_r^{'}$. And

$$p(L_i \cap L_r^{'}) \leq \frac{(k_i+1) \cdot \epsilon \cdot cost_r}{h} \leq p(U_i) + \frac{\epsilon \cdot cost_r}{h}.$$

By cases i) and ii), we have $U \subseteq L_r^{'}$ and $p(L_r^{'}) \leq p(U) + \epsilon cost_r$. Since $p(U) \leq cost_r \leq (1 + O(\epsilon))p(U) + 1$, the lemma follows.

□

We are now ready to prove our main result.

Theorem 1. *Our algorithm is an APTAS with time complexity* $O(n^{\epsilon^{-2}})$.

Proof. Let L be the input list and $L_b \subseteq L$ be the list in which all the items have the rejection cost at least $1/n$. So,

$$OPT(L_b) \leq OPT(L) \leq OPT(L_b) + 1. \tag{1}$$

Let $L_b = L_r \cup L_p$, where L_r (L_p) is the rejected (packed) list in some optimal solution of L_b. Then

$$OPT(L_b) = OPT_p + p(L_r), \tag{2}$$

where OPT_p is the number of bins used for L_p and $p(L_r)$ is the total cost for the rejected list L_r. By Lemma 3,

$$p(L_r) \leq cost_r \leq (1+\epsilon)p(L_r) + 1. \tag{3}$$

Lemma 4 says that there are no items with a rejection cost larger than 1 in L_r. After rounding down all the items in L_r at phase 1 of step 2.2, we obtain a new list $\bar{L}_r$. Then

$$p(\bar{L}_r) \leq p(L_r) \leq (1+\epsilon)p(\bar{L}_r).$$

Combining the latter inequality with (3) we obtain,

$$p(\bar{L}_r) \leq cost_r \leq (1 + O(\epsilon))p(\bar{L}_r) + 1.$$

By Lemma 8,

$$p(\bar{L}_r^{'}) \leq (1 + O(\epsilon))p(\bar{L}_r) + 1 \leq (1 + O(\epsilon))p(L_r) + 1, \tag{4}$$

where $\bar{L}_r^{'}$ is the rejected list guessed in step 2.2. Let $L_r^{'}$ be the rejected list before we round down list $\bar{L}_r^{'}$. Then

$$p(L_r^{'}) \leq (1+\epsilon)p(\bar{L}_r^{'}) \leq (1+O(\epsilon))p(L_r) + 1 + \epsilon. \tag{5}$$

Let A be our algorithm and $A(L)$ be the total cost by our algorithm. By (5), Lemma 2 and (2),

$$\begin{aligned} A(L) &\leq (1+\epsilon)OPT_p + O(\epsilon^{-2}) + (1+O(\epsilon))p(L_r) + 1 + \epsilon, \\ &\leq (1+O(\epsilon))OPT(L) + O(\epsilon^{-2}). \end{aligned}$$

As for the time complexity, in step 1, by Lemma 3, our algorithm takes $O(\epsilon^{-1}n\ln n)$ time, in step 2, by Lemma 6, our algorithm takes $O(n^{O(\epsilon^{-2})})$ time, in step 3, our algorithm takes $O(\epsilon^{-8}n\log n)$ time [9]. Then the time complexity of our algorithm is $O(n^{O(\epsilon^{-2})})$. □

3 An APTAS for Variable-Sized Bin Packing with Rejection

In variable-sized bin packing, we are given a set of items L and a set of available bin sizes B. (As in [7,11] we assume, without lost of generality, that the largest bin size in B is 1.) The object is to minimize the sum of the sizes of the bins used. It is clear that this problem is a generalization of the bin packing problem. If each item has both a size and a rejection cost associated with it and the object is to minimize the total cost, then the problem is the variable-sized bin packing with rejection.

In this section, we show that our approach can be extended to the variable-sized bin packing problem with rejection. This follows easily in the following way:

- The rounding technique in step 1 is still available, i.e, the guessed packing cost and rejection cost are near the optimal costs. (See Lemma 10.)
- The technique of rounding down the rejection cost in phase 2 of step 2 is still available, since the rounding is independent of the the packing.
- As with Lemma 2, there is an APTAS for variable-size bin packing as shown in Lemma 9.

Lemma 9. *[11] For the variable-sized bin packing problem, there is a fully polynomial time algorithm with the solution at most*

$$(1+\epsilon)OPT + O(\epsilon^{-4}),$$

where OPT is the optimal value and $\epsilon > 0$ is sufficiently small.

Since every item can be packed into one bin of size 1, the optimal value for n items is at most n. So $OPT_p \leq n$ and $OPT_r \leq n$, where OPT_p (OPT_r) is the total packed (rejection) cost in some optimal solution for variable-sized bin packing with rejection. Then the rounding technique in step 1 is still available and we can prove the following lemma:

Lemma 10. *By the same approach in step 1, there is a packed cost $cost_p$ such that*

$$cost_p - 1 \leq OPT_p \leq cost_p.$$

In our algorithm for bin packing with rejection, if we replace the APTAS for bin packing in [9] with the one for variable bin packing in [11], then we get an algorithm for the variable-sized bin packing with rejection. We can see Lemmas 4, 7 and 8 still hold for the variable-sized case. Using the similar proof in Theorem 1, we can prove the following theorem.

Theorem 2. *There is an APTAS for variable-sized bin packing with rejection in time $O(n^{\epsilon^{-2}})$.*

4 Concluding Remarks

In this paper, we give a fast APTAS for bin packing with rejection and show the approach can be extended to variable-sized bin packing with rejection. The existence of an *asymptotic fully polynomial time approximation scheme* for bin packing with rejection is open.

References

1. Bartal, W., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., Stougie, L.: Multiprocessor scheduling with rejection. SIAM Journal on Discrete Mathematics 13(1), 64–78 (2000)
2. Chekuri, C., Khanna, S.: A polynomial time approximation scheme for the multiple knapsack problem. SIAM J. Comput. 35(3), 713–728 (2005)
3. Coffman, E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: a survey. In: Hochbaum, D. (ed.) Approximation algorithms for NP-hard problems, pp. 46–93. PWS Publishing, Boston (1996)
4. Dósa, G., He, Y.: Bin packing problems with rejection penalties and their dual problems. Information and Computation 204(5), 795–815 (2006)
5. Epstein, L.: Bin packing with rejection revisited. In: Erlebach, T., Kaklamanis, C. (eds.) WAOA 2006. LNCS, vol. 4368, pp. 146–159. Springer, Heidelberg (2007)
6. de la Vega, W.F., Lueker, G.: Bin packing can be solved within $1+\varepsilon$ in linear time. Combinatorica 1, 349–355 (1981)
7. Friesen, D.K., Langston, M.A.: Variable sized bin packing. SIAM Journal on Computing 15(1), 222–230 (1986)
8. Hoogeveen, H., Skutella, M., Woeginger, G.: Preemptive scheduling with rejection. Mathematical Programming, Ser. B 94(2-3), 361–374 (2003)
9. Karmarkar, N., Karp, R.M.: An efficient approximation scheme for the one-dimensional bin-packing problem. In: Proc. 23rd Annual IEEE Symp. Found. Comput. Sci., pp. 312–320. IEEE Computer Society Press, Los Alamitos (1982)
10. Kellerer, H.: A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem. In: Hochbaum, D.S., Jansen, K., Rolim, J.D.P., Sinclair, A. (eds.) RANDOM 1999 and APPROX 1999. LNCS, vol. 1671, pp. 51–62. Springer, Heidelberg (1999)
11. Murgolo, F.D.: An efficient approximation scheme for variable-sized bin packing. SIAM Journal on Computing 16(1), 149–161 (1987)

Online Coupon Consumption Problem

Yiwei Jiang[1,2,3,⋆], An Zhang[1], and Zhiyi Tan[1,⋆⋆]

[1] Department of Mathematics, State Key Lab of CAD & CG,
Zhejiang University, Hangzhou 310027, P.R. China
[2] College of Science, Zhejiang Sci-tech University,
Hangzhou 310018, P.R. China
[3] Key Laboratory of Advanced Textile Materials and Manufacturing Technology,
Ministry of Education, Zhejiang Sci-tech University
tanzy@zju.edu.cn

Abstract. Nowadays, it is popular that the dealer makes profits by selling a kind of discount coupons, which can be used as money to purchase commodities with total cost less than or equal to the face value of the coupon. We can purchase a coupon at a price of $0 < s \leq 1$ times its face value and the number of potential purchasable coupons is a given integer l. The customer has the option to buy the goods by cash completely or by a discount coupon. However, each piece of goods can only use one coupon and the coupon used must have enough balance for the goods. The objective is to minimize the total cost for purchasing all the goods. In this paper, we reduce the problem to a special bin packing model. We consider the online problems for all $0 < s \leq 1$ and $1 \leq l \leq \infty$. We present optimal online algorithms for all $0 < s \leq 1$ when $l = \infty$ and $l = 1$. For $2 \leq l < \infty$, we give both a lower bound and an algorithm, and show the algorithm is optimal for $l = 2$.

1 Introduction

In the competitive world of business, the dealer makes lots of money by selling discount *coupons* which means that the customers can get a coupon at a price of $s \leq 1$ times its face value. We then call $1 - s$ the discount of a coupon. Each coupon can be used as money to purchase commodities with total cost less than or equal to the face value of the coupon. For convenience, we set this value to be 1 among the paper. Therefore each coupon costs exactly s. The customer has the option to buy the goods by cash completely or by a discount coupon. However, for the dealer, to maximize his profit, only one coupon is allowed to use for each piece of goods and the coupon used must have enough balance for the goods. Clearly, it is not always good to purchase goods by coupons. For example, when the total cost of all goods the customer wants is only $s/2$, it is better to purchase the goods by cash rather than by coupons. We should decide

⋆ Supported by Natural Science Foundation of Zhejiang Province (Y605316).
⋆⋆ Corresponding author. Supported by Natural Science Foundation of China (10671177, 60021201).

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 219–230, 2007.

to purchase the goods whether by cash or by coupons in order to minimize the total cost. This problem can be reduced to such a special bin packing model: The cost of a piece of goods is compared to the size of an item. Each coupon can be regarded as a potentially purchasable bin with a certain capacity, and there are at most l ($1 \leq l \leq \infty$) bins purchasable and no such bins provided initially. When a piece of goods is bought by cash, it can be regarded as an item rejected, in which a penalty identical to its size should be paid. On the other hand, those goods bought by a coupon are regarded as items packed into a bin. Hence, to minimize the total cost of all goods is equivalent to minimize the sum of the penalty of the rejected items(i.e., the cost of the goods purchased by cash) and s times the number of purchased bins (i.e., the cost of purchased coupons). We call this problem the *coupon consumption problem*.

The formally description of the problem is the following. We are given a sequence $\mathcal{J}$ of independent items with positive size $p_1, p_2, \ldots, p_n$, which must be rejected or be packed into one of the purchased bins. At most l, $1 \leq l \leq \infty$, bins $B_1, B_2, \cdots, B_l$ can be purchased. Each bin has a unit-capacity, i.e., only items with total size not more than 1 can be packed into it. As long as there exists an item packed into B_i, $1 \leq i \leq l$, B_i is *purchased*, which costs $s \leq 1$. If an item p_i is, however, rejected, one should pay its penalty p_i. The objective is to minimize the sum of the penalty of all rejected items and all the cost for purchasing bins.

We consider online version of the problem in this paper, and using the *competitive ratio* to measure the performance of an online algorithm. For an item sequence $\mathcal{J}$ and an algorithm A, let $W^A(\mathcal{J})$ (or shortly W^A) denote the objective value produced by A and let $W^*(\mathcal{J})$ (or shortly W^*) denote the optimal objective value in an offline version. Then the competitive ratio of A is defined as the smallest number C such that for any $\mathcal{J}$, $W^A(\mathcal{J}) \leq CW^*(\mathcal{J})$. An online problem has a *lower bound* ρ if no online algorithm has a competitive ratio smaller than ρ. An online algorithm is called *optimal* if its competitive ratio matches the lower bound.

There are some problems in the literature have relations with our problem. However, to the authors knowledge, coupon consumption problem has some different features and is a new one never discussed before.

Ski-rental problem ([5])**:** For this problem, it is well known that an optimal algorithm exists with competitive ratio 2. It is assumed that once the ski is purchased, it can be used unlimitedly. But for the coupon consumption problem, the coupon has a limited face value. Moreover, we can purchase more than one coupon when $l > 1$ whereas in the ski-rental problem we only need to purchase one pair of skis.

Bin-packing with rejection([3], [4])**:** In this problem, we are given a set of items and a number of unit-capacity bins. Each item has a size and a penalty. An item can be either rejected or packed into one bin without exceeding the bin capacity. The objective is to minimize the sum of used bins and the total penalties of all rejected items. However, the number of bins can be used is infinite while the number of the purchasable coupons in our problem is at most l. Even

for the case of $l = \infty$ in our problem, there still exists difference between the two problems. The size and penalty of each item are given independently in the bin-packing with rejection problem, whereas in our problem the ratio of size and penalty is fixed.

In this paper, we consider the online coupon consumption problem with l potentially purchasable coupons. We present optimal algorithms for all $0 < s \leq 1$ for the cases of $l = \infty$ and $l = 1$. For the case of $2 \leq l < \infty$, both a lower bound and an online algorithm are given. Moreover, the algorithm is optimal when $l = 2$.

The rest of the paper is organized as follows. Section 2 gives some basic notation and preliminary results. Section 3 and 4 consider the problem with special cases $l = \infty$ and $l = 1$, respectively. Section 5 considers the $2 \leq l < \infty$ case of the problem. Finally, Section 6 presents some concluding remarks.

2 Preliminary

To simplify the presentation, the following notation and definitions are required in the remainder of the paper.

$-k(\leq l)$	The number of purchased bins by an online algorithm A.
$-k^*(\leq l)$	The number of purchased bins in the optimal solution.
$-P_j$	The total size of the first j items.
$-L_j^i$	The content of bin B_i at time $j \geq 0$ (i.e., the moment right after the j-th item is rejected or packed) in the algorithm A, $i = 1, \cdots, l$.
$-L_j^0$	The total penalty of the rejected items at time $j \geq 0$.
$-L_n^*$	The total penalty of the rejected items in the optimal solution after all items have been rejected or packed.
$-W^A$	The objective function value yielded by A. It clearly follows that $W^A = L_n^0 + sk$.
$-W^*$	The optimal objective function value. Hence $W^* = L_n^* + sk^*$.

Also, an item p_j is said to *fit in* a bin if the curent content of the bin does not exceed $1 - p_j$. Let $\mathcal{J}' = \{p_i | p_i \in \mathcal{J}, p_i > 1\} \subseteq \mathcal{J}$ and $P' = \sum_{p_i \in \mathcal{J}'} p_i$. The following theorem shows that we only need to design algorithms for $\mathcal{J} \setminus \mathcal{J}'$.

Theorem 1. *If there is an algorithm A satisfying $\frac{W^A(\mathcal{J} \setminus \mathcal{J}')}{W^*(\mathcal{J} \setminus \mathcal{J}')} \leq C$, then $\frac{W^A(\mathcal{J})}{W^*(\mathcal{J})} \leq C$.*

Proof. As each bin has only a capacity of 1 and no item can be separated, all items in $\mathcal{J}'$ must be rejected regardless of the algorithm A or the optimal solution. Therefore, we can obtain that $W^A(\mathcal{J}) = W^A(\mathcal{J} \setminus \mathcal{J}') + P'$ and $W^*(\mathcal{J}) \geq W^*(\mathcal{J} \setminus \mathcal{J}') + P'$, which yields that

$$\frac{W^A(\mathcal{J})}{W^*(\mathcal{J})} \leq \frac{W^A(\mathcal{J} \setminus \mathcal{J}') + P'}{W^*(\mathcal{J} \setminus \mathcal{J}') + P'} \leq \frac{W^A(\mathcal{J} \setminus \mathcal{J}')}{W^*(\mathcal{J} \setminus \mathcal{J}')} \leq C.$$

□

Hence, in the remainder of this paper, we only need to consider the item set $\mathcal{J} \setminus \mathcal{J}'$. In other words, we can assume that the size of each item in $\mathcal{J}$ does not exceed 1.

Denote by $\lfloor x \rfloor$ the largest integer number not greater than x. It is easy to obtain some properties on the optimal objective value W^* as follows.

Lemma 1. *The optimal objective value W^* satisfies:*
(1) *If $0 < P_n \leq 1$, then $W^* = \min\{P_n, s\}$.*
(2) *If $j + z < P_n \leq j + 1$ for $0 \leq z < 1$ and some integer j, $0 \leq j \leq l - 1$, then $W^* \geq js + \min\{s, z\}$.*
(3) *If $P_n > l$, then $W^* \geq P_n - l + ls$.*
(4) $W^* \geq \min\{P_n - \lfloor P_n \rfloor, s\} + \lfloor P_n \rfloor s \geq P_n s$.

3 Problem with $l = \infty$

In this section, we present an optimal algorithm for the case that the number of coupons that customer can purchase is unlimited, i.e., $l = \infty$.

Denote

$$\alpha = \min\{2, \frac{1}{s}\} = \begin{cases} 2, \text{ if } 0 < s \leq \frac{1}{2}, \\ \frac{1}{s}, \text{ if } \frac{1}{2} < s \leq 1. \end{cases}$$

We partition our algorithm $A1$ into two parts according to the value of s as follows. A well-known algorithm for bin packing problem, namely *First Fit* (FF), which packs each item into the lowest indexed bin in which it fits, will be used in step 4.

Algorithm $A1$
Part 1(For $\frac{1}{2} < s \leq 1$)
Reject all items.
Part 2(For $s \leq \frac{1}{2}$)
1. Let $L_0^0 = 0$ and $j = 0$.
2. If no new item arrives, stop. Otherwise, set $j = j + 1$.
3. If $P_j \leq s$, then reject p_j. Return step 2.
4. If $P_j > s$, pack p_j by FF, Return step 2.

Suppose $A1$ has purchased k bins in the order of $B_1, B_2, \cdots, B_k$, thus we have a lemma below.

Lemma 2. ([2], [1]) *If $k \geq 2$, then $\sum_{i=1}^{k} L_n^i > \frac{k}{2}$.*

Theorem 2. *The algorithm $A1$ has a competitive ratio of α.*

Proof. We prove it by distinguishing the value of s as follows.

For $\frac{1}{2} < s \leq 1$, as Part 1 states that all items are rejected without purchasing any bin, we have $W^A = L_n^0 = P_n$. Lemma 1 (4) shows that $W^* \geq sP_n$, thus we have $\frac{W^A}{W^*} \leq \frac{1}{s}$.

For $s \leq \frac{1}{2}$, if $P_n \leq s$, then the algorithm also rejects all items by step 3 of Part 2. Therefore, we have $W^A = P_n \leq s$ and thus $W^* = P_n = W^A$ from Lemma 1 (1). If $P_n > s$, then $L_n^0 \leq s$ and the algorithm must purchase some bins. If $A1$ purchases only one bin, i.e., $k = 1$, then we have $W^A = s + L_n^0 \leq 2s$ and $s < P_n \leq s+1$ which implies that $W^* \geq s$ by Lemma 1 (2). It follows that $\frac{W^A}{W^*} \leq 2$. Finally, if $k \geq 2$, we have $W^A = ks + L_n^0$. From Lemma 2, we can obtain that $P_n \geq \sum_{i=0}^{n} L_n^i \geq \frac{k}{2} + L_n^0$. Furthermore, if k is an even, then $W^* \geq \frac{k}{2}s + L_n^0$ due to Lemma 1 (2) and thus $\frac{W^A}{W^*} \leq \frac{ks+L_n^0}{\frac{k}{2}s+L_n^0} < 2$. On the other hand, if k is an odd, then we have $P_n \geq \frac{k-1}{2} + \frac{1}{2} + L_n^0$, resulting in $W^* \geq \frac{k-1}{2}s + \min\{s, \frac{1}{2} + L_n^0\} = \frac{k+1}{2}s$ by Lemma 1 (2). It yields that $\frac{W^A}{W^*} \leq \frac{ks+L_n^0}{\frac{k+1}{2}s} \leq \frac{ks+s}{\frac{k+1}{2}s} = 2$. □

The following theorem shows that $A1$ is an optimal algorithm.

Theorem 3. *The competitive ratio of any algorithm for the problem with $l = \infty$ is at least α.*

Proof. Assume that an algorithm A exists with competitive ratio C. Let the size of each arriving item always be ϵ, a sufficiently small positive number, as long as the algorithm A does not purchase any bin for the previous items and the current total size (denoted by P) is less than 1. If the algorithm A purchases the first bin B_1 when $P < 1$, i.e., the total penalty of rejected items and the current content of B_1 are $P - \epsilon$ and ϵ, respectively, no new item arrives. Thus we have $W^A = s + P - \epsilon$, while the optimal cost is $W^* = \min\{P,\ s\}$ due to Lemma 1 (1). It follows that

$$C = \frac{W^A}{W^*} \geq \frac{s + P - \epsilon}{\min\{P,\ s\}} \geq 2 - \frac{\epsilon}{\min\{P,\ s\}} \to 2.$$

On the other hand, if the algorithm A always rejects items even when $1 - \epsilon \leq P < 1$, then no new item arrives and thus we have $W^A = P \geq 1 - \epsilon$, which yields that $C = \frac{W^A}{W^*} \geq \frac{1-\epsilon}{s} \to \frac{1}{s}$ with $W^* \leq s$. Hence, we conclude that for any algorithm A, $\frac{W^A}{W^*} \geq \alpha$. □

It is not hard to see that Theorem 3 holds for any $1 \leq l \leq \infty$, that is to say, the lower bound of the problem for any $1 \leq l \leq \infty$ is at least α. Note that for $\frac{1}{2} < s \leq 1$, Part 1 of $A1$ rejects all the items without purchasing any bins. Hence, it remains optimal for the case of $1 \leq l < \infty$. In other words, the customers need not purchase coupons if its discount is less than 50%. Hence, we only need to consider the case of $0 < s \leq \frac{1}{2}$ in the remainder of this paper.

4 Problem with $l = 1$

In this section, we present an optimal algorithm for the problem with $l = 1$, namely, the customer can purchase only one coupon B_1. We only consider the case of $0 < s \leq \frac{1}{2}$.

Denote by $x = \frac{\sqrt{9s^2+8s}-3s}{4}$ the larger solution to the equation of $\frac{s+1}{s+x} = \frac{s+2x}{s}$ with respect to x. Let $\beta = \frac{s+1}{s+x} = \frac{s+2x}{s} = \frac{\sqrt{9s^2+8s}-s}{2s}$. Note that

$$s + 2x \leq 1 \tag{1}$$

and

$$2x \geq s \tag{2}$$

when $0 < s \leq \frac{1}{2}$.

Theorem 4. *The competitive ratio of any algorithm for the problem with $l = 1$ is at least β.*

Proof. Assume that there exists an online algorithm A with competitive ratio C. The first item $p_1 = x$ arrives. If the algorithm purchases B_1 to pack it, then the second and last item $p_2 = 1$ arrives. Clearly, it has no choice but to reject, resulting in $W^A = s + 1$. However, the optimal solution is to reject p_1 and purchase B_1 for p_2, i.e., $W^* = s + x$. It follows that $C \geq \frac{W^A}{W^*} \geq \frac{s+1}{s+x} = \beta$. On the other hand, if A rejects p_1 initially, instead of purchasing B_1. Then the second item $p_2 = x + \epsilon$ arrives, where ϵ is a sufficiently small positive number. If A now purchases B_1 for packing p_2, then the third and last item $p_3 = 1 - x$ arrives, which must be rejected because of $p_2 + p_3 > 1$. It yields that $W^A = s + 1$ while we can obtain $W^* = s + x + \epsilon$ by rejecting p_2 and purchasing the bin for p_1 and p_3, and thus $C \geq \frac{W^A}{W^*} \geq \frac{s+1}{s+x+\epsilon} \to \beta(\epsilon \to 0)$. If A also rejects p_2, then let the size of each subsequently arriving item be ϵ, as long as A does not purchase B_1 and the current total size (denoted by P) is less than $s + 2x$. If A purchases B_1 when $P < s + 2x$, then no more item arrives. We have $W^A \geq (p_1 + p_2) + s = 2x + s + \epsilon$. If A does not purchase B_1 even when $s + 2x - \epsilon < P \leq s + 2x$, then no new item arrives. We have $W^A > s + 2x - \epsilon$. For both cases, we have $W^* \leq s$ by (1) and Lemma 1(1), thus $C \geq \frac{W^A}{W^*} \geq \frac{s+2x-\epsilon}{s} \to \beta(\epsilon \to 0)$. □

Algorithm $A2$

1. Let $L_0^0 = L_0^1 = 0$ and $j = 0$.
2. If no new item arrives, stop. Otherwise, set $j = j + 1$.
3. If $P_j \leq 2x$, then reject p_j. Return step 2.
4. If $P_j > 2x$, we do:
   4.1. If $L_{j-1}^0 + p_j \leq 2x$, then reject p_j. Return step 2.
   4.2. If $L_{j-1}^0 + p_j > 2x$ and $L_{j-1}^1 + p_j \leq 1$, then pack p_j into B_1. Return step 2.
   4.3. If $L_{j-1}^0 + p_j > 2x$ and $L_{j-1}^1 + p_j > 1$, then reject p_j. Return step 2.

We now give a simple and useful lemma below before going to analyze $A2$.

Lemma 3. *Let a, b, c and d be positive numbers.*
(1) *If $\frac{a}{b} > 1$, then $\frac{a+c}{b+c} \leq \frac{a}{b}$*
(2) *If $\frac{a}{b} \leq \frac{c}{d}$, then $\frac{a+c}{b+d} \leq \frac{c}{d}$.*

Theorem 5. *The algorithm A2 has a competitive ratio of β. Thus it is an optimal algorithm for the problem with $l = 1$ and $0 < s \leq \frac{1}{2}$.*

Proof. If $P_n \leq 2x$, then $A2$ does not purchase the bin B_1 and thus $W^A = L_n^0 = P_n$. Clearly, $W^* \geq \min\{s, P_n\}$. By (2),

$$\frac{W^A}{W^*} \leq \frac{P_n}{\min\{s, P_n\}} \leq \max\{\frac{P_n}{s}, 1\} \leq \frac{2x}{s} < \frac{s+2x}{s} = \beta.$$

We then consider the case that $P_n > 2x \geq s$. Let $t = \min\{j | P_j > 2x\}$. We can conclude that $A2$ must purchase B_1 to pack p_t in step 4.2. It is clear that $W^* \geq \min\{s, P_n\} = s$. If $L_n^0 \leq 2x$, we have $\frac{W^A}{W^*} \leq \frac{s+L_n^0}{s} \leq \frac{s+2x}{s} = \beta$. Therefore, we are left to show the result for the case of $L_n^0 > 2x$, which implies that there must exist items to be rejected by step 4.3.

Let p_h be the last item rejected by step 4.3. We conclude that $h > t$ and $p_h + L_{h-1}^1 > 1$, resulting in

$$p_h + L_n^1 > 1. \tag{3}$$

Let $\mathcal{J}^1 = \{p_i | p_i \text{ is packed into } B_1, i < h\}$. $\mathcal{J}^1 \neq \emptyset$ since $p_t \in \mathcal{J}^1$. Moreover, for any item $p_j \in \mathcal{J}^1$, we have $p_j + L_{j-1}^0 > 2x$. Since $L_n^0 - p_h = L_{h-1}^0 \geq L_{j-1}^0$ with $h > j$ and $L_n^1 \geq \sum_{p_i \in \mathcal{J}^1} p_i \geq p_j$, we have

$$L_n^1 + L_n^0 - p_h \geq p_j + L_n^0 - p_h > 2x. \tag{4}$$

To get the desired result, two cases are considered as follows.

Case 1. $L_n^0 - p_h > L_n^1$. Together with (4), we have

$$L_n^0 - p_h > x, \tag{5}$$

and

$$\frac{L_n^0 - p_h - L_n^1}{L_n^0 - p_h - x} \leq 2 \leq \frac{s+1}{s+x}. \tag{6}$$

By Lemma 1(3) with $l = 1$, we obtain $W^* \geq P_n - 1 + s = L_n^0 + L_n^1 - 1 + s$. Hence, by Lemma 3 and (3),(6), we obtain that

$$\begin{aligned} \frac{W^A}{W^*} &\leq \frac{L_n^0 + s}{L_n^0 + L_n^1 - 1 + s} = \frac{(p_h + L_n^1 - 1) + (L_n^0 - p_h - L_n^1) + (1+s)}{(p_h + L_n^1 - 1) + (L_n^0 - p_h - x) + (x+s)} \\ &\leq \frac{(L_n^0 - p_h - L_n^1) + (s+1)}{(L_n^0 - p_h - x) + (s+x)} \leq \frac{s+1}{s+x} = \beta. \end{aligned} \tag{7}$$

Case 2. $L_n^0 - p_h \leq L_n^1$. Combining it with (4), we obtain $L_n^1 > x$. Furthermore, if $L_n^0 - p_h > x$, from (7) and Lemma 3 (1), we have

$$\frac{W^A}{W^*} \leq \frac{(L_n^0 - p_h - L_n^1) + s + 1}{(L_n^0 - p_h - x) + s + x} < \frac{(L_n^0 - p_h - x) + s + 1}{(L_n^0 - p_h - x) + s + x} \leq \frac{s+1}{s+x} = \beta.$$

Then we assume that $L_n^0 - p_h \leq x$, resulting in $p_h > x$ with $L_n^0 > 2x$, and $p_j > x$ with (4) for any $p_j \in \mathcal{J}^1$. Since $p_h + \sum_{p_i \in \mathcal{J}^1} p_i = p_h + L_{h-1}^1 > 1$, it

is impossible to pack all items in $\mathcal{J}^1 \bigcup \{p_h\}$ into B_1 in any optimal solution. That is to say, at least one item $p_g \in \mathcal{J}^1 \bigcup \{p_h\}$ is rejected. Therefore, with $P_n = L_n^0 + L_n^1 > 2x + x \geq s + x$, we have $W^* \geq \min\{s + p_g, P_n\} > s + x$. If $L_n^0 \leq 1$, it is trivial that $\frac{W^A}{W^*} \leq \frac{s+1}{s+x} = \beta$. On the other hand, if $L_n^0 > 1$, then by Lemma 1 (3), we have $W^* \geq L_n^0 + L_n^1 - 1 + s$ and thus

$$\frac{W^A}{W^*} \leq \frac{L_n^0 + s}{L_n^0 + L_n^1 - 1 + s} = \frac{(L_n^0 - 1) + 1 + s}{(L_n^0 - 1) + L_n^1 + s} \leq \frac{1 + s}{L_n^1 + s} \leq \frac{s + 1}{s + x} = \beta$$

with Lemma 3(1) and $L_n^1 > x$.

Hence, the desired result is proved and the optimality of $A2$ is directly from Theorem 4. □

5 Problem with $2 \leq l < \infty$

In this section, we discuss the problem with $2 \leq l < \infty$. Lower bound and algorithm for all $2 \leq l < \infty$ are considered in subsection 5.1 and 5.2, respectively. Furthermore, in subsection 5.3, we reconsider the lower bound for the special case of $l = 2$ and show the algorithm is optimal for this case.

5.1 Lower Bound

We present a lower bound of the problem for all $2 \leq l < \infty$. Note that it is necessary to consider the case of $0 < s \leq \frac{1}{2}$.

Theorem 6. *The competitive ratio of any algorithm for the problem with* $2 \leq l < \infty$ *is at least*

$$\begin{cases} 1 + \frac{1}{(l+1)s}, & \text{if } 0 < s \leq \frac{1}{l+1}, \\ 2, & \text{if } \frac{1}{l+1} < s \leq \frac{1}{2}. \end{cases}$$

5.2 Algorithm *A3*

In this subsection, we present an algorithm $A3$ for all $2 \leq l < \infty$ as follows.

Algorithm $A3$

1. Let $L_0^0 = 0$ and $j = 0$.
2. If no new item arrives, stop. Otherwise, set $j = j + 1$.
3. If $L_{j-1}^0 + p_j \leq \frac{1}{2}$, then reject p_j. Return step 2.
4. If $L_{j-1}^0 + p_j > \frac{1}{2}$, pack p_j into the earliest purchased bin in which it fits. If such bin does not exist, we do:
   4.1. If the number of purchased bins is less than l, then purchase a new one to pack p_j. Return step 2.
   4.2. If all l bins have been purchased, then reject p_j. Return step 2.

Note that, in step 4, A3 is actually applying the FF algorithm to purchase bins and pack items. We can conclude that $L_n^0 \leq \frac{1}{2}$ unless there exist some items to be rejected by step 4.2, i.e. if $k < l$, then $L_n^0 \leq \frac{1}{2}$.

Theorem 7. *The competitive ratio of A3 is at most* $1+\frac{1}{2s}$.

Proof. It is clear that $W^A = P_n = L_n^0 \le \frac{1}{2}$ and $W^* = \min\{P_n, s\}$ if $A3$ does not purchase any bin, i.e., $k = 0$. Thus $\frac{W^A}{W^*} \le 1 + \frac{1}{2s}$. If $k = 1$, we can obtain $P_n = L_n^0 + L_n^1 > \frac{1}{2}$ and $L_n^0 \le \frac{1}{2}$. Thus we have $W^A \le s + \frac{1}{2}$ and $W^* \ge \min\{P_n, s\} = s$, which yields the desired competitive ratio. If $2 \le k < l$, then have $L_n^0 \le \frac{1}{2}$ and $P_n > \frac{k}{2} + L_n^0$ from Lemma 2. By Lemma 1 (4), we have $W^* \ge sP_n$. It follows that

$$\frac{W^A}{W^*} \le \frac{ks + L_n^0}{sP_n} \le \frac{ks + L_n^0}{(\frac{k}{2} + L_n^0)s} < 1 + \frac{1}{2s}$$

by $L_n^0 \le \frac{1}{2} < \frac{k}{2}$. Finally, we consider the case of $k = l$. If $L_n^0 \le \frac{k}{2}$, we can obtain the result by the similar argument above. Otherwise, $P_n > \frac{k}{2} + L_n^0 > \frac{k}{2} + \frac{k}{2} = k$. It follows that $W^* \ge P_n - k + ks$ from Lemma 1 (3). Furthermore, we can obtain $W^A \le ks + L_n^0 < ks + P_n - \frac{k}{2}$. Hence, we have $\frac{W^A}{W^*} \le \frac{ks + P_n - \frac{k}{2}}{P_n - k + ks} < 1 + \frac{1}{2s}$. □

5.3 Lower Bound for the Case of $l = 2$

In this subsection, we revisit the lower bound for the problem with $l = 2$ and thus show $A3$ is optimal for this case. For an arbitrary integer $m \ge 3$, we can show that the competitive ratio of any algorithm for the problem with $l = 2$ is at least $1 + \frac{m+1}{(2m+3)s}$ when $0 < s \le 1/2$.

Lemma 4. *The competitive ratio of any online algorithm A for the problem is at least* $1 + \frac{m+1}{(2m+3)s}$, *if any one of the following conditions occurs.*

($C1$) *At some time* $j < n$, $P_j \le 1$ *and A purchased both bins.*
($C2$) *At some time* $j < n$, $P_j \le 1$ *and* $L_j^0 \ge \frac{m+1}{2m+3}$.
($C3$) $W^A \ge 2s + \frac{2m+2}{2m+3}$ *and* $W^* \le 2s$.

Proof. For ($C1$), then the last item $p_n = 1$ arrives. Since both bins have been purchased at that time, p_n must be rejected. Thus we have $W^A \ge 1 + 2s$. It is easy to obtain that $W^* = 2s$ by purchasing one bin for p_n and the other bin for the first j items due to $P_j \le 1$. Hence, $\frac{W^A}{W^*} \ge 1 + \frac{1}{2s}$.

For ($C2$), if $P_j = 1$, then no new item arrives. If $P_j < 1$, then the last item $p_n = 1 - P_j$ arrives. No matter whether A purchase the bins or not, we have $W^A \ge \min\{P_n, L_j^0 + s\} \ge \min\{1, \frac{m+1}{2m+3} + s\} = \frac{m+1}{2m+3} + s$. Because $P_n = 1$, it is clear that $W^* \le s$ by purchasing one bin for all items. Thus we have $\frac{W^A}{W^*} \ge 1 + \frac{m+1}{(2m+3)s}$.

For ($C3$), it is trivial to obtain the result. □

Lemma 4 shows that we only need to construct the instances such that one of those conditions occurs. Before going to prove our main result, we give a useful lemma first. To simplify the presentation, in the remaining part of our paper, we denote by L_i the current content of B_i, $i = 1, 2$, also let L_0 be the current total penalty of rejected items and let p and q be the last two items.

Lemma 5. *If A has only purchased B_1 when the last two items p and q arrive, and the size of p and q satisfies that $p \geq q$, $L_1 + q > 1$ and $p + q > 1$, then we have $W^A \geq 2s + L_0 + q$.*

Proof. It is clear that none of the two items p and q can be packed to B_1 since $p \geq q$ and $L_1 + q > 1$. Note that $p \geq q$ and $p + q > 1$ implying $p > \frac{1}{2} \geq s$. Thus, if A does not purchase B_2, we have $W^A \geq s + L_0 + p + q > 2s + L_0 + q$. Otherwise, any algorithm can only pack one of p and q into B_2 due to $p + q > 1$. That is, at least one of p and q must be rejected, then $W^A \geq 2s + L_0 + \min\{p, q\} = 2s + L_0 + q$. □

We call an item with size $\frac{1}{2m+3}$ as a *tiny item*, and an item with size $\frac{3}{2(2m+3)}$ as a *small item*.

Theorem 8. *For the problem with $l = 2$ and $0 < s \leq \frac{1}{2}$, the competitive ratio of any algorithm is at least $1 + \frac{m+1}{(2m+3)s}$, where $m \geq 3$ is an arbitrary integer.*

Proof. We use adversary method to achieve the result by constructing the instances. The first two items p_1 and p_2 are both tiny items. If A purchases both bins for them, then clearly $(C1)$ is satisfied. Hence, we can assume that A purchases at most one bin, w.l.o.g, denote it by B_1 if it is purchased, for the first two items. Let a_0 and a_1 be the current number of items rejected and packed into B_1, respectively. Note that $a_1 = 0$ means that A does not purchase B_1 and thus no bins have been purchased.

If $a_0 = 0$ and $a_1 = 2$, that is, A purchases B_1 for the first two tiny items, i.e., $L_1 = \frac{2}{2m+3}$, then the last two items $p = q = \frac{2m+2}{2m+3}$. Note that $L_1 + q > 1$ and $p + q > 1$. From Lemma 5, we conclude that $W^A \geq 2s + L_0 + q = 2s + \frac{2m+2}{2m+3}$, while we have $W^* = 2s$ by packing p_1, p to B_1, and p_2, q to B_2. Hence, $(C3)$ is satisfied.

Now we are left to consider two cases: (i) $a_0 = 1$ and $a_1 = 1$; (ii) $a_0 = 2$ and $a_1 = 0$. We will construct the instances in the following such that

$$1 \leq a_0 = a_1 \leq m \tag{8}$$

unless A has satisfied one of $(C1)$ and $(C2)$. Clearly, (i) satisfies (8). Therefore, we only need to focus on (ii). In that case, let the tiny items arrive one by one. To avoid $(C1)$, A can only purchase at most one bin so long as the number of tiny items is not greater than $2m+3$. That is to say, the arriving item must be rejected alternatively or be packed to B_1. Note that initially we have $0 = a_1 < a_0 = 2$. If $1 \leq a_1 = a_0 \leq m$ at some time, then we are done. So we assume that $a_1 < a_0$ always holds till $a_0 = m$, then $m - a_1$ tiny items arrive. Since the current load of B_1 is $\frac{m}{2m+3}$, to avoid $(C1)$ and $(C2)$, the algorithm A must pack all of them into B_1, resulting in $a_1 = a_0 = m$. Thus we obtain (8). Two cases are considered according to the value of a_0 as follows.

Case 1. $a_1 = a_0 = m$. Both the rejected items and the items packed into B_1 are tiny with the number m, i.e., $L_0 = L_1 = \frac{m}{2m+3}$. Then the item $p_{2m+1} = \frac{1+\epsilon}{2m+3}$ arrives, where ϵ is a sufficiently small positive number. It must be packed to B_1

by A in order to avoid $(C1)$ and $(C2)$. Then we have $L_1 = \frac{m+1+\epsilon}{2m+3}$ and the last two items $p = \frac{m+3-\epsilon}{2m+3}$ and $q = \frac{m+2}{2m+3}$ arrive. Note that $p > q$, $p + q > 1$ and $L_1 + q > 1$. From Lemma 5, we have $W^A \geq 2s + L_0 + q = 2s + \frac{2m+2}{2m+3}$, while $W^* = 2s$ can be obtained by packing $m+1$ tiny items and q into B_1, $m-1$ tiny items and p, p_{2m+1} into B_2. $(C3)$ is satisfied.

Case 2. $a_1 = a_0 = u \leq m-1$. There must exist an unique integer $v > 0$ and an unique rational number $r \in [0, \frac{3}{2})$ such that

$$m + 1 - u = \frac{3}{2}v + r. \tag{9}$$

Then the small items of size $\frac{3}{2(2m+3)}$ arrive one by one. Since

$$2u\frac{1}{2m+3} + 2v\frac{\frac{3}{2}}{2m+3} = \frac{2m+2-2r}{2m+3} < 1$$

from (9), in order to avoid $(C1)$, A must reject the arriving small item alternatively or pack it to B_1 so long as the number of small items is not greater than $2v$. Let b_0 and b_1 be the numbers of small items rejected and packed into B_1, respectively. It is not hard to obtain that one of the following cases must happen: (I) $b_0 + 1 = b_1 \leq v$; (II) $b_0 = b_1 = v$.

In fact, we have $b_0 = b_1 = 0$ initially. Once $b_0 < b_1 \leq v$, there must be $b_1 = b_0 + 1$ at some time, i.e., (I) occurs. Therefore we only need to show (II) must happen if (I) does not occur. That is to say, $b_0 \geq b_1$ always holds till $b_0 = v$. If $b_0 = b_1 = v$, we are done. Then we assume $b_0 > b_1$ and $v - b_1$ small items arrive. Since

$$u\frac{1}{2m+3} + (v+1)\frac{\frac{3}{2}}{2m+3} = \frac{m+1}{2m+3} + \frac{\frac{3}{2}-r}{2m+3} > \frac{m+1}{2m+3},$$

A must pack these items into B_1 in order to avoid $(C1)$ and $(C2)$. Thus (II) occurs.

We will complete the proof after considering the above cases (I) and (II), only the proof of (I) will be given due to lack of space.

For (I), we have

$$L_0 = \frac{u}{2m+3} + \frac{\frac{3}{2}b_0}{2m+3} = \frac{u + \frac{3}{2}b_0}{2m+3}$$

and

$$L_1 = \frac{u}{2m+3} + \frac{\frac{3}{2}b_1}{2m+3} = \frac{u + \frac{3}{2}(b_0+1)}{2m+3}.$$

Then the last two item

$$p = \frac{2m + \frac{5}{2} - u - \frac{3}{2}b_0}{2m+3}$$

and

$$q = \frac{2m + 2 - u - \frac{3}{2}b_0}{2m+3}$$

arrive. It is easy to get that $p > q$, $p + q > 1$ and $L_1 + q > 1$. From Lemma 5, we obtain that

$$W^A \geq 2s + L_0 + q = 2s + \frac{2m+2}{2m+3}.$$

On the other hand, we can obtain $W^* = 2s$ by packing $u+1$ tiny items, b_0 small items and q on B_1, $u-1$ tiny items, b_0+1 small items and p on B_2. Hence, $(C3)$ is satisfied.

Corollary 1. *The algorithm A3 is optimal for the case of* $l = 2$ *and* $0 < s \leq 1/2$.

Proof. Theorem 8 shows that the lower bound of the problem with $l = 2$ is arbitrarily close to $1 + \frac{1}{2s}$ when m tends to infinity. Then the result follows from Theorem 7. □

6 Concluding Remarks

In this paper, we have proposed a new model for the so-called *coupon consumption problem*. We presented different online algorithms for different value of l, the number of the potential purchasable coupons with a discount of $1 - s$. For all $0 < s \leq 1$, we presented optimal algorithms for $l = \infty$ and $l = 1$, respectively. When $2 \leq l < \infty$, we gave both a lower bound and an algorithm, and showed the algorithm is optimal for the case of $l = 2$.

The results of this paper suggest a number of problems deserving further study. For the problem under consideration, it is quite reasonable to conceive that the lower bound of the problem with $3 \leq l < \infty$ is not less than that with $l = 2$. As the algorithm $A3$ has the same competitive ratio for all $2 \leq l < \infty$ and it is optimal for the case of $l = 2$, we may conjecture that $A3$ is optimal for all $2 \leq l < \infty$. Moreover, it is assumed that in this paper, each coupon has a same face value and discount, it can be very interesting to consider the problem with the coupons having different face value or discount.

References

1. Basse, S.: Computer algorithms: Introduction to design and analysis. Addison-Wesley, Reading (1988)
2. Coffman, E.G., Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing: A survey. In: Hochbaum, D. (ed.) Approximation algorithms for NP-hard problems, pp. 46–93. PWS Publishing (1997)
3. Dósa, G., He, Y.: Bin packing problems with rejection penalties and their dual problems. Information and Compution 204, 795–815 (2006)
4. Epstein, L.: Bin packing with rejection revisited. In: Erlebach, T., Kaklamanis, C. (eds.) WAOA 2006. LNCS, vol. 4368, Springer, Heidelberg (2007)
5. Fiat, A., Woeginger, G.: On-line Algohithms: The State of Art. LNCS, vol. 1442, pp. 1–12. Springer, Heidelberg (1998)

Application of Copula and Copula-CVaR in the Multivariate Portfolio Optimization

Manying Bai and Lujie Sun

School of Economics and Management, Beijing University of Astronautics and Aeronautics, F-100083, Beijing, People's Republic of China
baimy@buaa.edu.cn

Abstract. In this article we resort to the copula theory and CVaR measures in the portfolio management, using copula function and copula-CVaR to design the portfolio optimization. We initially apply the three-dimensional Archimedean copula in the empirical study. After estimating the multi-dimensional copula, we use Monte Carlo method to generate the scenarios for the calculation of portfolio's variance and CVaR. Then we apply the minimum of copula based standard variance and CVaR as the objective function of the portfolio programming. The multivariate demonstration indicates that the copula theory and copula based CVaR method does better in the portfolio management than the normal hypothesis.

Keywords: Copula; CVaR; Portfolio Optimization; GARCH.

1 Introduction

The theory of copula dates back to Scklar(1959)[1], but its application to the analysis of financial problems becomes a new and fast-growing field recently. There are quite a lot of works on the application of copula theory, including risk management[2,3], time series dependence[4], derivatives pricing and so on. Essentially, copula provides a straight-forward way to extend financial modeling from the usual joint normal hypothesis to more general joint distributions with any marginal ones. There are a lot of empirical studies with two-dimensional copula, however, the application of three-dimensional copula has not been worked out until now.

Conditional Value at Risk is the gradually used risk measure in current risk management. It has been shown that CVaR is a coherent risk measure which has many attractive characteristics including sub-additivity and convexity[5]. In particular, when CVaR restricted optimization problem is calculated by scenario simulation, it has an equivalent linear programming formulation and can be solved using linear programming methods[6]. The comparison between VaR and CVaR has already been studied in empirical analysis[7,8]. However, the use of copula in the CVaR has not yet been applied.

The purpose of this paper is to suggest the use of copula to perform a pretty analysis in the multivariate portfolio management theory. We focus on the

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 231–242, 2007.

copula based standard variance and copula-CVaR minimization problems for portfolio selection. This paper applies the empirical distribution for the marginal distributions, uses the maximum likelihood estimation (MLE) to estimate the parameters of Archimedean copula, and chooses the Kolmogorov-Smirnov test for the good-of-fit of the estimated copula function. After estimating the multi-dimensional copula, we use Monte Carlo method to generate the scenarios for the calculation of portfolio's variance and CVaR. Then we apply the minimum of standard variance and CVaR on the basis of copula as the objective function of the portfolio programming. In the empirical analysis, we consider three important stock indexes in global markets: Hong Kong HS index, Dow Jones industry index and Nikkei index. Under the generated series of return of the portfolio, we get series of optimal proportions and the "mean-standard variance" and "mean-CVaR" efficient frontiers. What's more, we creatively applies the three-dimensional Archimedean copula in the empirical study, including the estimation of the parameters, the good-of-fit test, and the Monte Carlo simulation using the three-dimensional copula. Besides this, the use of copula in the CVaR in this paper is also a contribution.

This paper is organized as follows. Section 2 discusses the definition of copula and the copula based Monte Carlo simulation method, and present the CVaR formulas for single variable and multi variables respectively. Section 3 presents the use of copula and copula-CVaR in the multivariate portfolio optimization. Section 4 is the empirical analysis.

2 Copula and CVaR

In modern financial analysis, evidence of non-normality of the distribution of financial return variables grows every year. The traditional financial risk management VaR approach and the Mean-Variance theory by Markowitz all depends on the normal hypothesis. Since the multivariate normal distribution is simply not a good model for the joint distribution of many financial variables, it leads us to find more appropriate multivariate models. Copula is just the suitable model, which provides an alternative to the normal specification among variables, and has gained increasing attention in the risk management, portfolio management, asset pricing and other applications.

2.1 Copula Function and It's Multivariate Monte Carlo Simulation

Copula function is the joint cumulative distribution function (cdf) of a pair of variables $x_1, x_2, \cdots, x_n$ with the marginal cdf $F_1(x_1), F_2(x_2), \cdots, F_n(x_n)$ respectively[9]. That is to say, the copula function $C(u_1, u_2, \cdots, u_n)$ satisfies $H(x_1, x_2, \cdots, x_n) = C(F_1(x_1), F_2(x_2), \cdots, F_n(x_n))$, where $H(x_1, x_2, \cdots, x_n)$ is the joint cumulative distribution function.

According to the definition of copula function, the use of copula allows us to overcome the issue of estimating the joint cdf by two parts:

i. Determining the marginal distributions $F_1(x_1), F_2(x_2), \cdots, F_n(x_n)$ which represent the distribution of each variable, and estimating their parameters.

The marginal cdf can be structured by the traditional time series model such as GARCH model, as well as the student-t distribution or empirical distribution;

ii. Determining the dependence structure of the variables $x_1, x_2, \cdots, x_n$, specifying a suitable copula function. The copula family contains the normal copula, t copula, Archimedean copula and mix copula, etc. Moreover, there are still more than 20 different kinds of Archimedean copula in the copula family.

Among the Archimedean copula, Clayton copula, Gumbel copula and Frank copula are the most popular copula with the following analytic expression:

Clayton copula: $C_C(u,v) = (u^{-\theta} + v^{-\theta} - 1)^{-\frac{1}{\theta}}$
Gumbel copula: $C_G(u,v) = \exp\{-[(-\ln u)^{\theta} + (-\ln v)^{\theta}]^{\frac{1}{\theta}}\}$
Frank copula: $C_F(u,v) = -\frac{1}{\theta}\ln[1 + \frac{(e^{-\theta u}-1)(e^{-\theta v}-1)}{e^{-\theta}-1}]$

In the empirical study, we can choose the appropriate copula suitable for the financial variable. The next step is to estimate the parameters of the chosen copula. This paper applies the empirical distribution $\hat{F}_i(x) = \frac{1}{T}\sum \mathbf{1}(X_{it} \le x)$ for the marginal cdf, uses the maximum likelihood estimation (MLE) for the parameters of Archimedean copula, and we choose the Kolmogorov-Smirnov for the good-of-fit test of estimated copula function.

As for the maximum likelihood estimation, we use the method of solving the extremum of the sample's joint pdf's multiplying value. The three-dimensional variable's joint pdf is $f(x_1, x_2, x_3) = c(u_1, u_2, u_3) \cdot f(x_1) \cdot f(x_2) \cdot f(x_3)$, where $f(x_1), f(x_2), f(x_3)$ are the pdfs of the variables x_1, x_2, x_3; and the function $c(u,v) = \partial C(u,v)/\partial u \partial v$ is the copula's density function. So the maximum likelihood function $L((x_1,y_1),(x_2,y_2),(x_3,y_3),\theta)$ can be denoted as

$$L = \prod f(x_1, x_2, x_3) = \prod c(u_1, u_2, u_3) \cdot f(x_1) \cdot f(x_2) \cdot f(x_3)$$

Through solving the extremum value, we can get the estimation of the copula's parameters. After structuring the copula model, the joint distribution containing all dependence structure can be described by the copula, through which the risk management and portfolio management will get better conclusions than the results under normal hypothesis.

After all parameters of the copula are known, the task is then to generate series of variables $x_1, x_2, \cdots, x_n$ whose joint cdf is $C(u_1, u_2, \cdots u_n)$, where the uniformly distributed variables $u_1, u_2, \cdots u_n$ equals the marginal cdf $F_1(x_1)$,$F_2(x_2)$, $\cdots$,$F_n(x_n)$ of random variables $x_1, x_2, \cdots, x_n$ respectively.

Considering the three-dimensional simulation, we simulate the first two series of variables $(u_{1i}, u_{2i})_n$ through the two-dimensional copula $C(u_1, u_2)$, where n is the number of the simulation samples. Then the third series of variables $(u_{3i})_n$ will be generated though the three-dimensional copula $C(u_1, u_2, u_3)$.

In order to reach this goal, we use the method of conditional distributions. Let C_{u_1} denotes the conditional distribution function for the random variable u_2 at a given value u_1 of the two-dimensional copula $C(u_1, u_2)$, with the following expression $C_{u_1}(u_2) = \partial C(u_1, u_2)/\partial u_1$. In addition, C_{u_1} are non-decreasing almost everywhere on $[0, 1]$. Consequently, the function $u_2 \to C_{u_1}(u_2) = \partial C(u_1, u_2)/\partial u_1$ is uniformly distributed on $[0, 1]$.

Just as the same for three-dimensional copula $C(u_1, u_2, u_3)$, the two steps conditional distribution function $u_3 \to C_{u_1,u_2}(u_3) = \partial C(u_1, u_2, u_3) \diagup \partial u_1 \partial u_2$ is also uniformly distributed on $[0, 1]$.

For the sake of simplicity, we can now use the method of variable transformation to generate the desired series of $(u_{1i}, u_{2i}, u_{3i})_n$. The transformation consists of three steps:

i. Generate three independent uniform variables $u_1, \omega, \pi \in [0, 1]$. Here u_1 is already the first variable we are looking for;

ii. Let the conditional distribution $C_{u_1}(u_2) = \omega$, and compute the inverse function of $C_{u_1}(u_2)$. So the variable $u_2 = C_{u_1}^{-1}(\omega)$ is the second variable we are looking for;

iii. After getting the first two variable u_1, u_2, let the two steps conditional distribution $C_{u_1,u_2}(u_3) = \pi$. Calculate the inverse function of $C_{u_1,u_2}(u_3)$. Then the third variable will be generated by the inverse expression $u_3 = C_{u_1,u_2}^{-1}(\pi)$.

According to the above three steps, we generate three series of variables, which are the cdf value of the original variables $(u_{1i}, u_{2i}, u_{3i})_n$. To get the wanted variables, we have $x_1 = F_1^{-1}(u_1), x_2 = F_2^{-1}(u_2), x_3 = F_3^{-1}(u_3)$.

2.2 CVaR for Single and Multi Variables

Value at Risk, or VaR in short, is a popular measure of risk in nowadays financial risk management. However, it suffers from being unstable and difficult to calculate through scenarios with normal distribution. Moreover, a very serious shortcoming of VaR is that it doesn't satisfy the characteristic of sub-additivity and convexity when non-elliptical distributions are considered, which makes it inappropriate for portfolio optimization.

Conditional Value at Risk (CVaR) is an alternative measure to replace VaR. CVaR is intuitively defined as the weighted average of VaR and the expected losses that are strictly greater than VaR. Consequently CVaR provides an upper bound for VaR. However, CVaR is a convex function and suitable for scenario calculation. Rockafellar and Uryasev (2000) have shown that CVaR can be minimized using linear programming techniques, which allows CVaR a more effective for portfolio management than VaR. Because a CVaR constraint is tighter than a VaR constraint when the CVaR and VaR bounds coincide, these portfolio choice results are also true and to a greater extent if a CVaR constraint is imposed.

Considering the return variable r of the financial asset, note the loss variable $x = -r$. Then the VaR & CVaR calculated are all based of the loss variable x. Assume $\beta - VaR$ is the VaR at a confidence level $\beta \in [0, 1]$, CVaR is defined as the conditional expected value: $CVaR = E(x \mid x \geq \beta - VaR)$, which can be obtained from the following expression

$$CVaR = E(x \mid x \geq \beta - VaR) = \frac{1}{1-\beta} \int_{\beta - VaR}^{+\infty} x f(x) dx$$

where $f(x)$ is the probability density function (pdf) of loss variable x.

For multivariate portfolio, consuming that the multi loss of the financial assets is the vector $(x_1, x_2, \cdots, x_n)$. The vector $(\omega_1, \omega_2, \cdots, \omega_n)$ is the proportions of the loss variables., where we have $\sum_{i=1}^{n} \omega_i = 1$. Then the loss of the portfolio denotes $x_p = \sum_{i=1}^{n} \omega_i x_i$, so the CVaR of the portfolio can be calculated by the following expression

$$CVaR = \min_{\alpha \in R} \left\{ \alpha + \frac{1}{1-\beta} \iint_{x_i \in R} \left[\sum_{i=1}^{n} \omega_i x_i - \alpha \right]^+ f(x_1, x_2, \cdots, x_n) dx_1 dx_2 \cdots dx_n \right\}$$

where $f(x_1, x_2, \cdots, x_n)$ is the joint pdf of the multi variables $(x_1, x_2, \cdots, x_n)$, and $[x - \alpha]^+ = \max(x - \alpha, 0)$.

Through scenario simulation $(x_{1j}, x_{2j}, \cdots, x_{nj})_{j=1:m}$, we will get the CVaR from the linear programming expression

$$CVaR = \min_{\alpha \in R} \left\{ \alpha + \frac{1}{m(1-\beta)} \sum_{i=1}^{m} \left[\sum_{i=1}^{n} \omega_i x_{ij} - \alpha \right]^+ \right\}$$

where n is the number of loss variables, and m is the number of scenario samples. When n equals one, we obtain CVaR expressions for single variable situation.

The scenario can be gained by several ways. Adopting the history data directly is indeed a good manner. However, this paper applies the copula based Monte Carlo simulation to gain numbers of scenarios, which we called copula-CVaR method.

3 Copula and Copula-CVaR in the Portfolio Optimization

Portfolio optimization is one of the most attracting areas in decision-making. The mean-variance formulation by Markowitz in 1950s paved a foundation for modern portfolio selection analysis[10]. On the basis of Markowitz, Tobin(1958) added the risk-free asset to the portfolio[11].

The advantage of using the variance for describing portfolio risk is principally due to the simplicity of the computation, but the variance is not a satisfactory measure due to the symmetry. The classical portfolio management theory supposes that the multivariate joint distribution of the return variables is joint normal distribution, however it is either not a good model for the real financial return variables.

In order to deal with the variables not elliptically distributed and to measure the risk more accurately, copula can be applied in the portfolio optimization by virtue of expressing the variances without normal hypothesis. In order to express the variance of the portfolio by copula, we calculate the variance in scenarios analytically through the copula based simulation.

The multivariate joint pdf can be denoted by the copula function, which is given by $f(x_1, x_2, \cdots, x_n) = c(u_1, u_2, \cdots, u_n) \cdot \prod_{i=1}^{n} f(x_i)$.

where $c(u_1, u_2, \cdots, u_n) = \frac{\partial C(u_1, u_2, \cdots, u_n)}{\partial u_1 \partial u_2 \cdots \partial u_n}$ is the pdf of copula in formula, the function $f(x_i)$ denotes the i^{th} variable's pdf. Let $f(\mathbf{r}) = f(r_1, r_2, \cdots, r_n)$, then the variance of the portfolio can be denoted by

$$\sigma_p^2 = \int_{-\infty}^{+\infty} [r_p - E(r_p)]^2 f(r_p) dr_p = \oint \left[\sum_{i=1}^{n} \omega_i r_i - E(r_p) \right]^2 f(\mathbf{r}) dr_1 dr_2 \cdots dr_n$$

To avoid the complex analytical calculation, the discrete samples can be used in the following expression

$$\sigma_p^2 = \frac{1}{m} \sum_{j=1}^{m} \left[\sum_{i=1}^{n} \omega_i r_i - E(r_p) \right]^2 = \frac{1}{m} \sum_{j=1}^{m} \left(\omega_1 r_{1j} + \omega_2 r_{2j} + \cdots + \omega_n r_{nj} - E(r_p) \right)^2$$

where n is the number of loss variables, and m is the number of discrete samples, which can be gained by the copula based Monte Carlo simulation.

Using the above analytical calculating formulation, the copula based optimization problem of a portfolio manager is the following one:

$$\min \sigma_p^2 = \frac{1}{m} \sum_{j=1}^{m} \left[\sum_{i=1}^{n} \omega_i r_i - \sum_{i=1}^{n} \omega_i E(r_i) \right]^2$$
$$s.t. E(r_p) = \sum_{i=1}^{n} \omega_i E(r_i) = \kappa, \sum_{i=1}^{n} \omega_i = 1$$

where κ is the portfolio target return.

By giving series of target returns κ, the mean-variance efficient frontier will be gained. Considering the added risk-free asset, in addition, the optimal proportions can be calculated by the following programming formula

$$\max \frac{E(r_p) - r_f}{\sigma_p} = \frac{E(r_p) - r_f}{\sqrt{\frac{1}{m} \sum_{j=1}^{m} \left[\sum_{i=1}^{n} \omega_i r_i - E(r_p) \right]^2}}$$
$$s.t. \sum_{i=1}^{n} \omega_i = 1 \omega_i > 0$$

As already suggested by Markowitz(1959), other risk measures can be used in the mean-risk approach, such as VaR and CVaR[12]. By virtue of convexity and suitable for scenario calculation, CVaR is better than VaR approach in the portfolio management.

The mean-CVaR programming formula of the portfolio optimization is as follows

$$\min CVaR; s.t. E(x_p) = -\kappa, \sum_{i=1}^{n} \omega_i = 1, \omega_i > 0, \alpha \in R$$

where κ is the portfolio target return, so $-\kappa$ denotes the target loss.

Giving series of target losses $-\kappa$, the mean-CVaR efficient frontier will be gained by series of optimizations. In addition, The optimal proportions under risk-free asset can be calculated by the following programming

$$\max \frac{E(r_p) - r_f}{CVaR}; s.t. \sum_{i=1}^{n} \omega_i = 1, \omega_i > 0$$

4 Empirical Analysis

After above theoretical analysis, we apply the minimum of copula based standard variance and CVaR as the objective function of the portfolio programming. In the empirical study, we consider three important stock indexes in global markets (Hong Kong HS index, Dow Jones industry index, and Nikkei index), with 488 daily data samples between May the 6th 2004 till May the 23th 2006. The risk-free asset which we choose is five years' Chinese public debt with the year rate 0.0214. The initial value of the portfolio is 1 unit. Suppose P denotes the asset's price. Then the respective return $r_i = \frac{P_i - P_{i-1}}{P_{i-1}}$ is the object variable we analyze.

4.1 Modeling the Marginal Distributions and the Dependence Structure with Copula

We fitted the GARCH models for the series r_1, r_2, r_3 as initial models with normal and student-t distribution. With analysis we find that the three series all have no autocorrelation. Then the simple GARCH(1,1) model we apply follows the following expression

$$r_t = \mu + a_t, \quad a_t = \sqrt{h_t} \cdot \varepsilon_t$$
$$h_t = c + \beta_1 \cdot a_{t-1}^2 + \beta_2 \cdot h_{t-1}$$

where ε_t is white noise processes with zero mean and unit variance, β_1 and β_2 follows the restriction $\beta_1 + \beta_2 < 1$. Table 1 presents the estimates of GARCH model using the Eviews software. Trough the table, we find that the HS index and Nikkei return variables satisfy the GARCH(1,1) model while the Dow Jones coincides the TAR-GARCH(1,1) model.

The distributions of the three return variables are forecasted through the GARCH model. Table 1 also presents the distributions with their parameters: mean $E(r)$ and standard variance σ.

Table 1. Parameter estimates of GARCH models and the distributions

parameters	HS index	Dow Jones index	Nikkei index
μ	0.000797	0.000109	0.001008
c	2.94E-06	2.58E-06	3.07E-06
β_1	–	-0.034828	0.081943
β_2	0.960807	0.925228	0.887853
TGARCH	–	0.103353	–
AIC	-6.716928	-7.196086	-6.331889
distribution	student-t	normal	normal
$E(r)$	0.000797	0.000109	0.001008
σ	0.00866691	0.007853746	0.012598705

The copula function we use is Clayton copula which mainly describes the dependence of left tail. The multi-dimensional Clayton copula is denoted in the following expression

$$C(u_1, u_2, \cdots, u_n) = \left(\sum_{i=1}^{n} u_i^{-\theta} + 1 - n \right)^{-\frac{1}{\theta}}$$

After the maximum likelihood estimation, the parameter θ of the two-dimensional copula between HS index and Dow Jones index is 0.1186. Then the two-dimensional copula function can be expressed in the following formula

$$C(u_1, u_2) = \left(u_1^{-\theta} + u_2^{-\theta} - 1\right)^{-\frac{1}{\theta}} = \left(u_1^{-0.1186} + u_2^{-0.1186} - 1\right)^{-\frac{1}{0.1186}}$$

As mentioned, the conditional distribution $u_2 \to C_{u_1}(u_2) = \frac{\partial C(u_1, u_2)}{\partial u_1}$ is uniformly distributed on $[0, 1]$. We apply the Kolmogorov-Smirnov method to test the values of $C_{u_1}(u_2)$ evaluated by the sample data (r_{1i}, r_{2i}). The test answer presents that the P value is 0.2869, which is higher than the confidence level 0.05. Consequently the two-dimensional Clayton copula fits the sample data well.

As well as two-dimensional copula, the two steps conditional distribution $C_{u_1,u_2}(u_1, u_2, u_3) = \frac{\partial C(u_1,u_2,u_3)}{\partial u_1 \partial u_3}$ is also uniformly distributed on [0,1]. The estimated value of θ is 0.0615 with P value 0.1047>0.05 by Kolmogorov-Smirnov test. Therefore the three-dimensional Clayton copula can be expressed in the formula

$C(u_1, u_2, u_3) = \left(u_1^{-0.0615} + u_2^{-0.0615} + u_3^{-0.0615} - 2\right)^{-\frac{1}{0.0615}}$

Applying the copula based Monte Carlo simulation; we get 10000 series of data for the following empirical analysis.

4.2 Copula Based Three-Dimensional Portfolio Optimization

1. traditional portfolio efficient frontier: We note $\omega_1, \omega_2, \omega_3$ are the proportions of the portfolio. On the basis of the traditional mean-variance theory, we get the minimum standard variance with the optimal proportions under given target mean of the portfolio. Table 2 presents the optimal data we calculated.

Join the risk-free asset with the daily rate $r_f = 0.00005863$. the optimal proportion is $\omega_1 = 0.52788$, $\omega_2 = 0.14994$, $\omega_2 = 0.32218$. Figure 1(a) shows the "mean-standard variance" efficient frontier and the tangent with risk-free rate.

2. copula based portfolio efficient frontier: Contrast to the traditional mean-variance theory, we apply the GARCH based marginal distributions and copula based Monte Carlo simulation to calculate the efficient frontier. The software we used is still the LINGO. The result is presented in table 3.

In order to see the good characteristic of the copula based portfolio optimization, we compare the traditional efficient frontier with the copula based one, as shown in figure 1(b). Figure 1(b) presents that the sphere of copula based frontier is almost less than the traditional one under joint normal distribution, which validates copula's good characteristic of dependence, because the copula based frontier considers the risk more than the traditional one.

Table 2. The optimal proportions of the traditional efficient frontier

Er	ω_1	ω_2	ω_3	σ	Er	ω_1	ω_2	ω_3	σ
0.00018	0	1	0	0.00667	0.00044	0.3768	0.4309	0.1923	0.00579
0.0002	0.0277	0.9723	0	0.00652	0.00046	0.4002	0.3872	0.2126	0.00595
0.00022	0.0750	0.9250	0	0.00628	0.00048	0.4244	0.3423	0.2333	0.00613
0.00024	0.1223	0.8777	0	0.00607	0.0005	0.4482	0.2980	0.2538	0.00634
0.00026	0.1616	0.8308	0.0076	0.0059	0.00052	0.4721	0.2537	0.2742	0.00657
0.00028	0.1849	0.7873	0.0278	0.00054	0.00054	0.4959	0.2095	0.2946	0.00682
0.0003	0.2100	0.7408	0.0492	0.00563	0.00056	0.5197	0.1652	0.3151	0.00708
0.00032	0.2338	0.6965	0.0697	0.00555	0.00058	0.5447	0.1187	0.3366	0.00738
0.00034	0.2576	0.6522	0.0902	0.00551	0.0006	0.5674	0.0766	0.3560	0.00767
0.00036	0.2803	0.6101	0.1096	0.0055	0.00062	0.5911	0.0324	0.3765	0.00798
0.00038	0.3053	0.5637	0.1310	0.00552	0.00064	0.5217	0	0.4783	0.00835
0.0004	0.3291	0.5194	0.1515	0.00558	0.000645	0.4248	0	0.5752	0.00862
0.00042	0.3529	0.4751	0.1720	0.00567	0.00065	0.3602	0	0.6398	0.00885

Table 3. The optimal proportions of the copula-GARCH based efficient frontier

Er	ω_1	ω_2	ω_3	σ	Er	ω_1	ω_2	ω_3	σ
0.00011	0.0015	0.9985	0	0.00777	0.00056	0.2597	0.4374	0.3029	0.00651
0.00014	0.0451	0.9549	0	0.00751	0.00059	0.2742	0.4006	0.3252	0.00663
0.00017	0.0715	0.9154	0.0131	0.00729	0.00062	0.2887	0.3638	0.3475	0.00677
0.0002	0.0859	0.8786	0.0355	0.0071	0.00065	0.3031	0.3271	0.3698	0.00693
0.00023	0.1004	0.8418	0.0578	0.00691	0.00068	0.3176	0.2903	0.3921	0.00711
0.00026	0.1149	0.8051	0.0800	0.00675	0.00071	0.3321	0.2535	0.4143	0.00732
0.00029	0.1294	0.7683	0.1023	0.00662	0.00074	0.3466	0.2168	0.4366	0.00753
0.00032	0.1439	0.7315	0.1246	0.0065	0.00077	0.3611	0.1799	0.4589	0.00777
0.00035	0.1583	0.6948	0.1469	0.00641	0.0008	0.3755	0.1432	0.4812	0.00802
0.00038	0.1728	0.6580	0.1691	0.00634	0.00083	0.3900	0.1064	0.5035	0.00828
0.00041	0.1873	0.6212	0.1915	0.00631	0.00086	0.4045	0.0697	0.5258	0.00855
0.00044	0.2018	0.5844	0.2138	0.00629	0.00089	0.4190	0.0329	0.5481	0.00884
0.00047	0.2163	0.5477	0.2360	0.00631	0.00092	0.4171	0	0.5829	0.00913
0.0005	0.2307	0.5109	0.2584	0.00635	0.00095	0.2749	0	0.7251	0.00978
0.00053	0.2452	0.4741	0.2807	0.00642	0.00098	0.1327	0	0.8673	0.01091

4.3 Copula-CVaR Restricted Multivariate Portfolio Optimization

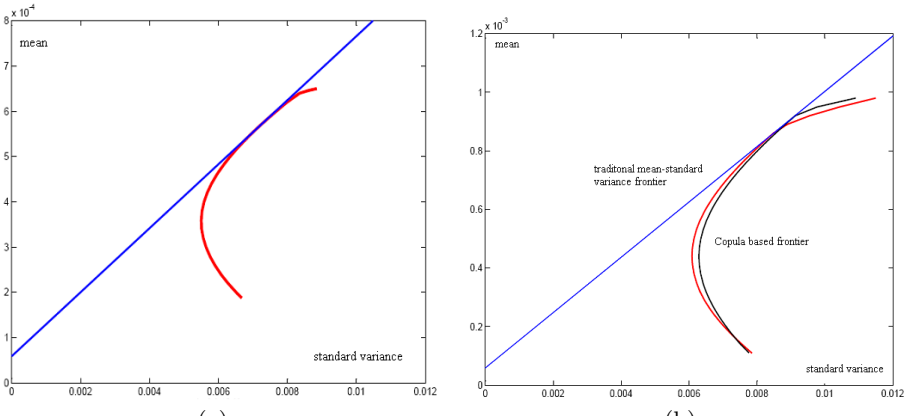
Taking the minimum of CVaR as the objective function, The mean-CVaR frontier can be calculated on condition that the confidence level is 0.95 and the scenario is generated by copula. Table 4 presents the optimal proportions under the given return of the portfolio.

In the same way, we draw the frontier figure through the Matlab software. Figure 2(a) presents the copula-CVaR restricted "mean-CVaR" efficient frontier and the tangent with risk-free rate.

Under the normal circumstance, CVaR can be expressed by the standard variance σ as $CVaR = \frac{\sigma}{\alpha\sqrt{2\pi}} \exp\left(-\frac{q_\alpha^2}{2}\right)$, where q_α is the quantitative α-percentile

Table 4. The optimal proportions of the copula-CVaR restricted "mean-CVaR" efficient frontier

Er	ω_1	ω_2	ω_3	σ	Er	ω_1	ω_2	ω_3	σ
0.00011	0.0015	0.9985	0	0.0159	0.00056	0.2133	0.4483	0.3384	0.0144
0.00014	0.0451	0.9549	0	0.0155	0.00059	0.2274	0.4116	0.3610	0.0147
0.00017	0.0666	0.9165	0.0169	0.0151	0.00062	0.2421	0.3748	0.3831	0.0150
0.0002	0.0748	0.8812	0.0440	0.0148	0.00065	0.2537	0.3387	0.4076	0.0153
0.00023	0.0929	0.8436	0.0635	0.0145	0.00068	0.2719	0.3010	0.4270	0.0157
0.00026	0.1082	0.8066	0.0851	0.0142	0.00071	0.2830	0.2650	0.4519	0.0161
0.00029	0.1162	0.7714	0.1124	0.0140	0.00074	0.2999	0.2277	0.4724	0.0165
0.00032	0.1209	0.7369	0.1422	0.0139	0.00077	0.3083	0.1924	0.4993	0.0171
0.00035	0.1325	0.7008	0.1666	0.0138	0.0008	0.3223	0.1557	0.5220	0.0175
0.00038	0.1425	0.6649	0.1916	0.0137	0.00083	0.3338	0.1196	0.5466	0.0180
0.00041	0.1509	0.6298	0.2193	0.0137	0.00086	0.3494	0.0826	0.5679	0.0185
0.00044	0.1614	0.5939	0.2447	0.0138	0.00089	0.3597	0.0468	0.5934	0.0190
0.00047	0.1772	0.5568	0.2659	0.0139	0.00092	0.3784	0.0906	0.6124	0.0196
0.0005	0.1927	0.5198	0.2874	0.0140	0.00095	0.2749	0	0.7251	0.0206
0.00053	0.0997	0.4848	0.3154	0.0142	0.00098	0.1327	0	0.8673	0.0226



(a) (b)

Fig. 1. (a) the traditional "mean-standard variance" efficient frontier and the tangent with risk-free rate; (b) the copula-GARCH based "mean-standard variance" efficient frontier together with the traditional one

of standard normal distribution. Under the confidence level of 0.95, we have $q_{0.05} = 1.64485$. Then the CVaR is calculated by $CVaR = 2.0627\sigma$. Consequently, the copula based mean-CVaR frontier can be compared with the traditional one converted by the equation between CVaR and standard variance. Figure 2(b) shows the copula based mean-CVaR frontier and the converted normal mean-CVaR frontier.

We find that the sphere of copula based frontier is almost less than the normal one. For the reason of considering more about the risk, this figure presents that

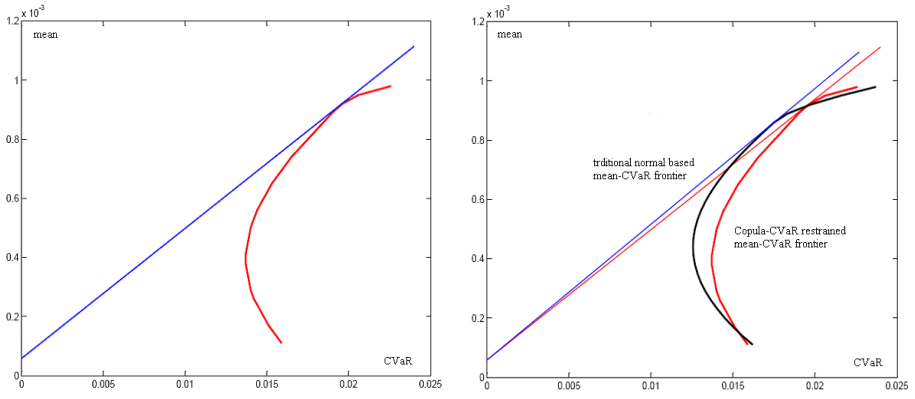


Fig. 2. (a) the copula-CVaR restricted "mean-CVaR" efficient frontier and the tangent with risk-free rate; (b) the copula-CVaR restricted "mean-CVaR" efficient frontier comparing with the normal based one

copula-CVaR restrained portfolio optimization can avoid some risk which the normal distribution based portfolio optimization allows. So we can conclude that the copula theory and copula-CVaR method do better in the optimization of portfolio management.

5 Conclusions

This paper applies the copula theory into the optimization of portfolio management. Our empirical analysis firstly uses three-dimensional copula for Monte Carlo simulation and CVaR calculation. With the copula based multi-dimensional simulated scenarios, we get the optimal investing proportions of the portfolio under the minimum of standard variance calculated by copula. Adding the risk-free asset, we also get the optimal proportion and the tangent of "the mean-standard variance" efficient frontier.

In addition, we apply the copula based CVaR into the portfolio optimization, which we call it copula-CVaR method. Under the objective function of minimum of copula-CVaR, we get another series of optimal investing proportions. Consequently, the "mean-CVaR" frontier is calculated with the tangent under risk-free asset. By comparison with the traditional normality based portfolio theory, we find that the copula theory and copula-CVaR method do better in the optimization of portfolio management.

Besides this, this paper creatively applies the three-dimensional Archimedean copula in the empirical study and makes use of copula theory in the CVaR caculation which is also a contribution. We hope the use of the multi-dimensional copula and copula-CVaR approach can be applied more widely in the future financial analysis.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (No. 70521001, 70531010 and 70571004).

References

1. Sklar, A.: Fonctions de repartition?n dimensions et leurs marges. Publication de l'Institut de Statistique de l'Université Paris 8, 229–231 (1959)
2. Cherubini, U.: Value-at-risk Trade-off and Capital Allocation with copulas. Note CRAS Paris 7, 235–256 (2001)
3. Helder, P., Luiz, K.H.: Using Conditional copula to Estimate Value at Risk. J. of Data Sci. 4, 93–115 (2006)
4. Patton, A.J.: Modeling Time-Varying Exchange Rate Dependence Using the Conditional copula, Ch. 1. University of California, San Diego (2002)
5. Rockafellar, R.T.: Stanislav Uryasev: Optimization of Conditional Value-at-Risk. J. of Risk 2(3), 21–41 (2000)
6. Rockafellar, R.T.: Stanislav Uryasev: conditional value at risk for central loss distributions. University of Florida, working paper (2001)
7. Siddharth, A., Coleman, T.F., Li, Y.Y.: Minimizing CVaR and VaR for a Portfolio of Derivatives (working paper). Cornell University, New York (2004)
8. Gordon, J.A., Alexandre, M.B.: A comparison of VaR and CVaR constrants on portfolio selection with the mean-variance model. Management. Sci. 50(9), 1261–1273 (2004)
9. Nelsen, R.B.: An Introduction to copulas. Springer, New York (1999)
10. Markowitz, H.: Portfolio Selection. J. of Finance 3, 77–91 (1952)
11. Tobin: Liquidity preference as behavior toward risk. Review of Economic Studies 25, 65–86 (1958)
12. Andreev, A., Kanto, A.: Conditional Value-at-Risk estimation using non-integer values of degrees of freedom in Student's t-distribution. J. of Risk 2, 55–62 (2005)

Online Capacitated Interval Coloring

Leah Epstein[1], Thomas Erlebach[2], and Asaf Levin[3]

[1] Department of Mathematics, University of Haifa, 31905 Haifa, Israel
lea@math.haifa.ac.il
[2] Department of Computer Science, University of Leicester, England
t.erlebach.mcs.le.ac.uk
[3] Department of Statistics, The Hebrew University, Jerusalem, Israel
levinas@mscc.huji.ac.il

Abstract. In the online capacitated interval coloring problem, a sequence of requests arrive online. Each of the requests is an interval $I_j \subseteq \{1, 2, \ldots, n\}$ with bandwidth b_j. Initially a vector of capacities $(c_1, c_2, \ldots, c_n)$ is given. Each color can support a set of requests such that the total bandwidth of intervals containing i is at most c_i. The goal is to color the requests using a minimum number of colors. We present a constant competitive algorithm for the case where the maximum bandwidth $b_{\max} = \max_j b_j$ is at most the minimum capacity $c_{\min} = \min_i c_i$. For the case $b_{\max} > c_{\min}$, we give an algorithm with competitive ratio $O(\log \frac{b_{\max}}{c_{\min}})$ and, using resource augmentation, a constant competitive algorithm. We also give a lower bound showing that constant competitive ratio cannot be achieved in this case without resource augmentation.

1 Introduction

Motivated by a routing problem in optical networks we consider the following problem. We are given a line network with links $1, 2, \ldots, n$ and a vector of base capacities $(c_1, c_2, \ldots, c_n)$. The requests arrive one by one, in an online fashion, and each request is identified by the interval of links that it uses $I_j = [s_j, t_j]$ where $1 \leq s_j \leq t_j \leq n$. Moreover, the request I_j is associated with a bandwidth b_j that is the *bandwidth request* of I_j. Each time a request arrives, a color must be assigned to it before the next request is revealed. A restriction on the coloring is that the total bandwidth of all requests that are assigned a common color and contain link i is at most c_i. The goal is to use a minimum number of colors. Naturally, we assume $b_j \leq c_i$ for all $i \in I_j$ (otherwise a feasible coloring would not exist). Without loss of generality we also assume (by scaling) that $\min_{i=1,2,\ldots,n} c_i = 1$.

As practical motivation of our study, consider an optical line network, where each color corresponds to a distinct frequency (this frequency is seen as a color as it is a frequency of light) in which the information flows. Different links along the line have different capacities, which are a function of intermediate equipment along the link (e.g., a link with an intermediate repeater may have reduced capacity for each color as a result of the repeater). Each request uses the same

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 243–254, 2007.

bandwidth on all links that this request contains. Moreover, requests arrive over time. As the number of distinct available frequencies is limited, minimizing the number of colors for a given sequence of requests is a natural objective. Changing the color allocation of a request causes a setup cost that we would like to avoid, and therefore we restrict ourselves to the online problem where once a request is allocated a color this color allocation cannot be changed.

From a theoretical point of view, the problem is interesting as it extends the previously studied case of uniform capacities ($c_i = 1$ for all i) to the setting with arbitrary capacities. For many problems of a similar flavor (both in the online and offline variants), the setting with arbitrary capacities is significantly more difficult to deal with than the uniform setting, and new techniques and ideas are often required. For example, a $\frac{3}{2}$-approximation algorithm for offline coloring of unit-bandwidth paths in trees with uniform edge capacities follows easily from the known results for the unit-capacity case [17], but nontrivial new techniques were needed to obtain a 4-approximation for the case with arbitrary capacities [6]. Similar observations can be made for the throughput version of such problems (i.e., maximizing the total bandwidth of requests that can be accepted with one available color). For example, the only known constant competitive algorithm for online throughput maximization in line or ring networks uses randomization and preemption and works only for the case of uniform edge capacities [2]. For the offline version of these problems, Chakrabarti et al. remark in [5] that most of the techniques that have been used for the uniform capacity case do not seem to extend to the case of arbitrary capacities.

In order to analyze our online algorithms for capacitated interval coloring, we use the common criterion of competitive analysis. For an algorithm $\mathcal{A}$, we denote its cost by $\mathcal{A}$ as well. The cost of an optimal offline algorithm that knows the complete sequence of intervals in advance is denoted by OPT. We consider the absolute competitive ratio that is defined as follows. The absolute competitive ratio of $\mathcal{A}$ is the infimum $\mathcal{R}$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \mathrm{OPT}$. If the absolute competitive ratio of an online algorithm is at most $\mathcal{C}$ we say that the algorithm is $\mathcal{C}$-competitive.

The problem studied in this paper is a generalization of the classical online interval graph coloring problem. If all capacities are 1, all bandwidth requests are 1, and the number of links in the network is unbounded, we arrive at the standard interval coloring problem.

Coloring interval graphs has been intensively studied. Kierstead and Trotter [14] constructed an online algorithm which uses at most $3\omega - 2$ colors where ω is the maximum clique size of the interval graph. They also presented a matching lower bound of $3\omega - 2$ on the number of colors in a coloring of an arbitrary online algorithm. Note that the chromatic number of interval graphs equals the size of a maximum clique, which is equivalent in the case of interval graphs to the largest number of intervals that intersect any point (see [11]). Many papers studied the performance of First Fit for this problem [12,13,16,7]. It is shown in [7] that the performance of First Fit is strictly worse than the one achieved by the algorithm of [14].

Generalizations of interval coloring received much attention recently. Adamy and Erlebach [1] introduced the interval coloring with bandwidth problem (which is also a special case of our problem). In this problem all capacities are 1 and each interval has a bandwidth requirement in $(0, 1]$. As in our problem, the intervals are to be colored so that at each point, the sum of bandwidths of intervals colored by a certain color does not exceed the capacity, which is 1. This problem was studied also in [15,8,3]. The best competitive ratio known for that problem is 10 [15,3]. A lower bound strictly higher than 3 was shown in [8].

Other previous work is concerned with the throughput version of related problems. In the demand flow problem on the line, each interval is associated with a profit, and the goal is to maximize the total profit of the accepted intervals without violating any edge capacity. This corresponds to maximizing the total profit of intervals that can receive the same color in our model. For the off-line version of that problem, constant-factor approximation algorithms have been presented in [5,6] for the case where $b_{\max} \leq c_{\min}$, where $b_{\max} = \max_j b_j$ is the maximum requested bandwidth and $c_{\min} = \min_{i=1,\ldots,n} c_i$ is the minimum edge capacity. For the general case, an approximation ratio $O(\log \frac{b_{\max}}{c_{\min}})$ was achieved in [5]. Recently, a quasi-polynomial time approximation scheme was presented [4].

Our results: In Sect. 3 we present our first main result, a constant competitive algorithm for capacitated interval coloring for the case in which the maximum bandwidth request is at most the minimum capacity, i.e., the case where $b_{\max} \leq c_{\min}$. (Note that this restriction means that the minimum edge capacity *anywhere* on the line must be at least as large as the maximum bandwidth of any request. This is stronger than the standard requirement that $b_j \leq c_i$ for all $i \in I_j$.) This is an important special case that contains the interval coloring problem with bandwidth studied in [1,15,3,8]. This restriction on the maximum bandwidth is common in work on demand flow problems as well, see e.g. [5,6]. While our algorithm uses the standard technique of partitioning the requests into different types and dealing with each type separately, in our case the different types need to share colors and so the bandwidth sharing scheme for the colors needs to be designed very carefully.

We also consider the general case, i.e., the case where $b_{\max}$ can be larger than $c_{\min}$. First, we remark that it is not difficult to design an $O(n)$-competitive algorithm for this case. This can be done by partitioning the requests into at most n sets, each of which contains all requests for which the bottleneck link (i.e., the link of smallest capacity) is link i. We are left with n disjoint instances of bin packing, and we can run e.g. First-Fit on each set. In Sect. 4, we first design an $O(\log b_{\max})$-competitive algorithm (the ratio is $O(\log \frac{b_{\max}}{c_{\min}})$ if the capacities are not normalized). Then we show that for any amount ε of resource augmentation on the capacities (i.e. increasing capacities by a multiplicative factor of at most $1+\varepsilon$), we can obtain a constant competitive algorithm (the ratio is a function of ε). Finally, we give our second main result, a lower bound showing that no online algorithm can achieve constant competitive ratio in the general case without resource augmentation. The basic idea of our lower bound is to adapt the known logarithmic lower bound for online coloring of trees [10] to our problem. However,

arbitrary trees cannot be represented as interval graphs, and hence we need to use the capacities and bandwidths in a very intricate way in order to encode the required tree structures. Furthermore, the construction must be such that the algorithm cannot benefit from the information that is conveyed by the encoding.

Several proofs are omitted due to space limitations.

2 Preliminaries

The following $KT_{\ell b}$ algorithm for the online interval coloring with bandwidth problem (where all edges have the same capacity) was studied by Epstein and Levy [8,9] (see also [15,3]). We are given an upper bound b on the maximum request bandwidth. We are also given a value of a parameter ℓ. The algorithm partitions the requests into classes and then colors each class using the First-Fit algorithm. The partition of the requests is performed online so that a request j is allocated to class m where m is the minimum value so that the maximum load of the requests that were allocated to classes $1, 2, \ldots, m$, together with the additional new request, is at most $m\ell$. (For a set of requests, the *load* created on a link is the sum of the bandwidths of the requests containing that link, and the *maximum load* is the largest load of all links.) For an interval v_i that was allocated to class m, a *critical point of* v_i is a point q in v_i such that the set of all the intervals that were allocated to classes $1, 2, \ldots, m-1$ prior to the arrival of v_i, together with the interval v_i, has total load greater than $(m-1)\ell$ in q (i.e., q prevents the allocation of v_i to class $m-1$). They proved the following lemmas:

Lemma 1. *Given an interval v_i that was allocated class m. For the set A_m of intervals that were allocated to class m, and for every critical point q of v_i, the total load of A_m in q is at most $b + \ell$.*

Lemma 2. *For every m, the set A_m of intervals that were allocated to class m has a maximum load of at most $2(b + \ell)$.*

Note that the set A_m of intervals assigned to class m can be colored with a single color if its maximum load does not exceed the capacity of any edge (cf. [15]).

Lemma 3. *The number of classes used by the algorithm is at most $\lceil \frac{\omega^*}{\ell} \rceil$ where ω^* is the maximum load.*

It was shown in [14] that using the above algorithm with $b = \ell = 1$ in the case where all intervals have unit bandwidth ($b_j = 1$ for all j) results in classes that have maximum load two and can be colored online with three colors per class (the first class can be colored using a single color), assuming unit edge capacity. (If the edges have capacity 2, one color suffices for each class. The same obviously holds also if $b = \ell = \frac{1}{2}$, all requests have bandwidth equal to $\frac{1}{2}$, and the edges have unit capacity.) We refer to this special case of algorithm $KT_{\ell b}$ as algorithm KT; it is the classical algorithm by Kierstead and Trotter that requires at most $3\omega - 2$ colors for coloring a set of intervals with maximum clique size ω.

3 Algorithm for the Case $\max_j b_j \leq \min_{i=1,2,\ldots,n} c_i$

The Algorithm. Since we assume that $\min_{i=1,2,\ldots,n} c_i = 1$, all bandwidth requests are at most 1. The *level* of request $I_j = [s_j, t_j]$ is $\lfloor \log_2 \min_{i \in I_j} c_i \rfloor$, i.e. the rounded down base 2 logarithm of the minimum capacity of any link along the request. A level $i > 0$ request is *small* if its bandwidth is at most 2^{i-3}, and a level 0 request is *small* if its bandwidth is at most $\frac{1}{4}$. A request that is not small is a *large request*. Note that large requests exist only in level 0, 1 and 2.

Our algorithm first rounds down all capacities to integer powers of 2, this does not change the classification into levels. Next it performs an online partition of the requests according to their levels. For all i, the small requests of level i are colored using an algorithm for online coloring along a line network with identical capacities, and these capacities are $\max\{1, 2^{i-1}\}$. For the coloring of these small requests we use the same set of colors for the requests of all levels. More specifically, requests of level 0 are allocated a capacity of 1 in each color, on every link. Requests of level $i > 0$ are allocated a capacity of 2^{i-1} in each color, on every link. To color the small requests, note that a small request has bandwidth at most 2^{i-3} for $i > 0$ and at most $\frac{1}{4}$ for level 0. Therefore we can apply the algorithm $KT_{\ell b}$ from Sect. 2, using $b = \ell = 2^{i-3}$ for $i > 0$ and $b = \ell = \frac{1}{4}$ for level 0. A new class is opened if a new request of some level opens a new class. Each class is colored using a single color, i.e., given color t, it is used for all requests assigned to class t, no matter which level they belong to. It is not difficult to show that this coloring is valid.

As for large requests we first define the following types. We define a *type 1 large request* to be a level 1 large request with bandwidth requirement that belongs to the interval $\left(\frac{1}{2}, 1\right]$. A large request that is not type 1 is called a *type 2 large request*. Each type of large request is packed independently using its own set of colors. We next describe the packing of each type of large requests.

Type 1 large requests. We round up all bandwidth requests to 1 and then apply algorithm KT, the online algorithm for interval coloring (without bandwidth) of Kierstead and Trotter [14]. However, unlike that algorithm, where each class was colored using three colors, we can use a single color for each class, similarly to the algorithm for coloring requests of bandwidth in $(\frac{1}{4}, \frac{1}{2}]$ in [15], see also Sect. 2.

Type 2 large requests. We partition the type 2 large requests into three subgroups according to their levels. For each new open color we allocate a total unit capacity for all the type 2 large requests of level 0. Moreover for each link whose rounded capacity is at least two we also allocate a unit capacity for all the type 2 large request of level 1. For each link whose rounded capacity is at least four we allocate two units of capacity for all the type 2 large requests of level 2. We then apply the following algorithms depending on the level of the large request.

A level 0 large request of type 2. We further partition these requests into two sub-families of requests according to their bandwidth request. The first

sub-family consists of requests with bandwidth in $\left(\frac{1}{4}, \frac{1}{2}\right]$, and the second sub-family consists of requests with bandwidth in $\left(\frac{1}{2}, 1\right]$. For each sub-family we use its own set of colors (note that all these colors can be used also by large requests of type 2 from levels 1 and 2). For each request in the first sub-family we round up its bandwidth request to $\frac{1}{2}$ and then apply algorithm KT, the online algorithm for interval coloring (without bandwidth) of Kierstead and Trotter, where each class can be packed into a common color, as is done for type 1. For the second sub-family we also round up its bandwidth request to 1 and afterwards apply algorithm KT, where each class is packed using three colors, exactly as in [14].

A level 1 large request of type 2. We recall that such a request has bandwidth at most $\frac{1}{2}$. We round up its bandwidth request to $\frac{1}{2}$ and then apply algorithm KT, where each class can be packed into a common color.

A level 2 large request of type 2. We round up its bandwidth request to 1 and apply algorithm KT, where each class can be packed into a common color.

Analysis. First, one can show that the solution returned by the algorithm is feasible. In fact, even the rounded capacity constraints are satisfied by the coloring produced by the algorithm. The next lemma is a trivial consequence of the fact that the colors used to color small requests of the different levels are shared among the levels.

Lemma 4. *Let s_j be the number of colors used to color the small requests of level j. Then, the number of colors used by the algorithm for coloring the small requests is exactly* $\max_{j \geq 0} s_j$.

Furthermore, we can show that $\mathrm{OPT} \geq \frac{s_j}{32}$ for all j. Together with Lemma 4, this gives the following:

Corollary 1. *The number of colors used to color the small requests is at most* $32 \cdot \mathrm{OPT}$.

It remains to analyze the number of colors used by the large requests.

Lemma 5. *Let b be a fixed value that is either $\frac{1}{2}$ or 1 and let c be a fixed value that is either b or $2b$. Assume that we are given a subset $\mathcal{S}$ of large request of level i, $0 \leq i \leq 2$, each with bandwidth in the interval $\left(\frac{b}{2}, b\right]$ and we first round up the bandwidth to b and afterwards use Kierstead and Trotter's algorithm KT with color capacity c. Then, if $b < c$ the number of colors used to color all the requests of this family is at most $2 \cdot \left(\frac{2^{i+2}}{b} - 1\right) \cdot \mathrm{OPT}$, and otherwise (if $b = c$) the number of colors used to color all the requests of this family is at most $6 \cdot \left(\frac{2^{i+2}}{b} - 1\right) \cdot \mathrm{OPT}$.*

Lemma 5 implies, using $b = 1$, $c = 2$ and $i = 1$, that the number of colors that are used by the algorithm to color all type 1 large requests is at most $14 \cdot \mathrm{OPT}$.

Furthermore, we get that the number of colors that are used to color all type 2 large requests of level 0 is at most $32 \cdot \mathrm{OPT}$; this follows by using $b = \frac{1}{2}$, $c = 1$ and $i = 0$ for the requests with bandwidth in $\left(\frac{1}{4}, \frac{1}{2}\right]$, and $b = 1$, $c = 1$

and $i = 0$ for the requests with bandwidth in $\left(\frac{1}{2}, 1\right]$. Similarly, we get that the number of colors for type 2 large requests of level 1 is at most $30 \cdot \mathrm{OPT}$ (using $b = \frac{1}{2}$, $c = 1$ and $i = 1$), and the number of colors for type 2 large requests of level 2 is at most $30 \cdot \mathrm{OPT}$ (using $b = 1$, $c = 2$ and $i = 2$). As the colors used to color type 2 large requests of different levels are shared among the levels, the number of colors used to color all type 2 large requests is the maximum among the numbers of colors used to color type 2 large requests of level i for $i = 0, 1, 2$. By considering the different cases above, this maximum is at most $32 \cdot \mathrm{OPT}$.

Theorem 1. *The algorithm is* 78*-competitive.*

Proof. Each color is used to either color small requests, or to color large requests of type 1, or to color large requests of type 2. By Corollary 1 there are at most $32 \cdot \mathrm{OPT}$ colors that are used to color small requests. As discussed above, at most $14 \cdot \mathrm{OPT}$ colors are used to color large requests of type 1, and at most $32 \cdot \mathrm{OPT}$ colors are used to color large requests of type 2. The claim follows since $32 \cdot \mathrm{OPT} + 14 \cdot \mathrm{OPT} + 32 \cdot \mathrm{OPT} = 78 \cdot \mathrm{OPT}$. □

4 Algorithms and Lower Bound for the General Case

An $O(\log b_{\max})$-Competitive Algorithm. Recall that $b_{\max}$ denotes the maximum bandwidth of a request. We now deal with the general case where $b_{\max}$ can be larger than $c_{\min}$. We still assume that the edge capacities are normalized so that $c_{\min} = 1$. In order to present an $O(\log b_{\max})$-competitive algorithm, we first note that the algorithm of the previous section is designed in such a way that it can handle small requests even if they have bandwidth requests which are larger than 1 and provides a solution whose cost is at most $32 \cdot \mathrm{OPT}$ for these requests. Therefore, it suffices to consider the large requests. Recall that for $i > 0$, a level i request is large if its bandwidth is at least 2^{i-3}, and it has a link with capacity that is smaller than 2^{i+1}.

In our algorithm we perform an online partition of large requests into levels, and pack the large requests of each level separately using colors that are dedicated to the level. To pack the large requests of level i, we disregard the capacities and bandwidth of the requests, and we pack the requests using Kierstead and Trotter's algorithm KT assuming unit capacities and unit bandwidths. This completes the definition of the algorithm, and it remains to analyze it.

First, it is not difficult to verify that the algorithm produces a feasible solution. The number of levels is $O(\log b_{\max})$, as our algorithm uses colors to color large requests of level i only if there is at least one large request of level i. Furthermore, we can show that for each i our algorithm uses $O(\mathrm{OPT})$ colors to color all the large requests of level i. We thus obtain the following theorem.

Theorem 2. *There exists an $O(\log b_{\max})$-competitive algorithm for the general case of the capacitated interval coloring problem.*

Note that we have assumed $c_{\min} = 1$ without loss of generality. In the case where $c_{\min}$ is not normalized to 1, the ratio becomes $O(\log \frac{b_{\max}}{c_{\min}})$.

Resource Augmentation Algorithm. Given a fixed positive number $0 < \varepsilon < 1$ such that $\frac{1}{\varepsilon}$ is an integer, we allow the online algorithm to use colors such that the total bandwidth of requests that are assigned a common color and contain the link i is at most $(1+\varepsilon)c_i$. I.e., the online algorithm is allowed to use slightly larger capacities than the offline algorithm is allowed. Let $\delta = \frac{\varepsilon}{3}$.

We perform an online partition of the requests into large requests and small requests. We pack the small requests similarly to the previous section with at most $32 \cdot \mathrm{OPT}$ colors. We next describe the algorithm to obtain a coloring of the large requests.

Let $\tilde{c_j}$ denote c_j rounded up to the nearest integer power of $(1+\delta)$. We now define the level of a request $[s_i, t_i]$ to be the logarithm with respect to the base $(1+\delta)$ of the minimum rounded capacity of a link along this request, i.e., $\log_{1+\delta} \min_{s_i \le j \le t_i} \tilde{c_j}$. For each level i we use algorithm KT to compute a packing of its large requests into colors, using a capacity of $(1+\delta)^i$ on each link. For each i, one can show that the algorithm uses $O(\mathrm{OPT})$ colors to color all the large requests of level i.

For each i, we define the type of i to be $i \bmod \frac{1}{\delta^2}$. Therefore, there are exactly $\frac{1}{\delta^2}$ types. For all levels with a common type we use the same set of colors, whereas for different types we use disjoint sets of colors. Therefore, the total number of colors used by our algorithm is at most $O(\frac{1}{\delta^2}) \cdot \mathrm{OPT}$, and this provides a constant competitive ratio for all constant values of δ. Furthermore, we can show that the edge capacities are violated at most by a factor of $(1+\varepsilon)$: Given a color c that is used to color large requests of type i, and a link j whose capacity is c_j, the total bandwidth of requests that are colored c and contain j is at most $(1+3\delta)\, c_j = (1+\varepsilon)c_j$. We obtain the following theorem.

Theorem 3. *For every constant $\varepsilon > 0$, there is a constant competitive algorithm for the general case of the capacitated interval coloring problem with resource augmentation by a factor of $1+\varepsilon$.*

Competitive Lower Bound. We finally outline a lower bound construction showing that no deterministic algorithm can achieve constant competitive ratio in the general case (without resource augmentation). Let $\mathcal{A}$ be any deterministic online algorithm for the problem. We imagine the links of the line numbered from left to right, starting with link 1 as the leftmost link. The capacity of link j is set to 3^j, for all $j \ge 1$. We identify colors with positive integers. Whenever $\mathcal{A}$ uses a new color, and it has used $i-1$ distinct colors prior to using that color, the new color is defined to be color i.

In the adversary construction, each newly presented interval has its left endpoint strictly to the right of all left endpoints of previously presented intervals, and it has a strictly larger bandwidth than all previously presented intervals. In fact, an interval with leftmost link L has bandwidth at least $3^L - 3^{L-1} > 3^{L-1}$. Furthermore, the set of all presented intervals can be colored optimally with two colors.

The adversary strategy makes use of a component (i.e., a subroutine that is used as part of the construction) denoted by $C_F(\ell)$, where F can be any set of

positive integers (the set of forbidden colors) and ℓ can be any positive integer. The goal of $C_F(\ell)$ is to force the algorithm to use a color that is not in F. Furthermore, the interval I on which $\mathcal{A}$ uses a color not in F is the last interval presented in the component. The length of I is at least ℓ. A component $C_F(\ell)$ is placed on a part of the line with leftmost link L (i.e., no interval presented in $C_F(\ell)$ contains a link to the left of L). An instance of $C_F(\ell)$ with leftmost link L is also called a $C_F(\ell)$ *at* L. Note that different incarnations of $C_F(\ell)$ may contain different (non-isomorphic) sets of intervals, as the intervals presented by the adversary depend on the actions of the on-line algorithm $\mathcal{A}$. The construction of $C_F(\ell)$ for $|F| > 1$ is recursive and makes use of smaller components $C_{F'}(\ell')$ for $F' \subset F$.

A component $C_F(\ell)$ at L requires a part of the line consisting of $g(\ell, |F|)$ links, for a suitable function g. Note that $g(\ell, |F|) \geq \ell$ must always hold, since already the last interval of $C_F(\ell)$ has length at least ℓ.

The adversary construction satisfies the following invariants.

Invariant 1: When the adversary presents a $C_F(\ell)$ at L, the total bandwidth of all previously presented intervals containing L is at most $\beta_L := 3^{L-1}$.

Invariant 2: Let R' be the leftmost among the rightmost ℓ links of the last interval I of the component $C_F(\ell)$ at L presented by the adversary (i.e., $R' = R - \ell + 1$ if R is the rightmost link of I). The construction ensures that the total bandwidth of intervals from $C_F(\ell)$ that contain R' is at most $3^{R'-1} - 3^{L-1}$.

After a $C_F(\ell)$ at L has been presented, only intervals with left endpoint R' (as defined in Invariant 2) or further to the right will be presented. Note that Invariant 2, together with Invariant 1, implies that the bandwidth of intervals starting to the left of R' and containing R' is at most $3^{L-1} + (3^{R'-1} - 3^{L-1}) = 3^{R'-1}$, so that Invariant 1 automatically holds again for components placed at R' or further to the right.

For $F = \emptyset$, a $C_F(\ell)$ at L consists of a single interval of length $\ell+1$ with leftmost link L and bandwidth $3^L - \beta_L$. The length of the part of the line required for a $C_F(\ell)$ with $|F| = 0$ is thus $g(\ell, 0) = \ell + 1$. As another simple case to start with, consider the case $F = \{f_1\}$ for some positive integer f_1. The adversary first presents an interval I_1 of length $\ell + 1$ with leftmost link L and bandwidth $3^L - \beta_L$. If $\mathcal{A}$ assigns a color different from f_1 to I_1, the component $C_F(\ell)$ is finished (and I_1 is the last interval of that component). If $\mathcal{A}$ assigns color f_1 to I_1, the adversary next presents an interval I_2 of length $\ell + 1$ whose leftmost link is the rightmost link R of I_1. The bandwidth of I_2 is $3^R - \beta_L$. Algorithm $\mathcal{A}$ must color I_2 with a color different from f_1, because I_1 and I_2 cannot receive the same color (their bandwidths add up to $3^L - \beta_L + 3^R - \beta_L = 3^R + (3^L - 2 \cdot 3^{L-1}) > 3^R$ and both intervals contain link R). The component $C_F(\ell)$ is finished, and I_2 is its last interval. Note that I_1 has rightmost link R and hence does not overlap the rightmost ℓ links of I_2. Therefore, the bandwidth occupied by this $C_F(\ell)$ on its rightmost ℓ links (starting with link $R + 1$) is bounded by $3^R - \beta_L$, showing that Invariant 2 is satisfied. The length of the part of the line required for the $C_F(\ell)$ with $|F| = 1$ is thus $g(\ell, 1) = 2\ell + 1$.

Let $|F| = k$ for some $k > 1$. The idea underlying the construction of $C_F(\ell)$ is to repeatedly use components $C_{F'}(\ell')$ for suitable subsets $F' \subset F$ to force $\mathcal{A}$ to use all colors from F on intervals that all intersect on a common link; then, a new interval that contains that link and is in conflict with the previous intervals containing that link is presented and must receive a color outside F. On the other hand, if the algorithm already uses a color outside F to color an interval presented in one of the recursive constructions $C_{F'}(\ell')$, the construction of $C_F(\ell)$ finishes right away. We can assume (by induction) that Invariant 2 has been shown to hold for the recursive constructions $C_{F'}(\ell')$ that are used in the construction of $C_F(\ell)$, and we will show that Invariant 2 holds again for $C_F(\ell)$.

Assume that $F = \{f_1, f_2, \ldots, f_k\}$. We will show how to construct $C_F(\ell)$ on a part of the line with leftmost link L. The construction proceeds in rounds. There will be at most k rounds, and after the last round one additional final interval may be presented.

For round 0, let $F_0 = \emptyset$. First, the adversary presents a $C_{F_0}(\ell_0)$ for suitable $\ell_0 \geq \ell$ starting at L. If $\mathcal{A}$ assigns a color outside F to the last (and only) interval of $C_\emptyset(\ell_0)$, the construction of $C_F(\ell)$ is finished. Otherwise, we can assume w.l.o.g. that $\mathcal{A}$ assigns color f_1 to the last interval I_0 of $C_\emptyset(\ell_0)$. Let R_0 be the rightmost link of I_0. Let $R'_0 = R_0 - \ell_0 + 1$. The remaining rounds up to round $k-1$ will take place inside the rightmost ℓ_0 links of I_0; only the final interval that may be presented after round $k-1$ extends beyond the right end of I_0.

For round 1, let $F_1 = \{f_1\}$. The adversary presents a $C_{F_1}(\ell_1)$ starting at $L_1 = R'_0$. Observe that the total bandwidth of intervals presented earlier that contain R'_0 is bounded by $3^{R'_0-1}$: bandwidth at most 3^{L-1} from intervals presented before the current $C_F(\ell)$ (by Invariant 1), and bandwidth at most $3^{R'_0-1} - 3^{L-1}$ from the $C_{F_0}(\ell_0)$ that was presented in round 0. The last interval I_1 of $C_{F_1}(\ell_1)$ receives some color c. If $c \notin F$, the construction of $C_F(\ell)$ is finished. If $c \in F$, we can assume w.l.o.g. that $c = f_2$.

In general, assume that round j of the construction of $C_F(\ell)$ has finished. The last interval I_j of the $C_{F_j}(\ell_j)$ presented in round j has received color f_{j+1}. Let R_j be the rightmost link of I_j. Let $R'_j = R_j - \ell_j + 1$. Arguing as above, we know that the total bandwidth of intervals containing R'_j that were presented so far is at most $3^{R'_j-1}$. Let $F_{j+1} = \{f_1, f_2, \ldots, f_{j+1}\}$. The adversary presents a $C_{F_{j+1}}(\ell_{j+1})$ starting at $L_{j+1} = R'_j$. This component will be placed completely inside the rightmost ℓ_j links of I_j. The last interval I_{j+1} of $C_{F_{j+1}}(\ell_{j+1})$ receives some color c. If $c \notin F$, the construction of $C_F(\ell)$ is finished. If $c \in F$, we can assume w.l.o.g. that $c = f_{j+2}$. This finishes round $j+1$.

After round $k-1$, either the construction has finished early and we are done, or the algorithm has used colors $f_1, f_2, \ldots, f_k$ on the intervals $I_0, I_1, \ldots, I_{k-1}$ that were the last intervals of the components $C_{F_j}(\ell_j)$ for $j = 0, \ldots, k-1$. In the latter case, let R_{k-1} be the rightmost link of I_{k-1}. Note that R_{k-1} is also contained in $I_0, \ldots, I_{k-2}$. The adversary presents an interval I_k with leftmost link R_{k-1}, length ℓ_k, and bandwidth $3^{R_{k-1}} - \beta_L$. Note that I_k is in conflict with $I_0, \ldots, I_{k-1}$ on link R_{k-1}, as each of $I_0, \ldots, I_{k-1}$ has bandwidth at least $3^L - 3^{L-1} = 3^L - \beta_L > \beta_L$. Therefore, the algorithm $\mathcal{A}$ must assign a color

outside F to I_k, and the construction of $C_F(\ell)$ is finished. ℓ_k is chosen in such a way that the interval I_k extends ℓ links further to the right than any of the previous intervals presented as part of this component $C_F(\ell)$. Note that no other interval (other than I_k) from this $C_F(\ell)$ overlaps the rightmost ℓ links of I_k. Let R_k be the rightmost link of I_k, and let $R'_k = R_k - \ell + 1$. The total bandwidth of intervals from this $C_F(\ell)$ that overlap the rightmost ℓ links of I_k is equal to the bandwidth of I_k, which is less than $3^{R'_k-1} - 3^{L-1}$. Therefore, Invariant 2 is satisfied for this $C_F(\ell)$.

As we know that previously presented intervals of total bandwidth at most 3^{L-1} contain the link L (by Invariant 1), we can conclude that the total bandwidth of intervals overlapping the rightmost ℓ links of I_k is bounded by $3^{L-1} + 3^{R'_k-1} - \beta_L = 3^{R'_k-1}$, so that Invariant 1 continues to hold for components placed at R'_k or further to the right. One can also show that the construction of $C_F(\ell)$ ensures that Invariant 2 is maintained. Furthermore, it is clear that the component $C_F(\ell)$ forces the algorithm to use a color outside the set F.

The length $g(\ell, k)$ of the part of the line that is needed to place a $C_F(\ell)$, for $\ell > 0$, with $|F| = k$ can be calculated to be $g(\ell, k) = \ell + 1$ for $k = 0$ and $g(\ell, k) = a_k(\ell + 1) - 1$ for $k > 0$. Here, the sequence a_n for $n \geq 0$ is defined by $a_0 = 1$ and $a_{n+1} = 1 + \prod_{i=0}^{n} a_i$. We have $a_0 = 1$, $a_1 = 2$, $a_2 = 3$, $a_3 = 7$, $a_4 = 43$, etc. This sequence is known as Sylvester's sequence or the sequence of Euclid numbers. For $n \geq 1$, it satisfies $a_{n+1} = a_n^2 - a_n + 1$. It is known that $a_n = \lfloor c^{2^{n-1}} \rfloor + 1$, where $c \approx 1.59791$ (see [18], sequences A000058 and A007018).

Next, we consider the optimal coloring of $C_F(\ell)$. Consider a $C_F(\ell)$ placed at some link L. Let R be the rightmost link of its last interval I. Let $R' = R - \ell + 1$ be the link at which later components could potentially be placed. Call the set of intervals from $C_F(\ell)$ that contain R' and are different from I the *siblings* of I. We can prove by induction on the size of F that every $C_F(\ell)$ can be colored with 2 colors in such a way that all intervals from the $C_F(\ell)$ containing R' (these are the last interval of $C_F(\ell)$ and its siblings) are assigned the same color. Furthermore, the coloring is such that in each of the two color classes, there is a free capacity of at least 3^{L-1} on all links of the component.

For any $k \geq 1$, we can let $F = \{1, 2, \ldots, k-1\}$ and place a $C_F(1)$ starting at link 1. By the discussion above, the on-line algorithm $\mathcal{A}$ uses a color $\geq k$ on this instance, while the optimum can color all intervals with 2 colors. This shows that $\mathcal{A}$ cannot have competitive ratio better than $k/2$. As k is arbitrary, we obtain the following theorem. Note that the number of links needed to place a $C_F(1)$ is $g(1, k) = 2a_k - 1 = 2(\lfloor c^{2^{k-1}} \rfloor + 1) - 1$, where $c \approx 1.59791$. Thus $k = \Theta(\log \log n)$, where n is the length of the line, and $k = \Theta(\log \log \log c_{\max})$, since the capacity of link i is 3^i.

Theorem 4. *There is no deterministic on-line algorithm for capacitated interval coloring with non-uniform capacities with constant competitive ratio. Moreover, the competitive ratio of any deterministic on-line algorithm for the problem is at least* $\Theta(\log \log n)$ *for lines of length* n *and at least* $\Theta(\log \log \log c_{\max})$ *for lines with maximum edge capacity* $c_{\max}$ *and minimum edge capacity* 1.

References

1. Adamy, U., Erlebach, T.: Online coloring of intervals with bandwidth. In: Solis-Oba, R., Jansen, K. (eds.) WAOA 2003. LNCS, vol. 2909, pp. 1–12. Springer, Heidelberg (2004)
2. Adler, R., Azar, Y.: Beating the logarithmic lower bound: Randomized preemptive disjoint paths and call control algorithms. In: SODA'99. Proceedings of the 10th Annual ACM–SIAM Symposium on Discrete Algorithms, pp. 1–10. ACM Press, New York (1999)
3. Azar, Y., Fiat, A., Levy, M., Narayanaswamy, N.: An improved algorithm for online coloring of intervals with bandwidth. Theoretical Computer Science 363(1), 18–27 (2006)
4. Bansal, N., Chakrabarti, A., Epstein, A., Schieber, B.: A quasi-PTAS for unsplittable flow on line graphs. In: STOC'06. Proceedings of the 38th Annual ACM Symposium on Theory of Computing, pp. 721–729. ACM Press, New York (2006)
5. Chakrabarti, A., Chekuri, C., Gupta, A., Kumar, A.: Approximation algorithms for the unsplittable flow problem. In: Jansen, K., Leonardi, S., Vazirani, V.V. (eds.) APPROX 2002. LNCS, vol. 2462, pp. 51–66. Springer, Heidelberg (2002)
6. Chekuri, C., Mydlarz, M., Shepherd, F.: Multicommodity demand flow in a tree. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 410–425. Springer, Heidelberg (2003)
7. Chrobak, M., Ślusarek, M.: On some packing problems relating to dynamical storage allocation. RAIRO Journal on Information Theory and Applications 22, 487–499 (1988)
8. Epstein, L., Levy, M.: Online interval coloring and variants. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 602–613. Springer, Heidelberg (2005)
9. Epstein, L., Levy, M.: Online interval coloring with packing constraints. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 295–307. Springer, Heidelberg (2005)
10. Gyárfás, A., Lehel, J.: On-line and first-fit colorings of graphs. Journal of Graph Theory 12, 217–227 (1988)
11. Jensen, T.R., Toft, B.: Graph coloring problems. Wiley, Chichester (1995)
12. Kierstead, H.A.: The linearity of first-fit coloring of interval graphs. SIAM Journal on Discrete Mathematics 1(4), 526–530 (1988)
13. Kierstead, H.A., Qin, J.: Coloring interval graphs with First-Fit. SIAM Journal on Discrete Mathematics 8, 47–57 (1995)
14. Kierstead, H.A., Trotter, W.T.: An extremal problem in recursive combinatorics. Congressus Numerantium 33, 143–153 (1981)
15. Narayanaswamy, N.S.: Dynamic storage allocation and online colouring interval graphs. In: Chwa, K.-Y., Munro, J.I.J. (eds.) COCOON 2004. LNCS, vol. 3106, pp. 329–338. Springer, Heidelberg (2004)
16. Pemmaraju, S.V., Raman, R., Varadarajan, K.R.: Buffer minimization using max-coloring. In: SODA'04. Proceedings of 15th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 562–571. ACM Press, New York (2004)
17. Raghavan, P., Upfal, E.: Efficient routing in all-optical networks. In: STOC'94. Proceedings of the 26th Annual ACM Symposium on Theory of Computing, pp. 134–143. ACM Press, New York (1994)
18. Sloane, N.J.A.: On-line encyclopedia of integer sequences (1996–2007) Available on-line at http://www.research.att.com/~njas/sequences/Seis.html

Energy Efficient Heuristic Scheduling Algorithms for Multimedia Service*

Sungwook Kim[1] and Sungchun Kim[2]

[1] Department of Computer Science, Sogang University,
Shinsu-dong 1, Mapo-ku, Seoul, 121-742, South Korea
swkim01@sogang.ac.kr
[2] Department of Computer Science, Sogang University,
Shinsu-dong 1, Mapo-ku, Seoul, 121-742, South Korea
ksc@mail.sogang.ac.kr

Abstract. In this paper, we propose new adaptive processor control algorithm to manage various multimedia communications. The most important feature of our proposed technique is its adaptability, flexibility and responsiveness to current system conditions. Simulation results clearly indicate that our scheme maintains well-balanced system performance considering all conflicting criteria.

1 Introduction

Heterogeneous multimedia data service can be categorized into two classes according to the required Quality of Service (QoS): class I (real-time) traffic services and class II (non real-time) traffic services. Class I data traffic is highly delay sensitive. Based on different tolerance characteristics, class I data type has higher priority than class II data type during system operations [1]-[5].

A processor is one of the most significant energy-consuming components of the system [4],[6]. Therefore, the processor management becomes a key factor in enhancing system performance. Nowadays, a lot of research effort has gone into the development of efficient processor management techniques [1]-[8]. Recently, the Dynamic Voltage Scaling (DVS) technique [4],[6]-[8] has been investigated and developed to improve energy efficiency. This technique dynamically changes supply voltage and speed of processor at runtime. However, earlier research on DVS algorithms assumes that complete knowledge about the requested services is available *a priori*, which is unrealistic and impractical.

Motivated by the above discussion, we propose new processor management algorithms for multimedia data services. Main design goal of our proposed algorithms is to maximize system performance while maintaining energy efficiency as possible.

* This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD, Basic Research Promotion Fund) (KRF-2006-331-D00475) and was supported by the Sogang University, under the 2006 new faculty support program and was supported by the Brain Korea 21 Project in 2007.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 255–259, 2007.

Earlier work reported in [1]-[3],[8] has also considered processor management algorithms. These schemes dynamically control system processor to save energy based on some observations of the current workload situations. Compared to these existing schemes [1]-[3],[8], our proposed online scheme provides excellent system performance under different system load distributions.

2 Proposed Online Management Algorithm

While operating systems, we make a decision on how to efficiently provide appropriate grade of service level by bandwidth adjustment. At this time, a bandwidth adaptation technique is needed to possibly reduce requested or already connected call bandwidth allocation. With multiple grades of service quality, bandwidth allocation of existing individual connections can be dynamically adjusted. However, class I data type requests - time critical real time applications, which do not have delay tolerance - are designed to be transmitted at a fixed bandwidth with a specified grade of service. Class II data type requests can have some delay latency because of human tolerance to degradation in quality as long as it is above a certain minimum acceptable threshold. The bandwidth adaptation strategy is applicable only to class II data connections.

A multimedia call request is defined as a request for bandwidth (including a minimum bound on bandwidth requirements), along with a priority level. Bandwidth allocation is a discrete quantity in terms of the number of basic units (BUs). Therefore, all calls are allocated bandwidth from the discrete set B = $\{ b_{min}, b_{min+1}, b_{min+2} \cdots b_{max} \}$ where $b_{min+i} < b_{min+(i+1)}$, b_{min} is the minimum bound and b_{max} is the maximum bound for bandwidth allocation.

In the mixed *class I* and *class II* heterogeneous multimedia services, delay sensitive *class I* services should be executed immediately with a fixed communication speed. However, delay tolerant *class II* services need only to guarantee its deadline. Therefore, between its arrival time and the deadline, the *class II* services are amenable to adaptation with variable communication speeds while satisfying each requirement.

The processor has different power states based on various speed assignments. Each power state of the processor is characterized by different performance and energy consumption. It satisfies the physical law [1],[4] that energy is reduced in direct proportion to the processor power state. Therefore, it is more energy efficient to slow down the processor power as much as possible.

As mentioned earlier, one promising power and energy reduction technique is voltage control. However, there is an associated control overhead for dynamic processor power/speed changes. Whenever a speed transition takes place, extra energy overhead is incurred [1]-[3]. Therefore, energy saving by the power control mechanism causes potential transition penalty. The control energy overhead of each processor power state transition is assumed to be a fixed energy amount (T_e) [3],[7]. That is, for $i < j < k$ processor power/speed states, the cost to go from i to j and then from j to k is the same as the cost of going from i directly up to k. Our power management strategy tries to minimize E for providing efficient system performance. As mentioned earlier, heterogeneous services make the control problem more complex. Our main goal here

is to maximize system energy efficiency while ensuring adaptive multimedia services. In order to provide system efficiency and the ability to satisfy the QoS control simultaneously, our proposed scheme makes control decisions in a flexible online manner in order to strike appropriate system performance.

In contrast to a *class I* service, the start time and processing rate of a *class II* service can be dynamically changed. Therefore, when service requests continually arrive to be processed, the system should schedule the requested workload. In this paper, we design an adaptive online service scheduling algorithm for this purpose. Due to the partial knowledge of the input requests and future uncertainty, our scheduling algorithm is dynamically adjustable and improves the responsiveness of the current situations by reacting to system changes in real time.

In order to minimize the transition overhead for processor power/speed changes, we dynamically reschedule the system load to be balanced over time. Therefore, when a new request enters the system or a running service is completed, our proposed scheduling algorithm tries to keep constant processor speed if possible. Based on the combination of the online scheduling technique and dynamic power management, we try to minimize power consumption while maximizing the system performance. Our strategy for service scheduling is summarized in the following.

Design challenge 1. For energy saving, we try to maintain the processor power state as low as possible while avoiding the adverse effect of running too slow to meet the required demand.

Design challenge 2. Due to the state transition overhead of dynamic power management, constant processor speed by load balancing over time is more energy efficient than frequent speed changes. We try to reduce speed changes to the extent possible.

To implement our energy-efficient service scheduling idea, we try to minimize the peak processor speed (PP_s). When a service request arrives or a service is completed, our online scheduling algorithm is responsible for scheduling and adapting the *class II* service based on the current system situations. To maintain a constant processor speed, our online scheduling strategy is to minimize the difference between the current processor power/speed state ($S_p(c_t)$) and PP_s. Therefore, we adaptively reschedule *class II* services for adaptive processor assignment. There are two kinds of adaptive processor assignment techniques: for degradation and for upgradation. When the degradation (upgradation) policy is applied to the processor at the current time, the assigned processor capacity ($PS_i(c_t)$) for running *class II* services can be decreased (increased). Therefore, if $S_p(c_t) > PP_s$ ($S_p(c_t) < PP_s$), we degrade (upgrade) the $PS_i(c_t)$ of running *class II* services to make $S_p(c_t)$ close to PP_s. This adaptive rescheduling technique based on real time feedback tries to balance the system load between the current and future times. Therefore, our service scheduling algorithm can avoid abrupt power/speed state changes over time as much as possible. This constant $S_p(c_t)$, which is defined as the expected processor power/speed state ($E_{pro}(c_t)$), reduces the extra transition overhead of processor power/speed fluctuations for minimizing the total energy consumption.

3 Simulation Experiments

In this section, we evaluate the performance of our proposed scheme using a simulation model. Based on this simulation model, we compare the performance of our scheme with other existing schemes [1]-[3],[8]. The existing schemes [1]-[3],[8] are also designed to control processor adaptively. However, there are some disadvantages. They do not consider service prioritization based on different QoS requirements. Therefore, under a heavy system load situation, these schemes can not provide performance assurance for higher priority services. In addition, these schemes are designed for a specific system performance parameter. Therefore, they cannot maintain well-balanced performance among conflicting criteria under widely different system load distributions. In Fig. 1 - Fig. 2, we present simulation results by comparing the performance of our scheme with the OPMMS, the APMLC, the DSSPS and the OSVVP schemes [1]-[3],[8].

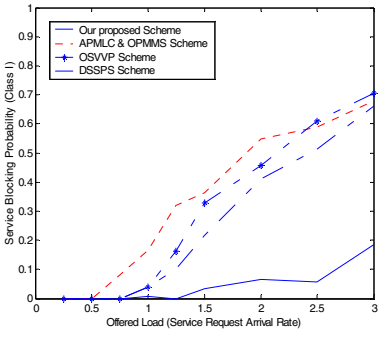


Fig. 1. CBP_{class_I} of service requests

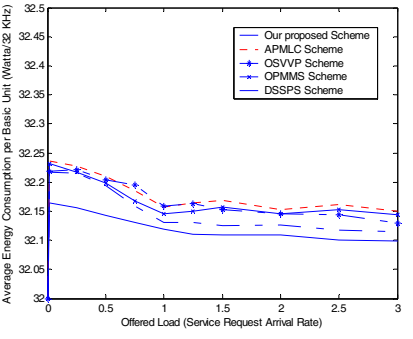


Fig. 2. Average power consumption

Fig. 1 shows the CBP_{class_I} of each scheme. For low service request arrival rates (λ<0.5), all the schemes have enough processor capacity to accept new service requests. Therefore, the CBP_{class_I} of all the schemes is identical. As λ increases, the amount of unused processor capacity decreases. So, new requests are likely to be rejected and CBP_{class_I} increases. However, due to the advantage of our online QoS control strategy, our proposed scheme attains excellent performance for higher priority (*class I*) service requests. The curves in Fig. 2 indicate the average power consumption per basic unit (Watts/32 Kbps) in the system. From the simulation results we obtained, it can be seen that all the schemes have similar trends. However, based on our service scheduling strategy, our proposed scheme can reduce transition energy overhead. Therefore, our scheme performs more efficiently than the other existing schemes from low to heavy system load distributions.

4 Summary and Conclusions

In this paper, we proposed new processor management algorithms for heterogeneous multimedia services. The main novelty of our approach is its adaptability, flexibility

and responsiveness to current system conditions. This feature is highly desirable for real time system management. Performance evaluation results indicate that our scheme maintains well-balanced performance between contradictory QoS requirements in widely different system load intensities while other existing schemes can not offer such an attractive trade off.

References

1. Ramanathan, D., Irani, S., Gupta, R.K.: An Analysis of System Level Power Management Algorithms and Their Effects on Latency. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems 21(3), 291–305 (2002)
2. Irani, S., Shukla, S., Gupta, R.: Algorithms for Power Savings. In: Symposium on Discrete Algorithms, pp. 37–46 (2003)
3. Irani, S., Shukla, S., Gupta, R.: Online Strategies for Dynamic Power Management in Systems with Multiple Power-Saving States. HYPERLINK 2(3), 325–346 (2003)
4. Govil, K., Chan, E., Wasserman, H.: Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU. In: International Conference on Mobile Computing and Networking, pp. 13–25 (1995)
5. Hwang, C.-H., Wu, A.C.-H.: A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation. In: ICCAD'97. International conference on Computer-aided design, pp. 28–32 (1997)
6. Pillai, P., Shin, K.G.: Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems. In: SOSP'2001. Proceedings of the 18th ACM Symposium on Operating System Principles, pp. 89–102 (2001)
7. Mochocki, B., Hu, X.S., Quan, G.: A Realistic Variable Voltage Scheduling Model for Real-Time Applications. In: ICCAD-02. International Conference on Computer Aided Design, pp. 726–731 (2002)
8. Hong, I., Potkonjak, M., Srivastava, M.B.: On-Line Scheduling of Hard Real- Time Tasks on Variable Voltage Processor. In: ICCAD-98. International Conference on Computer Aided Design, pp. 653–656 (1998)
9. Burd, T.D., Brodersen, R.W.: Processor design for portable systems. Journal of VLSI Signal Processing 13, 203–222 (1996)

Call Control and Routing in SONET Rings*

Shuqiang Chen and Qizhi Fang**

Department of Mathematics, Ocean University of China
Qingdao 266071, Shandong, P.R. China
qfang@ouc.edu.cn

Abstract. In this paper, we provide a polynomial-time approximation algorithm for Call Control and Routing problems in SONET rings. In this problem, we are given a SONET ring and a set of calls, each of which is described by a source-destination pair of nodes together with an integer specifying the call demand, the aim is to devise a routing scheme such that the total demand transmitted is maximum subject to the bandwidth restriction. We first give an $\mathcal{NP}$-hardness proof for this problem. Then a polynomial-time approximation algorithm is provided. When $d_{max} \leq \frac{1}{K}d^*$ (where $K > 2$ is a constant, d^* is the available bandwidth of the ring and d_{max} is the largest call demand among all the calls), the algorithm outputs a routing scheme with total demand transmitted at least as $(1 - \frac{7}{2K+3})$ times the optimum.

Keywords: SONET ring, call, routing, $\mathcal{NP}$-hard, approximation algorithm.

1 Introduction

Nowadays, the Synchronous Optical Network (SONET) has been adopted by many network service providers as faster, more efficient and less expensive transport technology [6]. The nodes on a SONET network send and receive messages via add/drop multiplexers (ADMs), and the actual bandwidth available on the network is determined by the capacity of these multiplexers. In this environment, there is an additional hardware cost to support the desired level of bandwidth of the network, even through the fiber can offer virtually unlimited bandwidth. Thus, many optimization problems concerning the efficient operation on data transmission have been proposed and received considerable research attention.

A SONET ring usually consists of two working counter rotated fiber rings carrying the data stream in opposite directions, which is appealing for both its implicity and self-healing properties. In a SONET ring, a communication request (or *call*) is characterized by a source-destination pair and a demand (the size of the call), where the source originating data stream to be sent to the destination. Each call requires a routing, i.e., a virtual path to be established for its transmission. Therein, a well studied optimization problem is *load-balanced routing*

* Research is supported by NCET(No. 05-098), NSFC(No. 10371114).
** Corresponding author.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 260–270, 2007.

problem. The aim of this problem is to devise a routing scheme for a set of calls in the ring, so that the bandwidth required to transmit all the calls is minimized. In practice, however, the available bandwidth offered is limited. Consequently, the number of necessary bandwidth, even if minimized, may exceed the number of available bandwidth; in such case only a fraction of communication requests can be satisfied without delay. It is, therefore, meaningful to employ algorithmic technique in order to satisfy as many requests as possible with the available bandwidth. The problem derived from this background is called *Call Control and Routing* problem in SONET rings (CCR-Ring), which will is the focal point of this paper.

In detail, the problem of CCR-Ring is to find, for a given set of calls, a subset of calls and their routing paths such that the total demand of accepted calls is maximum subject to the bandwidth capacity restriction.

There are several optimization problems closely related to the problem of CCR-Ring. One is the load-balanced routing problem, which has been extensively studied in [3,5,8,9,10]. This problem is in general $\mathcal{NP}$-hard. While when all calls have unit demand, Wilfong and Winkler [9] presented an interesting polynomial time optimal algorithm. Wan and Yang [10] further presented a polynomial time approximation scheme (PTAS) when the calls possess different demands. Another related problem is *Call Control* problem. In this problem, the calls are given together with their routing paths (prescribed routing), and the objective is to compute a subset of the paths with maximum total call demands such that no edge capacity is violated. Adamy, et al., [1] gave a polynomial time algorithm to solve the Call Control problem in ring network optimally when all calls have unit demand.

As for the problem of CCR-Ring, there have been some results obtained with different algorithmic techniques. Sai and Erlebach [7] considered the CCR-Ring problem when each call has unit demand in undirected rings. They presented an approximation algorithm achieving an objective value only smaller than optimum by 3, as well as a PTAS. Chekuri, Mydlarz and Shepherd [2] studied a more general model where each call has a profit value and links may have different capacities. In their work, a $(2+\epsilon)$-approximation algorithm was proposed, which also derives a performance ratio $1+\frac{1}{\sqrt{K}}$ when the demands of the calls are bounded by $1/K$ times the minimum edge capacity. The main contribution of this work is providing a different approximation algorithm for CCR-Ring problems. We are motivated by the "rounding technique" discussed in [8,9,10], and give a deep application of it. When $d_{max} \leq \frac{1}{K}d^*$, where $K > 2$ is a constant, d^* is the bandwidth capacity of the ring and d_{max} is the largest call demand among all the calls, our algorithm can output a routing scheme with the total demand transmitted at least as $(1-\frac{7}{2K+3})$ times the optimum.

In practice, the calls with large demand may usually be divided into several parts to be transmitted independently so that the demand of each part is smaller, and consequently, the call demand is in general much smaller than the

ring capacity (i.e., the value of K is large). Therefore, our algorithm has good performance in practice.

The remainder of this paper is organized as follows. In the next section, we introduce some preliminaries and the integer program formulation of the CCR-Ring problem. Moreover, an $\mathcal{NP}$-hardness result is obtained. Section 3 is dedicated to an approximation algorithm for CCR-Ring problems. Finally in Sect. 4, we conclude with some remarks.

2 Preliminaries and Model

Throughout this paper, we assume that a SONET ring G is a bi-directed ring consisting of n nodes labeled 0 through $n-1$ in a clockwise manner. All arithmetic involving nodes will be done implicitly using modulo n operations. The link set of G is $\{(i, i+1), (i+1, i) : i = 0, 1, \cdots, n-1\}$, where $(i, i+1)$ and $(i+1, i)$ are the clockwise and counter-clockwise links between nodes i and $i+1$, respectively. In our model, we further assume that a positive integer d^* is given to represent the available bandwidth in the ring, called the ring capacity.

A call $C = \langle s, t; d\rangle$ is an ordered pair of nodes $s, t \in V$ completed by an integer $d > 0$ specifying the call demand, where s is the source originating the data stream to be sent to the destination t. Given a set of calls in the SONET ring G, denoted by $\mathcal{C} = \{C_i = \langle s_i, t_i; d_i\rangle : i = 1, 2, \cdots, m\}$, we let P_i^+ and P_i^- denote the clockwise and counter-clockwise paths from s_i to t_i respectively, and $d_{max} = \max\{d_i : i = 1, 2, \cdots, m\}$. A *routing* of $\mathcal{C}$ can be written as a set $\mathcal{P} = \{P_i : i = 1, 2, \cdots, m\}$ with $P_i \in \{P_i^+, P_i^-, \emptyset\}$, where P_i^+, P_i^- and $\emptyset$ represent three choices made for the call C_i: transmit clockwise, counter clockwise and not transmit, respectively. A routing is called *feasible* if the total demand transmitted through each link does not exceed the ring capacity d^*. The total demand transmitted by a routing is called its *throughput*.

Then Call Control and Routing problem in a SONET ring network can be stated formally as follows:

CALL CONTROL AND ROUTING IN SONET RING (CCR-Ring):
INSTANCE : Given a bi-directed ring G with n nodes and a ring capacity d^*, a set of calls $\mathcal{C} = \{C_i = \langle s_i, t_i; d_i\rangle : i = 1, 2, \cdots, m\}$.

QUESTION : Find a feasible routing $\mathcal{P} = \{P_i : i = 1, 2, \cdots, m\}$ such that the throughput $\sum\{d_i : P_i \neq \emptyset$ and $P_i \in \mathcal{P}\}$ is maximum.

The instance of CCR-Ring will be denoted briefly by $\{n; d^*; \mathcal{C} = \{C_i = \langle s_i, t_i; d_i\rangle : i = 1, 2, \cdots, m\}\}$. To make the CCR-Ring a decision problem as in [4], we append a target value K^* to the instance and ask whether there is a feasible routing $\mathcal{P}$ with throughput no less than K^*?

The $\mathcal{NP}$-hardness proof is in much spirit as that of Wan and Yang's work [10]. For the sake of briefness, we denoted by $[s, t)$ the node sequence $\{s, s+1, \cdots, t-1\}$, and by $[s, t]$ the node sequence $\{s, s+1, \cdots, t\}$. Two calls $C_i = \langle s_i, t_i; d_i\rangle$ and $C_j = \langle s_j, t_j; d_j\rangle$ are said to be *parallel* if either $[s_i, t_i] \subseteq [s_j, t_j]$ or $[s_j, t_j] \subseteq [s_i, t_i]$; otherwise, they are said to be *crossing*.

Theorem 1. *The decision problem of CCR-Ring is $\mathcal{NP}$-complete even with the following patterns of calls:*

1) All calls share a common source;

2) Each pair of calls are crossing;

3) Each pair of calls are parallel but do not share common source and destination.

Proof. The technique we use here is a polynomial transformation from a basic $\mathcal{NP}$-complete problem, PARTITION [4]. In an instance of PARTITION, we are given m positive integers $d_1, d_2, \cdots, d_m$, and the question is whether we can divide them into two groups of equal sum. From this, we construct three different instances of the decision problem of CCR-Ring.

Case 1. Set $n = m+1$ and give a bi-direction ring with nodes set $\{0, 1, \cdots, n-1\}$ labeled clockwise. There are m calls $C_1, C_2, \cdots, C_m$. For C_j $(j = 1, 2, \cdots, m)$, the source is node 0, the destination is node j and the demand is d_j.

Case 2. Set $n = 2m$ and give a bi-direction ring with nodes set $\{0, 1, \cdots, n-1\}$ labeled clockwise. There are m calls $C_1, C_2, \cdots, C_m$. For C_j $(j = 1, 2, \cdots, m)$, the source is node $j-1$, the destination is node $m+j-1$ and the demand is d_j.

Case 3. Set $n = 2m$ and give a bi-direction ring with nodes set $\{0, 1, \cdots, n-1\}$ labeled clockwise. There are m calls $C_1, C_2, \cdots, C_m$. For C_j $(j = 1, 2, \cdots, m)$, the source is node $j-1$, the destination is node $2m-j$ and the demand is d_j.

For each of the three cases, we define the ring capacity $d^* = \frac{1}{2}\sum_{j=1}^{m} d_j$ and $K^* = \sum_{j=1}^{m} d_j$. This guarantees that the throughput achieves K^* if and only if all the m calls are all transmitted. It is easy to verify that in all cases, the answer to the instance of PARTITION is YES if and only if there is a feasible routing with throughput K^* for the CCR-Ring instances, as desired. ∎

In the rest of this section we give an integer program formulation for the CCR-Ring problem. First let us introduce two sets of variables to represent a routing of $\mathcal{C}$. For $i = 1, 2, \cdots, m$, we set

$$x_i = \begin{cases} 1 & \text{if the call } C_i \text{ is chosen and transmitted clockwise} \\ 0 & \text{otherwise.} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if the call } C_i \text{ is chosen and transmitted counter-clockwise} \\ 0 & \text{otherwise.} \end{cases}$$

Let $x = (x_1, x_2, \cdots, x_m)$ and $y = (y_1, y_2, \cdots, y_m)$. Then such defined (x, y) corresponds to a routing of $\mathcal{C}$, and this notation will be used in the following.

Given a routing (x, y), the total demand transmitted through links $(k, k+1)$ and $(k+1, k)$, called the *link loads*, are written as

$$L_k^+(x, y) = \sum_{i:k \in [s_i, t_i)} d_i x_i \quad \text{and} \quad L_k^-(x, y) = \sum_{i:k \notin [s_i, t_i)} d_i y_i,$$

respectively $(k = 0, 1, \cdots, n-1)$. A routing (x, y) is feasible if and only if $L_k^+(x, y) \le d^*$ and $L_k^-(x, y) \le d^*$ for all $k \in \{0, 1, \cdots, n-1\}$. The throughput of the routing (x, y) is denoted by $T(x, y)$.

Let $M = \{m_{ij}\}_{n\times m}$ and $N = \{n_{ij}\}_{n\times m}$ are two link-path adjacent matrices, where

$$m_{ij} = \begin{cases} 1 & \text{link } (i, i+1) \text{ belongs to path } P_j^+ \\ 0 & \text{otherwise.} \end{cases}$$
$$n_{ij} = \begin{cases} 1 & \text{link } (i+1, i) \text{ belongs to path } P_j^- \\ 0 & \text{otherwise.} \end{cases}$$

Then the CCR-Ring problem can be formulated as the following $\{0, 1\}$-program:

$$\max T(x, y) = \sum_{i=1}^{m}(d_i x_i + d_i y_i)$$
$$\text{s.t.} \begin{cases} M(d_1 x_1, d_2 x_2, \cdots, d_m x_m)^\tau \le \mathbf{d}^* \\ N(d_1 y_1, d_2 y_2, \cdots, d_m y_m)^\tau \le \mathbf{d}^* \\ x_i + y_i \le 1 \qquad i = 1, 2, \cdots, m \\ x_i, y_i = 0, 1 \qquad i = 1, 2, \cdots, m \end{cases} \tag{1}$$

where $\mathbf{d}^*$ is an n-dimensional vector with each component being d^*.

3 Approximation Algorithm

In this section we focus on a polynomial-time approximation algorithm for CCR-Ring problems. In order to explain the algorithm clearly, we first introduce some definitions and give a sketch of the algorithm.

Let $\{n; d^*; \mathcal{C} = \{C_i = \langle s_i, t_i; d_i\rangle, i = 1, 2, \cdots, m\}\}$ be an instance of CCR-Ring problem. For the convenience of description, we extend the concept of routing to a relaxed version. Let $x = (x_1, x_2, \cdots, x_m)$ and $y = (y_1, y_2, \cdots, y_m)$, we also call (x, y) a *(fractional) routing* of $\mathcal{C}$ if $\forall i = 1, 2 \cdots, m$: $x_i + y_i \le 1$ and $x_i, y_i \ge 0$. A routing (x, y) is called *split* if there is at least one call C_i satisfying $x_i \cdot y_i > 0$, i.e., it is transmitted partly clockwise and partly counter-clockwise; a routing (x, y) is called *semi-unsplit* if $x_i \cdot y_i = 0$ for each call C_i in $\mathcal{C}$, i.e., there is at most one possible transmission path for each call; a routing (x, y) is called *unsplit* if both x and y are integral, i.e., for each call in $\mathcal{C}$, either it is transmitted entirely in one direction or it is rejected. The main steps of the algorithm is as follows:

1. Solve the relaxed CCR-Ring problem to get an optimal fractional routing (x, y);
2. Transform (x, y) into a parallel routing $(\bar{x}, \bar{y})$, in which no parallel calls are both split;
3. Transform $(\bar{x}, \bar{y})$ into a semi-unsplit routing $(\tilde{x}, \tilde{y})$ by rounding technique;
4. By rounding technique again, obtain a final unsplit routing (x^*, y^*).

Let $K > 2$ be a constant and assume that $d_{max} \le \frac{1}{K} d^*$. Then our main result in this section is as follows.

Theorem 2. *For an instance of CCR-Ring* $\{n; d^*; \mathcal{C} = \{C_i = \langle s_i, t_i; d_i\rangle : i = 1, 2, \cdots, m\}\}$ *with* $d_{max} \le \frac{1}{K} d^*$ *$(K > 2)$, we can obtain, in polynomial time, a feasible unsplit routing* (x^*, y^*) *of* $\mathcal{C}$ *satisfying*

$$T(x^*, y^*) \ge (1 - \frac{7}{2K + 3}) T_{opt},$$

where T_{opt} *is the optimum throughput of the instance.*

3.1 Relaxed CCR-Ring Problem

Relaxing the constraints $x_i, y_i \in \{0,1\}$ to $0 \le x_i, y_i \le 1$ $(\forall\, i = 1, 2, \cdots, m)$ in (1), we obtain the relaxed CCR-Ring problem:

$$\max T(x,y) = \sum_{i=1}^{m}(d_i x_i + d_i y_i)$$
$$\text{s.t.} \begin{cases} M(d_1 x_1, d_2 x_2, \cdots, d_m x_m)^\tau \le \mathbf{d}^* \\ N\,(d_1 y_1, d_2 y_2, \cdots, d_m y_m)^\tau \le \mathbf{d}^* \\ x_i + y_i \le 1 \qquad i = 1, 2, \cdots, m \\ 0 \le x_i, y_i \le 1 \qquad i = 1, 2, \cdots, m \end{cases} \tag{2}$$

This is now a linear program that is solvable in polynomial time, and of course its optimal value, denoted by T^*_{LP}, satisfies $T^*_{LP} \ge T_{opt}$.

Given a feasible solution (x, y) of (2), a call $C_i \in \mathcal{C}$ is split by the routing (x, y) if it satisfies both $0 < x_i < 1$ and $0 < y_i < 1$; the routing (x, y) is called split if there is at least one call in $\mathcal{C}$ which is split by (x, y).

3.2 Transform to Parallel Routing

Recall that two calls $C_i = \langle s_i, t_i; d_i\rangle$ and $C_j = \langle s_j, t_j; d_j\rangle$ are said to be parallel if either $[s_i, t_i] \subseteq [s_j, t_j]$ or $[s_j, t_j] \subseteq [s_i, t_i]$; otherwise, they are said to be crossing. In particular, when either $s_i = s_j$ or $t_i = t_j$ occurs, the two calls are parallel. A routing (x, y) is said to be *parallel* if no two parallel calls are both split by (x, y). That is, in a parallel routing, any two split calls must be crossing, and therefore, cannot share common source and destination. In this subsection, the "unsplitting" technique is utilized to transform a fractional routing (x, y) into a parallel routing $(\bar{x}, \bar{y})$, which is similar as that discussed in [9,10].

Given a fractional routing (x, y), without loss of generality, we assume that there are two parallel calls C_i and C_j with $[s_i, t_i] \supseteq [s_j, t_j]$ both split by (x, y):

(a) if $d_i x_i + d_j x_j > d_i$, then we define a new routing $(\bar{x}, \bar{y})$ by setting

$$\begin{array}{ll} \bar{x}_i = 1, & \bar{y}_i = 0; \\ \bar{x}_j = x_j - \frac{d_i}{d_j}(1 - x_i), & \bar{y}_j = y_j + \frac{d_i}{d_j} y_i; \\ \bar{x}_k = x_k,\ \bar{y}_k = y_k & \forall k \ne i, j; \end{array}$$

(b) if $d_i x_i + d_j x_j \le d_i$, then we define a new routing $(\bar{x}, \bar{y})$ by setting

$$\begin{array}{ll} \bar{x}_i = x_i + \frac{d_j}{d_i} x_j, & \bar{y}_i = y_i - \frac{1}{d_i}\min\{d_i y_i, d_j x_j\}; \\ \bar{x}_j = 0, & \bar{y}_j = y_j + \frac{1}{d_i}\min\{d_i y_i, d_j x_j\}; \\ \bar{x}_k = x_k,\ \bar{y}_k = y_k\ \forall k \ne i, j. & \end{array}$$

It is easy to see that after the "unsplitting" procedure, at least one of the two calls is no longer split, and each link load is maintained or reduced. Moreover, the throughput remains unchanged. We may repeat the "unsplitting" procedure at most m times until no two parallel calls are both split.

Proposition 1. *Any fractional routing (x, y) can be transformed into a parallel routing $(\bar{x}, \bar{y})$ in polynomial time such that every link load does not increase and*

$$\sum_{i=1}^{m} d_i x_i = \sum_{i=1}^{m} d_i \bar{x}_i, \quad \sum_{i=1}^{m} d_i y_i = \sum_{i=1}^{m} d_i \bar{y}_i.$$

3.3 Rounding to Semi-unsplit Routing

In this subsection, we assume that $(\bar{x}, \bar{y})$ is an optimal parallel routing of the relaxed CCR-Ring problem. In this and the next subsection, we are much motivated by the "rounding technique" discussed in [8,9,10]. Here, the main idea is applying "rounding technique" to compel each of the split calls to be transmitted along only one possible direction, i.e., to obtain a semi-unsplit routing.

Since all crossing calls can not share common source and destination, the number of split calls (split by $(\bar{x}, \bar{y})$) is at most n (the size of the ring). For the sake of briefness, we first renumber the calls of $\mathcal{C}$. Assume that there are r ($r \leq n$) calls split by $(\bar{x}, \bar{y})$, say $\mathcal{C}_s = \{C_{l_1}, C_{l_2}, \cdots, C_{l_r}\}$. Denote

$$d^s_{max} = \max\{d_{l_i}(\bar{x}_{l_i} + \bar{y}_{l_i}) : i = 1, 2, \cdots, r\}.$$

Obviously, $d^s_{max} \leq d_{max}$. We renumber the r split calls as $C_1, C_2, \cdots, C_r$ ordered clockwise by their sources (or destinations). At the same time, for each call C_i, its source, destination and demand are also relabeled according to the renumbering, i.e., $C_i = \langle s_i, t_i; d_i \rangle$ (for each $i = 1, 2, \cdots, r$). The calls not in $\mathcal{C}_f$ are renumbered arbitrary as $C_{r+1}, C_{r+2}, \cdots, C_m$, respectively.

After renumbering, for any $k \in \{0, 1, \cdots, n-1\}$, the indices of the calls in $\mathcal{C}_s$ parallel to the clockwise link $(k, k+1)$ is an interval $[i_k, j_k] \subseteq \{1, 2, \cdots, r\}$; and accordingly, the indices of the calls in $\mathcal{C}_s$ parallel to the counter-clockwise link $(k+1, k)$ is just the complement of $[i_k, j_k]$, namely the interval $[j_k + 1, i_k - 1] \subseteq \{1, 2, \cdots, r\}$. Here the intervals $[i_k, j_k]$ and $[j_k + 1, i_k - 1]$ are interpreted under modulo r operations.

We first transform the parallel routing $(\bar{x}, \bar{y})$ to a semi-unsplit routing $(\bar{x}', \bar{y}')$ which can be carried out recursively as follows:

$$\bar{x}'_j = \begin{cases} \bar{x}_j + \bar{y}_j \text{ if } \left| -d_j \bar{x}_j + \sum_{h=1}^{j-1} d_h(\bar{x}'_h - \bar{x}_h) \right| > \left| d_j \bar{y}_j + \sum_{h=1}^{j-1} d_h(\bar{x}'_h - \bar{x}_h) \right| \\ 0 \qquad \text{otherwise} \end{cases} \tag{3}$$

$$\bar{y}'_j = (\bar{x}_j + \bar{y}_j) - \bar{x}'_j \qquad j = 1, 2, \cdots, r.$$

Lemma 1. *For the routing $(\bar{x}', \bar{y}')$ constructed by (3), every partial sum satisfies*

$$\textstyle\sum_{h=1}^{j} d_h(\bar{x}'_h - \bar{x}_h) \in \left[-\frac{1}{2}d^s_{max}, \frac{1}{2}d^s_{max}\right) \quad \forall\, 1 \leq j \leq r.$$

The proof of this lemma can be given by induction on j, which is omitted.

Proposition 2. *The semi-unsplit routing $(\bar{x}', \bar{y}')$ constructed by (3) satisfies that $T(\bar{x}', \bar{y}') = T(\bar{x}, \bar{y})$ and the increment of each link load is no more than $\frac{3}{2}d^s_{max}$.*

Proof. The equality $T(\bar{x}', \bar{y}') = T(\bar{x}, \bar{y})$ follows directly from the construction of $(\bar{x}', \bar{y}')$. Since only the split calls parallel to a link affect its link load, for each $k = 0, 1, \cdots, n-1$, we have

$$L^+_k(\bar{x}', \bar{y}') = L^+_k(\bar{x}, \bar{y}) + \textstyle\sum_{h \in [i_k, j_k]} d_h(\bar{x}'_h - \bar{x}_h),$$
$$L^-_k(\bar{x}', \bar{y}') = L^-_k(\bar{x}, \bar{y}) + \textstyle\sum_{h \notin [i_k, j_k]} d_h(\bar{x}'_h - \bar{x}_h).$$

Based on Lemma 1, if $i_k \le j_k$ then

$$\begin{aligned} L_k^+(\bar{x}', \bar{y}') - L_k^+(\bar{x}, \bar{y}) &= \textstyle\sum_{h=1}^{j_k} d_h(\bar{x}'_h - x_h) - \sum_{h=1}^{i_k-1} d_h(\bar{x}'_h - x_h) \\ &< \tfrac{1}{2}d^s_{max} - (-\tfrac{1}{2}d^s_{max}) = d^s_{max}; \end{aligned}$$

and if $i_k > j_k$ then

$$\begin{aligned} & L_k^+(\bar{x}', \bar{y}') - L_k^+(\bar{x}, \bar{y}) \\ &= \textstyle\sum_{h=1}^{r} d_h(\bar{x}'_h - \bar{x}_h) + \sum_{h=1}^{j_k} d_h(\bar{x}'_h - \bar{x}_h) - \sum_{h=1}^{i_k-1} d_h(\bar{x}'_h - \bar{x}_h) \\ &< \tfrac{1}{2}d^s_{max} + \tfrac{1}{2}d^s_{max} - (-\tfrac{1}{2}d^s_{max}) = \tfrac{3}{2}d^s_{max}. \end{aligned}$$

Therefore, $L_k^+(\bar{x}', \bar{y}') - L_k^+(\bar{x}, \bar{y}) < \frac{3}{2}d^s_{max}$. A symmetric argument for the counter-clockwise link $(k+1, k)$ derives $L_k^-(\bar{x}', \bar{y}') - L_k^-(\bar{x}, \bar{y}) < \frac{3}{2}d^s_{max}$ as well. ∎

We note that the semi-unsplit routing $(\bar{x}', \bar{y}')$ may not satisfy the ring capacity restriction. On the other hand, in the final step of our algorithm (in the next subsection), the "rounding technique" will be used again which will also lead to some increment on link loads. Therefore, it is a nature idea to "round down" the routing $(\bar{x}', \bar{y}')$ so as to get a routing with smaller link loads. Of course, the "rounding down" procedure is companied with the decreasing of the routing throughput.

Define $(\tilde{x}, \tilde{y}) = \lambda^*(\bar{x}', \bar{y}')$, that is,

$$\tilde{x}_i = \lambda^* \bar{x}', \quad \tilde{y}_i = \lambda^* \bar{y}', \qquad \forall i = 1, 2, \cdots, m \tag{4}$$

where λ^* is defined as

$$\lambda^* = \frac{d^* - 2d_{max}}{\max_k\{L_k^+(\bar{x}', \bar{y}'), L_k^-(\bar{x}', \bar{y}')\}}.$$

It follows directly from Proposition 2 that

$$\lambda^* \ge (d^* - 2d_{max})/(d^* + \tfrac{3}{2}d_{max}).$$

Combined Proposition 2 with the definition of $(\tilde{x}, \tilde{y})$, the following result can be derived directly.

Proposition 3. *The semi-unsplit routing $(\tilde{x}, \tilde{y})$ constructed by (4) satisfies that each link load is no more than $d^* - 2d_{max}$ and $T(\tilde{x}, \tilde{y}) = \lambda^* T(\bar{x}, \bar{y})$.*

3.4 Rounding to an Unsplit Routing

Now we are in the position to transform the semi-unsplit routing $(\tilde{x}, \tilde{y})$ into an unsplit feasible routing. Here for each call C_i in $\mathcal{C}$, at least one of $\tilde{x}_i$ and $\tilde{y}_i$ must be zero. Denote

$$\begin{aligned} \mathcal{C}_I &= \{C_i \in \mathcal{C} : 0 < \tilde{x}_i < 1\}; \\ \mathcal{C}_J &= \{C_j \in \mathcal{C} : 0 < \tilde{y}_j < 1\}. \end{aligned}$$

Obviously, $\mathcal{C}_I \cap \mathcal{C}_J = \emptyset$. The main idea in this step is to round the fractional calls in $\mathcal{C}_I$ and $\mathcal{C}_J$ independently. In detail, for each call in $\mathcal{C}_I$ (and $\mathcal{C}_J$) we decide either to transmit it entirely clockwise (and counter-clockwise) or throw it away.

Renumber the calls in $\mathcal{C}_I$ as $C_{i_1}, C_{i_2}, \cdots, C_{i_p}$ ordered clockwise by their sources; accordingly, renumber the calls in $\mathcal{C}_J$ as $C_{j_1}, C_{j_2}, \cdots, C_{j_q}$ ordered counter-clockwise by their sources. Therein, as in the previous subsection, for any $k \in \{0, 1, \cdots, n-1\}$, the indices of the calls in $\mathcal{C}_I$ parallel to the clockwise link $(k, k+1)$ is denoted by an interval $[s_k, s'_k] \subseteq \{i_1, i_2, \cdots, i_p\}$ and the indices of the calls in $\mathcal{C}_J$ parallel to the counter clockwise link $(k+1, k)$ is denoted by an interval $[t_k, t'_k] \subseteq \{j_1, j_2, \cdots, j_q\}$, where the intervals $[s_k, s'_k]$ and $[t_k, t'_k]$ are interpreted under modulo p and q operations, respectively.

Now we use the "rounding technique" again to construct an unsplit routing (x^*, y^*) from $(\tilde{x}, \tilde{y})$ recursively:

1) For $C_{i_l} \in \mathcal{C}_I$ $(l = 1, 2, \cdots, p)$:

$$\begin{aligned} y^*_{i_l} &= 0 \qquad l = 1, 2, \cdots, p \\ x^*_{i_l} &= \begin{cases} 1 \text{ if } -d_{i_l}\tilde{x}_{i_l} + \sum_{h=0}^{l-1} d_{i_h}(x^*_{i_h} - \tilde{x}_{i_h}) < 0 \\ 0 \text{ otherwise;} \end{cases} \end{aligned} \tag{5}$$

2) For $C_{j_l} \in \mathcal{C}_J$ $(l = 1, 2, \cdots, q)$:

$$\begin{aligned} x^*_{j_l} &= 0 \qquad l = 1, 2, \cdots, q \\ y^*_{j_l} &= \begin{cases} 1 \text{ if } -d_{j_l}\tilde{y}_{j_l} + \sum_{h=0}^{l-1} d_{j_h}(y^*_{j_h} - \tilde{y}_{j_h}) < 0 \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{6}$$

3) For each call $C_l \notin \mathcal{C}_I \cup \mathcal{C}_J$, define $x^*_l = \tilde{x}_l$ and $y^*_l = \tilde{y}_l$.

Lemma 2. *For the unsplit routing (x^*, y^*) constructed by (5) and (6), every partial sum satisfies*

$$\sum_{l=1}^{h} d_{i_l}(x^*_{i_l} - \tilde{x}_{i_l}) \in [0, d_{max}), \quad \sum_{l=1}^{h'} d_{j_l}(y^*_{i_l} - \tilde{y}_{i_l}) \in [0, d_{max}),$$

for $h = 1, 2, \cdots, p$ and $h' = 1, 2, \cdots, q$.

Proposition 4. *We can obtain an unsplit routing (x^*, y^*) from a semi-unsplit routing $(\tilde{x}, \tilde{y})$ in polynomial time, such that $T(x^*, y^*) \geq T(\tilde{x}, \tilde{y})$ and the increment of each link load is no more than $2d_{max}$.*

Proof. By Lemma 2, we have

$$\sum_{l=1}^{p} d_{i_l} x^*_{i_l} \geq \sum_{l=1}^{p} d_{i_l}\tilde{x}_{i_l} \quad \text{and} \quad \sum_{l=1}^{q} d_{j_l} y^*_{j_l} \geq \sum_{l=1}^{q} d_{j_l}\tilde{y}_{j_l},$$

which implies that $T(x^*, y^*) \geq T(\tilde{x}, \tilde{y})$. With the similar arguments as in the proof of Proposition 2, it can be shown that for clockwise link $(k, k+1)$ and counter clockwise link $(k+1, k)$,

$$\begin{aligned} L^+_k(x^*, y^*) - L^+_k(\tilde{x}, \tilde{y}) &= \textstyle\sum_{i_l \in [s_k, s'_k]} d_{i_l}(x^*_{i_l} - \tilde{x}_{i_l}) \leq 2d_{max}; \\ L^-_k(x^*, y^*) - L^-_k(\tilde{x}, \tilde{y}) &= \textstyle\sum_{j_l \in [t_k, t'_k]} d_{j_l}(y^*_{j_l} - \tilde{y}_{j_l}) \leq 2d_{max}. \end{aligned}$$

That is, the increment of each link load is no more than $2d_{max}$, as desired. ∎

Proof of Theorem 2. Based on Proposition 1-4, the routing (x^*, y^*) obtained is in fact a feasible unsplit routing of $\mathcal{C}$ with throughput $T(x^*, y^*)$ at least λ^* times the optimum. Let $K > 2$ be a constant, and assume that $d_{max} \leq \frac{1}{K} d^*$, then

$$\lambda^* = \frac{d^* - 2d_{max}}{d^* + \frac{3}{2}d_{max}} \geq 1 - \frac{7}{2K+3}.$$

On the other hand, it it obvious that the algorithm of finding (x^*, y^*) can be carried out in polynomial time. Therefore, the theorem is proved. ∎

We note that when d_{max} is large, the performance ratio of the algorithm is rather bad. However, in a practical data transmission network, the call demand is in general much smaller than the network capacity. Even if there are some calls with large demand, they can usually be divided into several parts to be transmitted independently. Therefore, this algorithm may perform well in practice.

4 Conclusion

We have proved that the problem of CCR-Ring is $\mathcal{NP}$-hard, and presented a polynomial-time approximation algorithm for it. There are two distinct "rounding" phases in the algorithm. The first involves determining the possible direction for each call transmission, and the second involves determining which calls are accepted. In practice when d_{max} is small compared with d^*, the performance ratio of the algorithm is good.

There are some related questions for future research. First, with more techniques applied to the calls with large demand (e.g., [5,10]), it is probably to obtain an approximation algorithm with better performance ratio.

Second, when all the call demands are unit (un-weighted case), the computational complexity of the CCR-Ring problem is not known yet. But we guess it is polynomially solvable. The guess is based on the fact that our algorithm applied to the un-weighted case leads to an approximation algorithm with ratio $1 - \frac{7}{2d^*+3}$, which seems "better than" a PTAS when d^* is considered as a part of the input size.

References

1. Adamy, U., Abbuehl, C., Anand, R.S., Erlebach, T.: Call control in rings. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 788–799. Springer, Heidelberg (2002)
2. Chekuriy, C., Mydlarzz, M., Shepherd, F.B.: Multicommodity demand flow in a tree. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 410–425. Springer, Heidelberg (2003)
3. Cosares, S., Saniee, I.: An optimization problem related to balancing loads on SONET rings. Telecommunications Systems 3, 165–181 (1992)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability: A guide to the Theory of $\mathcal{NP}$-Completeness. W.H. Freeman, San Fransisco (1979)

5. Khanna, S.: A polynomial-time approximation scheme for the SONET ring loading problem. Bell Labs Technical Journal 2(2) (1997)
6. Ramaswami, R., Sivarajan, K.N.: Optical Networks: A Practical Perspective, 2nd edn. Morgan Kaufmann Publishers, San Francisco (2002)
7. Anand, R.S., Erlebach, T.: Routing anc call control algorithms for ring networks. In: Dehne, F., Sack, J.-R., Smid, M. (eds.) WADS 2003. LNCS, vol. 2748, pp. 186–197. Springer, Heidelberg (2003)
8. Schrijver, A., Seymour, P., Winkler, P.: The ring loading problem. SIAM Journal on Discrete Mathematics 11, 1–14 (1998)
9. Wilfong, P., Winkler, P.: Ring routing and wavelength translation. In: SODA. Proceedings of Ninth Annual ACM-SIAM Symposium on Discrete Algorithm, pp. 333–341. ACM Press, New York (1998)
10. Wan, P., Yang, Y.: Load-balanced routing in counter rotated SONET rings. Networks 35, 279–286 (2000)

Fast Matching Method for DNA Sequences*

Jin Wook Kim[1], Eunsang Kim[2], and Kunsoo Park[2]

[1] HM Research,
San 56-1, Sillim-dong, Gwanak-gu, Seoul, 151-742, Korea
jwkim@theory.snu.ac.kr
[2] School of Computer Science and Engineering,
Seoul National University, Seoul, 151-742, Korea
{eskim, kpark}@theory.snu.ac.kr

Abstract. DNA sequences are the fundamental information for each species and a comparison between DNA sequences of different species is an important task. Since DNA sequences are very long and there exist many species, not only fast matching but also efficient storage is an important factor for DNA sequences. Thus, a fast string matching method suitable for encoded DNA sequences is needed. In this paper, we present a fast string matching method for encoded DNA sequences which does not decode DNA sequences while matching. We use four-characters-to-one-byte encoding and combine a suffix approach and a multi-pattern matching approach. Experimental results show that our method is about 5 times faster than AGREP and the fastest among known algorithms.

1 Introduction

In molecular biology, DNA sequences are the fundamental information for each species and a comparison between DNA sequences is an interesting and basic problem. Since a DNA sequence is represented by a sequence of four bases - A, C, T, and G, the comparison problem is the same as a matching problem between strings. There are various kinds of comparison tools and the famous two are BLAST [5] and FASTA [11]. These tools provide approximate matching. In fact, however, they are based on exact matching to speed up.

The exact matching problem is to find all the occurrences of a given pattern P in a large text T, where both T and P are sequences of characters from a finite alphabet Σ. Many algorithms have been developed for exact matching and they are divided into three approaches [20]: Prefix approach, suffix approach, and factor approach. For the prefix approach, there are the Knuth-Morris-Pratt (KMP) algorithm [16], the Shift-Or algorithm [4] and its variants [14]. For the suffix approach, there are the Boyer-Moore (BM) algorithm [6] and its variants – the Horspool algorithm [15], the Sunday algorithm [25], and the hybrid algorithm [12]. For the factor approach, there are the Backward Nondeterministic Dawg Matching (BNDM) algorithm [19] and the Backward Oracle Matching (BOM) algorithm [3]. In addition, there exists some results for small alphabet [7,26]. In [26], Tarhio and Peltola (TP) use q-gram for shifts. The approximate pattern

* This work was supported by FPR05A2-341 of 21C Frontier Functional Proteomics Project from Korean Ministry of Science & Technology.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 271–281, 2007.

matching tool, AGREP [27,28], has the exact match routine which has been widely used. For more information, see [20].

A comparison between DNA sequences of different species is an important task. Since DNA sequences are very long and there exist many species, not only fast matching but also efficient storage is an important factor for DNA sequences. Thus, a fast string matching method suitable for encoded DNA sequences is needed.

In fact, various encoded string matching algorithms have been developed for general string matching after the compressed matching problem was first defined by Amir and Benson [1]. Manber [17] used an encoding method which compresses the phrase of length 2 to one byte. However, the problem of this method is that the pattern may have more than one encoding. Moura et al. [18] proposed an algorithm which consists of the Sunday algorithm with a semi-static word-based modeling and a Huffman coding. It is 2 times faster than AGREP. Amir, Benson, and Farach [2] gave the first matching algorithm for Ziv-Lempel encoded texts. Recently, for Ziv-Lempel encoding method, Navarro and Raffinot [21] used the Shift-Or algorithm, and Navarro and Tarhio [22] used the BM algorithm. For byte pair encoding method, Shibata et al. [23] used the Knuth-Morris-Pratt algorithm and Shibata et al. [24] used the BM algorithm. The algorithm in [23] is 2 times faster than AGREP for GenBank data set with $m \leq 16$ where m is the pattern length and the algorithm in [24] is 3 times faster with $m \leq 30$.

Recently there have been efforts to develop practical and fast string matching algorithms just for DNA sequences [8,13]. The following algorithms use a similar encoding method which is suitable for small alphabet. Fredriksson [13] used an encoded character called a super-alphabet. A super-alphabet of size σ^s is defined as packing s symbols of text T to a single super-symbol where σ is the size of the alphabet. Fredriksson's Shift-Or algorithm with a super-alphabet of size $4^4 = 256$ is 5 times faster than the original Shift-Or algorithm with DNA sequences. Chen, Lu, and Ram [8] used the following encoding method: each DNA base is encoded to two-bit code and thus four bases can be considered at a time. Chen, Lu, and Ram's d-BM algorithm makes four encoded patterns and then does matching using BM for each encoded pattern. It is faster than AGREP with $m > 50$.

In this paper, we present a fast string matching method for encoded DNA sequences. Our method uses encoded texts for matching and does not decode them. We use four-characters-to-one-byte encoding and combine a suffix approach and a multi-pattern matching approach, i.e., a combination of a multi-pattern version of the Sunday algorithm and a simplified version of the Commentz-Walter algorithm [9,10]. Through this combination, we get the most efficient string matching method for encoded DNA sequences. We implement various algorithms and compare with our method. Experimental results show that our method is about 5 times faster then AGREP and the fastest string matching method for encoded DNA sequences among known algorithms.

2 Preliminaries

We first give some definitions and notations that will be used in this paper. Let Σ be an alphabet and σ be the size of Σ. A string is concatenations of zero or more characters from alphabet Σ. The length of a string S is denoted by $|S|$. Let $S[i]$ denote ith character

of a string S for $1 \leq i \leq |S|$ and $S[i..j]$ denote a substring $S[i]S[i+1]\ldots S[j]$ of S for $1 \leq i \leq j \leq |S|$.

Let T be a text string of length n and P be a pattern string of length m. The string matching problem is defined as follows: Given a text T and a pattern P, find all occurrences of P in T.

In this paper, we consider DNA sequences. There are four characters, A, C, T, and G, i.e., $\Sigma = \{\text{A, C, T, G}\}$ and thus $\sigma = 4$. The problem we consider in this paper is as follows.

Problem 1. Let T be a text, P be a pattern and T' be an encoded text of T. Given T' and P, find all occurrences of P in the original text T without decoding T'.

There are various encoding methods: Ziv-Lempel, Huffman, BPE, etc. If an encoding method E such that $E(a) = a$ for $a \in \Sigma$ is used, then $T' = T$ and Problem 1 is the same as the string matching problem.

3 Proposed Method

We want to solve Problem 1 for DNA sequences as fast as possible. To do this, we will propose an algorithm that uses the following methods:

- Encoding method : A fixed-length encoding method.
- Matching method : A suffix approach and a multi-pattern matching approach.

We first explain the encoding method of our algorithm, and then explain the matching method of our algorithm.

3.1 Encoding

We use a fixed-length encoding method. Since $\Sigma = \{\text{A, C, T, G}\}$, i.e., $\sigma = 4$, we need only two bits for each character. Thus we can define a mapping M:

$$M(\text{A}) = 00, M(\text{C}) = 01, M(\text{T}) = 10, M(\text{G}) = 11.$$

Since there are eight bits in one byte, we can encode four characters to one byte. Given a substring $S[i..i+3]$, we define an encoding method E such that

$$E(S[i..i+3]) = M(S[i]) \parallel M(S[i+1]) \parallel M(S[i+2]) \parallel M(S[i+3])$$

where $\parallel$ is the concatenation operator. Then, the size σ' of the encoded alphabet is $4^4 = 256$.

We explain text encoding and pattern encoding.

Text Encoding. Given a text T, an encoded text T' is defined as $[T_m, T_b, Mask_T]$ where T_m is an encoded byte sequence of length $\lceil n/4 \rceil - 1$, T_b is the last encoded byte and $Mask_T$ is a bit mask for T_b.

The encoded byte sequence T_m is defined as

$$T_m[i] = E(T[4i-3..4i])$$

for $1 \le i \le n'$ and $n' = \lceil n/4 \rceil - 1$. The last encoded byte T_b is defined as

$$T_b = M(T[4n'+1]) \parallel \cdots \parallel M(T[4n'+r]) \parallel \overbrace{00 \parallel .. \parallel 00}^{4-r}$$

where $r = n - 4n'$, $1 \le r \le 4$. The bit mask $Mask_T$ is defined as

$$Mask_T = \overbrace{11 \parallel .. \parallel 11}^{r} \parallel \overbrace{00 \parallel .. \parallel 00}^{4-r}.$$

Pattern Encoding. Given a pattern P, we make four encoded patterns. Since we will match an encoded text and an encoded pattern, using only one encoded pattern we cannot find all occurrences but only some positions such that their positions modulo 4 are all the same. Thus we need four encoded patterns such that the possible occurrence positions of them modulo 4 are 0, 1, 2 and 3, respectively.

An encoded pattern P^i for $0 \le i \le 3$ is defined as $[P^i_f, P^i_m, P^i_l, Mask^i_f, Mask^i_l]$ where P^i_m is an encoded byte sequence of length $\lceil (n+i)/4 \rceil - 2$, P^i_f and P^i_l are the first and last encoded bytes and $Mask^i_f$ and $Mask^i_l$ are bit masks for P^i_f and P^i_l, respectively.

The encoded byte sequence P^i_m is defined as

$$P^i_m[j] = E(P[4j+1-i..4j+4-i])$$

for $1 \le j \le m_i$ and $m_i = \lceil (n+i)/4 \rceil - 2$. The first encoded byte P^i_f is defined as

$$P^i_f = \overbrace{00 \parallel .. \parallel 00}^{i} \parallel M(P[1]) \parallel \cdots \parallel M(P[4-i])$$

where $0 \le i \le 3$. The bit mask $Mask^i_f$ is defined as

$$Mask^i_f = \overbrace{00 \parallel .. \parallel 00}^{i} \parallel \overbrace{11 \parallel .. \parallel 11}^{4-i}.$$

And the last encoded byte P^i_l is defined as

$$P^i_l = M(P[4(m_i+1)-i+1]) \parallel \cdots \parallel M(P[4(m_i+1)-i+r]) \parallel \overbrace{00 \parallel .. \parallel 00}^{4-r}$$

where $r = m - 4(m_i+1) - i$, $1 \le r \le 4$. The bit mask $Mask^i_l$ is defined as

$$Mask^i_l = \overbrace{11 \parallel .. \parallel 11}^{r} \parallel \overbrace{00 \parallel .. \parallel 00}^{4-r}.$$

Note that P^i_f and P^i_l are non-empty bytes.

Table 1. Four encoded patterns for P =ATCAACGAGAGATC

i	P_f^i	P_m^i	P_l^i	$Mask_f^i$	$Mask_l^i$
0	00100100	00011100 11001100	10010000	11111111	11110000
1	00001001	00000111 00110011	00100100	00111111	11111100
2	00000010	01000001 11001100	11001001	00001111	11111111
3	00000000	10010000 01110011 00110010	01000000	00000011	11000000

For example, given a pattern P =ATCAACGAGAGATC, four encoded patterns are shown in Table 1.

3.2 Matching

We combine a suffix approach and a multi-pattern matching approach, i.e., a combination of a multi-pattern version of the Sunday algorithm and a simplified version of the Commentz-Walter algorithm. After the encoding stage, we have one encoded text T' and four encoded patterns P^i for $0 \leq i \leq 3$. Since we do not decode T' while matching, we must use four encoded patterns. Thus, to get the most efficient performance for matching T' with four encoded patterns P^i, we adopt a multi-pattern matching approach. For each encoded pattern, we adopt a Boyer-Moore approach which shows the best results among known string matching algorithms.

The matching stage consists of two phases: preprocessing phase and searching phase. We first explain the preprocessing phase, and then explain the searching phase.

Preprocessing. In the preprocessing phase, we make a shift table Δ for encoded patterns. The role of the shift table Δ is to find the nearest candidate position for the next occurrence of any one of four encoded patterns at any position.

We make a shift table Δ via two steps. First, we compute a shift candidate table d^i for each encoded pattern P^i. The shift candidate table d^i is in fact a shift table for one pattern matching. We use the method that there must be at least one byte shift at any time [25]. For each P^i, we compute d^i using P_m^i such that

$$d^i[\alpha] = \min\{m_i + 1, \min\{m_i + 1 - k | P_m^i[k] = \alpha, 1 \leq k \leq m_i\}\}$$

where α is an encoded character, $0 \leq \alpha \leq 255$.

Then, we compute a shift table Δ from d^i. Since each d^i means the minimum offset for the next candidate position of P^i, we choose the minimum of d^i to find the next candidate position of four encoded patterns, i.e.,

$$\Delta[\alpha] = \min\{d^i[\alpha] | 0 \leq i \leq 3\}.$$

In addition, we make an index list for each encoded character α. When α appears in the last position of P^i, i.e., $P_m^i[m_i] = \alpha$, i is inserted in the index list for α. Using this, we can search for only the encoded patterns that P_m ends with α.

Searching. In the searching phase, we find all occurrences of four encoded patterns in an encoded text. The searching phase is divided into two parts: match part and shift part. Figure 1 shows a pseudocode for the searching phase.

```
while i ≤ n'
    for r in the index list of T_m[i]            // match part
        k ← i − 1, j ← m_r − 1
        while j > 0 and T_m[k] = P_m^r[j]
            k ← k − 1, j ← j − 1
        end while
        if j = 0
            if i < n'
                if P_f^r = T_m[k]&Mask_f^r and P_l^r = T_m[i + 1]&Mask_l^r
                    pattern occurs at position 4 ∗ k − (3 − r)
                end if
            end if
            else if P_f^r = T_m[k]&Mask_f^r and P_l^r = T_b&Mask_l^r
                pattern occurs at position 4 ∗ k − (3 − r)
            end if
        end if
    end for
    do                                           // shift part
        i ← i + Δ[T_m[i + 1]]
    while i ≤ n' and Δ[T_m[i]] ≠ 1
end while
```

Fig. 1. Pseudocode for searching phase

In the match part, we match each character of encoded patterns and the encoded text. Let i be a pointer to the encoded text T_m. When the index list for the character of $T_m[i]$ contains some entries, we try to match T_m and encoded patterns which are indicated by the index list. Since the first and last characters of the encoded patterns have their bit masks, a text character is masked by using the bit mask of the pattern. If we find a match, output the match position at the original text. After that, we start the shift part.

In the shift part, we move a pointer i from left to right via a shift table Δ as far as possible. At first, i is shifted by $\Delta[i+1]$ [25]. Then, if the encoded text character at the shifted position i has no entry in its index list, shift again. This guarantees that at least one shift occurs at each iteration. Lemma 1 shows the correctness of our algorithm.

Lemma 1. *Using a shift table Δ, we can find every candidate position for four encoded patterns.*

Proof. Let i be the current position of the encoded text T_m and let α be the encoded character at the position $i+1$ of T_m. There are four values $d^i[\alpha]$ for $0 \leq i \leq 3$ and, W.L.O.G., let $\Delta[\alpha] = d^0[\alpha]$. Suppose that an encoded pattern P_m^1 is matched at $i + d^1[\alpha] - m_1 + 1$ of T_m and $\Delta[\alpha] < d^1[\alpha]$. Then to prove this lemma, it is sufficient to show that the current position comes to $i + d^1[\alpha]$. Since the current position is now updated to $i+\Delta[\alpha]$, the next shift added to $i+\Delta[\alpha]$ is $\Delta[\beta]$, where $\beta = T_m[i+\Delta[\alpha]+1]$. Because P_m^1 occurs at $i + d^1[\alpha] - m_1 + 1$, we get $\beta = P_m^1[m_1 - (d^1[\alpha] - \Delta[\alpha]) + 1]$.

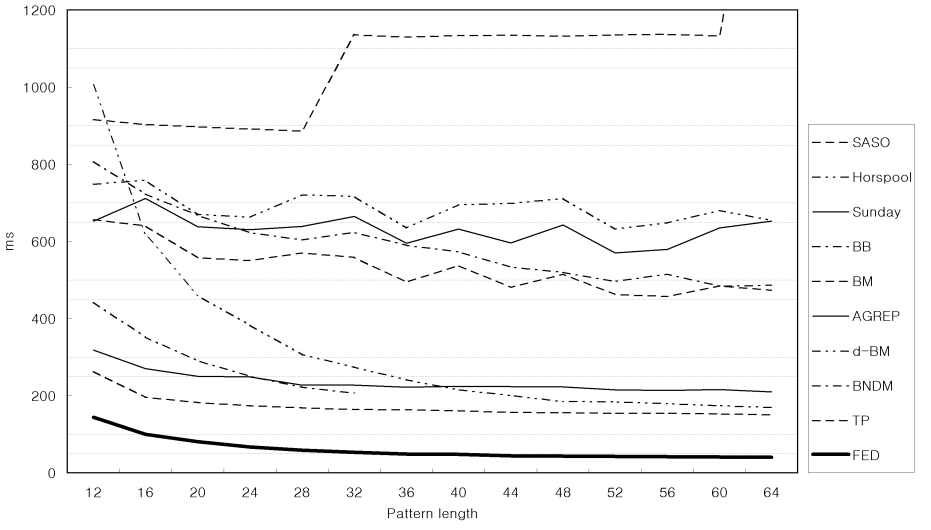


Fig. 2. Running time for *Mus Musculus* fragment set with the pattern length between 12 and 64

Thus, $d^1[\beta] \leq d^1[\alpha] - \Delta[\alpha]$ and $\Delta[\beta] \leq d^1[\beta]$. Therefore $\Delta[\alpha] + \Delta[\beta] \leq d^1[\alpha]$ and $i + \Delta[\alpha] + \Delta[\beta] \leq i + d^1[\alpha]$. Since Δ is larger than 1, after repeating the above step, the current position comes to $i + d^1[\alpha]$, eventually.

3.3 Analysis

The worst case time complexity is $O(n'km')$, where n' is the length of the encoded text T_m, m' is the maximum of the lengths of four encoded pattern P_m^i and k is 4, and the best case time complexity is $O(n'/m' + m'occ)$, where occ is the number of all occurrences of the pattern in the text.

4 Experimental Results

We had experiments with our algorithm FED (fast matching with encoded DNA sequences) and the following algorithms: BM (Boyer-Moore) [6], Horspool [15], Sunday [25], AGREP [27,28], TP (Tarhio and Peltola) [26], BNDM (backward nondeterministic dawg matching) [19], BB (BM on byte pair encoding) [24], SASO (super-alphabet shift-or) [13], and d-BM [8]. BB is the compressed pattern matching algorithm on byte-pair encoding and SASO, d-BM and FED are the compressed pattern matching algorithms on fix-length encoding. Others are the original pattern matching algorithms.

We implemented all algorithms by ourselves, except AGREP and BNDM. All the algorithms had been implemented in C, compiled with gcc 3.2. We ran the experiments in 2.4GHz Xeon with 2 GB RAM, running GNU/Linux 2.4.20-28.

We had experiments on ten real DNA sequence data sets from NCBI: The sizes of data sets are varied from 7.6MB to 220MB. For each data set, the patterns were

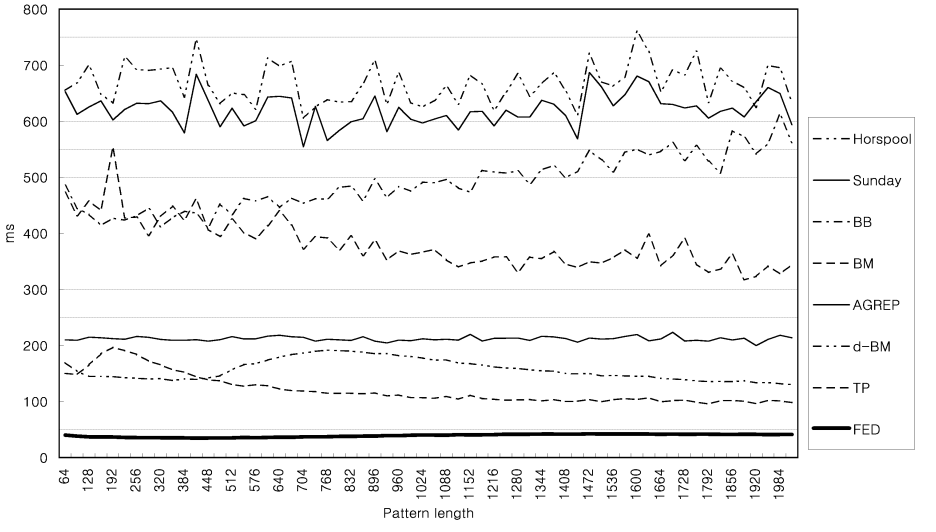


Fig. 3. Running time for *Mus Musculus* fragment set with the pattern length between 64 and 2016

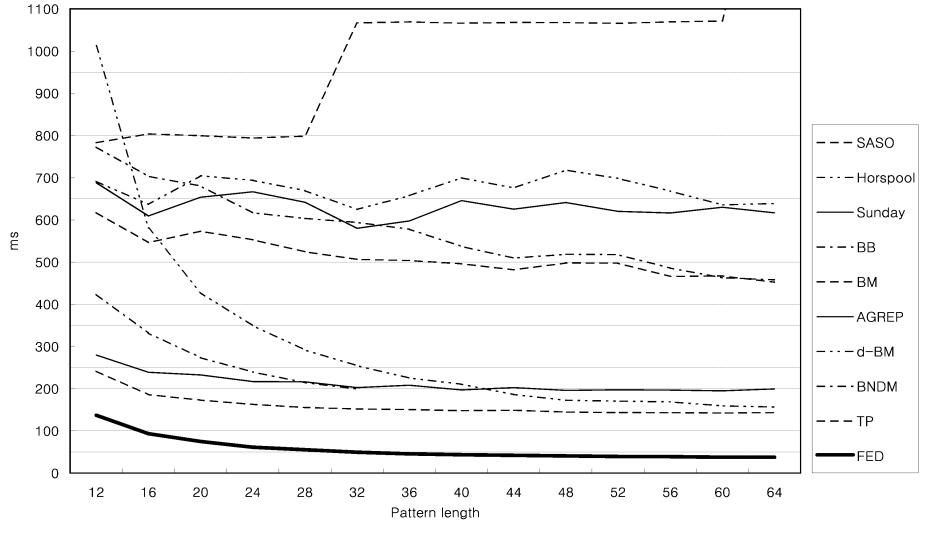


Fig. 4. Running time for *Homo sapiens* chromosome 1 with the pattern length between 12 and 64

randomly extracted from the texts, and each test was repeated 50 times. Since the experimental results are similar, we show typical two examples: *Mus Musculus* fragments set (220MB) and *Homo sapiens* chromosome 1 long sequence (210MB). We report the average time in milliseconds which includes the pattern encoding, preprocessing and searching times. Figures 2–5 shows the experimental results.

Fig. 5. Running time for *Homo sapiens* chromosome 1 with the pattern length between 64 and 2016

The proposed algorithm FED is faster than all the others from short patterns to long patterns. Figures 2 and 4 show the running times for short patterns whose lengths are from 12 to 64. FED is $2 \sim 5$ times faster than AGREP and $2 \sim 3.5$ times faster than TP. In addition, FED is at least 3 times faster than BNDM. A recent algorithm FAOSO (fast average optimal shift or) in [14] is reported about 2 times faster than BNDM on DNA sequences, and thus FED is still faster than FAOSO. Figures 3 and 5 show the running times for long patterns whose lengths are from 64 to 2016. FED is 5 times faster than AGREP and $2.5 \sim 5$ times faster than TP.

5 Conclusions

We have presented a string matching algorithm suitable for encoded DNA sequences and shown that our algorithm is the fastest among known algorithms. In addition, since the matching process is done with the encoded text as it is, we can save the time and space overhead for decoding.

References

1. Amir, A., Benson, G.: Efficient Two-Dimensional Compressed Matching. Data Compression Conference, 279–288 (1992)
2. Amir, A., Benson, G., Farach, M.: Let Sleeping Files Lie: Pattern Matching in Z-compressed Files. In: 5th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 705–714 (1994)

3. Allauzen, C., Crochemore, M., Raffinot, M.: Efficient experimental string matching by weak factor recognition. In: Amir, A., Landau, G.M. (eds.) CPM 2001. LNCS, vol. 2089, pp. 51–72. Springer, Heidelberg (2001)
4. Baeza-Yates, R., Gonnet, G.H.: A New Approach to Text Searching. Communications of the ACM 35(10), 74–82 (1992)
5. BLAST, http://www.ncbi.nlm.nih.gov/BLAST
6. Boyer, R.S., Strother Moore, J.: A Fast String Searching Algorithm. Communications of the ACM 20(10), 762–772 (1977)
7. Charras, C., Lecroq, T., Daniel Pehoushek, J.: A Very Fast String Matching Algorithm for Small Alphabets and Long Patterns. In: Farach-Colton, M. (ed.) CPM 1998. LNCS, vol. 1448, pp. 55–64. Springer, Heidelberg (1998)
8. Chen, L., Lu, S., Ram, J.: Compressed Pattern Matching in DNA Sequences. In: CSB 2004. IEEE Computational Systems Bioinformatics Conference, pp. 62–68 (2004)
9. Commentz-Walter, B.: A String Matching Algorithm Fast on the Average. In: Maurer, H.A. (ed.) Automata, Languages, and Programming. LNCS, vol. 71, pp. 118–132. Springer, Heidelberg (1979)
10. Commentz-Walter, B.: A String Matching Algorithm Fast on the Average. Technical Report TR 79.09.007, IBM Germany, Heidelberg Scientific Center (1979)
11. FASTA, http://www.ebi.ac.uk/fasta
12. Franek, F., Jennings, C.G., Smyth, W.F.: A Simple Fast Hybrid Pattern-Matching Algorithm. In: Apostolico, A., Crochemore, M., Park, K. (eds.) CPM 2005. LNCS, vol. 3537, pp. 288–297. Springer, Heidelberg (2005)
13. Fredriksson, K.: Shift-Or String Matching with Super-Alphabets. Information Processing Letters 87(4), 201–204 (2003)
14. Fredriksson, K., Grabowski, S.: Practical and Optimal String Matching. In: Consens, M.P., Navarro, G. (eds.) SPIRE 2005. LNCS, vol. 3772, pp. 376–387. Springer, Heidelberg (2005)
15. Nigel Horspool, R.: Practical Fast Searching in Strings. Software Practice and Experience 10(6), 501–506 (1980)
16. Knuth, D.E., Morris Jr, J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM Journal on Computing 6, 323–350 (1977)
17. Manber, U.: A Text Compression Scheme That Allows Fast Searching Directly in the Compressed File. ACM Transactions on Information Systems 15(2), 124–136 (1997)
18. de Moura, E.S., Navarro, G., Ziviani, N., Baeza-Yates, R.: Direct Pattern Matching on Compressed Text. In: 5th International Symposium on String Processing and Information Retrieval, pp. 90–95. IEEE Computer Society Press, Los Alamitos (1998)
19. Navarro, G., Raffinot, M.: Fast and Flexible String Matching by Combining Bit-Parallelism and Suffix Automata. ACM Journal of Experimental Algorithmics 5(4) (2000)
20. Navarro, G., Raffinot, M.: Flexible Pattern Matching in Strings: Practical On-Line Search Algorithms for Texts and Biological Sequences. Cambridge University Press, Cambridge (2002)
21. Navarro, G., Raffinot, M.: Practical and Flexible Pattern Matching over Ziv-Lempel Compressed Text. Journal of Discrete Algorithms 2(3), 347–371 (2004)
22. Navarro, G., Tarhio, J.: LZgrep: a Boyer-Moore String Matching Tool for Ziv-Lempel Compressed Text. Software-Practice and Experience 35(12), 1107–1130 (2005)
23. Shibata, Y., Kida, T., Fukamachi, S., Takeda, M., Shinohara, A., Shinohara, T., Arikawa, S.: Speeding Up Pattern Matching by Text Compression. In: Bongiovanni, G., Petreschi, R., Gambosi, G. (eds.) CIAC 2000. LNCS, vol. 1767, pp. 306–315. Springer, Heidelberg (2000)
24. Shibata, Y., Matsumoto, T., Takeda, M., Shinohara, A., Arikawa, S.: A Boyer-Moore Type Algorithm for Compressed Pattern Matching. In: Giancarlo, R., Sankoff, D. (eds.) CPM 2000. LNCS, vol. 1848, pp. 181–194. Springer, Heidelberg (2000)

25. Sunday, D.M.: A Very Fast Substring Search Algorithm. Communications of the ACM 33(8), 132–142 (1990)
26. Tarhio, J., Peltola, H.: String Matching in the DNA Alphabet. Software-Practice and Experience 27(7), 851–861 (1997)
27. Wu, S., Manber, U.: Fast Text Searching Allowing Errors. Communications of the ACM 35(10), 83–91 (1992)
28. Wu, S., Manber, U.: AGREP - A Fast Approximate Pattern-matching Tool. In: The Winter 1992 USENIX Conference, pp. 153–162 (1992)

All-Pairs Ancestor Problems in Weighted Dags

Matthias Baumgart, Stefan Eckhardt, Jan Griebsch, Sven Kosub, and Johannes Nowak

Fakultät für Informatik, Technische Universität München,
Boltzmannstraße 3, D-85748 Garching bei München, Germany
{baumgart,eckhardt,griebsch,kosub,nowakj}@in.tum.de

Abstract. This work studies (lowest) common ancestor problems in (weighted) directed acyclic graphs. We improve previous algorithms for the all-pairs representative LCA problem to $O(n^{2.575})$ by using fast rectangular matrix multiplication. We prove a first non-trivial upper bound of $O(\min\{n^2 m, n^{3.575}\})$ for the all-pairs all lowest common ancestors problem. Furthermore, classes of dags are identified for which the problem can be solved considerably faster. Our algorithms scale with the maximal number of LCAs for one pair and—based on the famous Dilworth's theorem—with the size of a maximum antichain (i.e., width) of the dag. We extend and generalize previous results on computing shortest ancestral distances. It is shown that finding shortest distance common ancestors in weighted dags is not harder than computing all-pairs shortest distances, up to a polylogarithmic factor. Finally, we present a solution for the general all-pairs shortest distance LCA problem based on computing all-pairs all LCAs.

1 Introduction

Directed acyclic graphs (dags) are powerful tools for modelling causality systems or other kinds of entity dependencies. If we think of causal relations among a set of events, natural questions come up, such as: Which events are entailed by two given events? What is the first event which is entailed by two given events? In dags, these questions can be answered by computing common ancestors (CAs), i.e., vertices that are reachable via a path from each of the given vertices, and computing lowest common ancestors (LCAs), i.e., those common ancestors that are not reachable from any other common ancestor of the two given vertices.

Although LCA algorithms for general dags are indispensable computational primitives, they have been found an independent subject of studies only recently [6, 7, 15]. There is a lot of sophisticated work devoted to LCA computations for the special case of trees (see, e.g., [14, 19, 7]), but due to the limited expressive power of trees they are often applicable only in restrictive or over-simplified settings. In [7], a list of examples can be found where LCA queries on dags are necessary. We add two more applications.

A first one concerns *phylogenetic networks.* LCA algorithms have been frequently used in the context of phylogenetic trees, i.e., trees that depict the ancestor relations of species, genes, or features. Bacteria obtain a large portion of their genetic diversity through the acquisition of distantly related organisms, via horizontal gene transfer (HGT) or recombination. While views as to the extent of HGT and cross species recombination in bacteria differ, it is widely accepted that they are among the main processes

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 282–293, 2007.

driving prokaryotic evolution and are (with random mutations) mainly responsible for the development of antibiotic resistance. Such evolutionary developments cannot be modeled by trees, and thus there has been an increasing interest in phylogenetic dag networks and appropriate analysis methods [16,17]. Unfortunately, many of the established approaches from phylogenetic trees cannot trivially be extended to dags. This is particularly true for the computation of ancestor relationships.

A second application is related to *Internet protocols.* Currently, inter-domain routing is mainly done using the Border Gateway Protocol (BGP). BGP allows participating autonomous systems to announce and withdraw routable paths over physical connections with their neighbors. This process is governed by local routing policies which are rationally based on commercial relationships between autonomous systems. It has been recognized that, even if globally well-configurated, these local policies have a critical influence on routing stability and quality. An orientation of the underlying connectivity graph imposed by customer-to-provider relations can be viewed as a dag. Routes through the Internet have the typical structure of uphill and downhill parts to the left and right of a top provider in the middle of the route (see, e.g., [13]). Computing such top providers, which are just CAs and LCAs, is needed for reliability or efficiency analyses. In some experimental setting, we have a small Internet sample which constitutes a dag with 11,256 vertices and 13,655 edges. Finding top providers for each of the 63,343,140 pairs makes fast CA and LCA algorithms an issue.

The contribution of this work is twofold. On the one hand, we study the problem of finding all LCAs for each vertex pair. In spite of its importance in both the applications given in [7] and above, this has—to the best of our knowledge—not been studied so far. On the other hand, we improve, extend, and generalize the study of efficient all-pairs LCA computations in dags. In particular, we consider the following problems:

ALL-PAIRS REPRESENTATIVE LCA: Compute one (representative) LCA for each pair of vertices.

ALL-PAIRS ALL LCA: Compute the set of all LCAs for each pair of vertices.

ALL-PAIRS SHORTEST DISTANCE (L)CA: Compute the (L)CA that minimizes the ancestral distance for each pair of vertices. We note that we also compute the corresponding minimal ancestral distances as a byproduct.

Related work. LCA algorithms have been extensively studied in the context of trees with most of the research rooted in [1,21]. The first optimal algorithm for the all-pairs LCA problem in trees, with linear preprocessing time and constant query time, was given in [14]. The same asymptotic was reached using a simpler and parallelizable algorithm in [19]. More algorithmic variants can be found in, e.g., [8,23,22,9].

In the more general case of dags, a pair of nodes may have more than one LCA, which leads to the distinction of *representative* versus *all* LCA solutions. In early research both versions still coincide by considering dags with each pair having at most one LCA. Extending the work on LCAs in trees, in [18], an algorithm was described with linear preprocessing and constant query time for the LCA problem on arbitrarily directed trees (or, causal polytrees). Another solution was given in [2], where the

representative problem in the context of object inheritance lattices was studied. The approach in [2], which is based on poset embeddings into boolean lattices yielded $O(n^3)$ preprocessing and $\log n$ query time on lower semilattices.

The representative LCA problem on general dags has been recently studied in [6, 7, 15]. These works rely on fast matrix multiplications to achieve $\tilde{O}(n^{\frac{\omega+3}{2}})$ [7] and $O(n^{2+\frac{1}{4-\omega}})$ [15] preprocessing time on dags with n nodes and m edges. Currently, the fastest known algorithm for matrix multiplication needs $O(n^\omega)$, with $\omega < 2.376$ [10] implying $\frac{\omega+3}{2} < 2.688$ and $2 + \frac{1}{4-\omega} < 2.616$. For sparse dags, an $O(nm)$ algorithm has been presented in [15] as well.

Bender et al. [6] have shown that shortest ancestral (CA) distances in unweighted dags can be computed in time $O(n^{2.575})$. In the same work, the authors consider a restricted version of the problem of computing shortest LCA distances. Here, additional information from genealogical data is used to rank candidate LCAs in pedigree graphs. However, the general version seems to be much more difficult.

Results. We summarize the technical contributions of this paper. All proofs are omitted due to space limitations and can be found in [4].

ALL-PAIRS REPRESENTATIVE LCA: We improve previous approaches in [7, 15] to $O(n^{2+\mu})$ by straightforward application of fast rectangular matrix multiplication.[1] Throughout this work, $\omega(x, y, z)$ is the exponent of the algebraic matrix multiplication of an $n^x \times n^y$ with an $n^y \times n^z$ matrix. Let μ satisfy $\omega(1, \mu, 1) = 1 + 2\mu$. The fastest known algorithms for rectangular matrix multiplication imply $\mu < 0.575$ [11].

ALL-PAIRS ALL LCA: We give the first non-trivial upper bounds for this problem. In particular, we show that this problem can be solved in time $O(\min\{n^2 m, n^{3+\mu}\})$ on general dags. We further identify classes of dags for which ALL-PAIRS ALL LCA can be solved considerably faster. Namely, for dags of bounded width $\mathrm{w}(G)$, our construction exhibits nice scaling properties. The approach is based on the well known Dilworth's theorem. In particular, the above bounds improve to $O(\mathrm{w}(G) \min\{nm, n^{2+\mu}\})$. Additionally, we give an algorithm that scales with the size k of the largest LCA set, achieving an $O(k^2 \min\{nm, n^{2+\mu}\})$ upper bound. For sparse dags of small width $o(\frac{n}{\sqrt{m}})$ we thereby settle an open question posed by Ukkonen and Nykänen [18].

ALL-PAIRS SHORTEST DISTANCE (L)CA: We extend previous results for unweighted dags to weighted dags. We give an easy dynamic programming algorithm for the ALL-PAIRS SHORTEST DISTANCE CA problem which is optimal on sparse dags. We prove that both computing shortest ancestral distances and shortest distance common ancestors on weighted dags is not harder than computing all-pairs shortest distances (APSD), up to a polylogarithmic factor. To obtain this result, we slightly modify the construction of [7] and adapt techniques to identify witnesses for shortest paths (see [20, 24]). Finally, we show how ALL-PAIRS SHORTEST DISTANCE CA can be solved by reducing it to ALL-PAIRS ALL LCA, thereby achieving first non-trivial upper bounds. Intriguingly, it is an open question whether this can be done better.

[1] Most recently, we have learnt that a similar improvement has been independently described in a technical report [12].

2 Preliminaries

Let $G = (V, E)$ be a directed acyclic graph (with edge weights). Throughout this work we denote by n the number of vertices and by m the number of edges. Let $\mathrm{TC}(G)$ denote the transitive closure of G, i.e., the graph having an edge (u, v) if v is reachable from u over some directed path in G. The length of the shortest path from u to v is denoted by $d(u, v)$. A dag $G = (V, E)$ imposes a partial ordering on the vertex set. Let N be a bijection from V onto $\{1, \ldots, n\}$. N is said to be a *topological ordering* if $N(u) < N(v)$ whenever v is reachable from u in G. Such an ordering is consistent with the partial ordering of the vertex set imposed by the dag. A value $N(v)$ is said to be the *topological number* of v with respect to N. Observe that a graph G is a dag if and only if it allows some topological ordering (folklore). Moreover, a topological ordering can be found in time $O(n + m)$. We will refer to a vertex z which has the maximal topological number $N(z)$ among all vertices in a set as the *rightmost* vertex.

Let $G = (V, E)$ be a dag and $x, y, z \in V$. The vertex z is a *common ancestor* (CA) of x and y if both x and y are reachable from z, i.e., (z, x) and (z, y) are in the transitive closure of G. By $\mathrm{CA}(x, y)$, we denote the set of all CAs of x and y. A vertex z is a *lowest common ancestor* (LCA) of x and y if and only if $z \in \mathrm{CA}(x, y)$ and for each $z' \in V$ with $(z, z') \in \mathrm{TC}(G)$ we have $z' \notin \mathrm{CA}(x, y)$. $\mathrm{LCA}(x, y)$ denotes the set of all LCAs of x and y. For any $z \in V$ the ancestral distance of (x, y) with respect to z is $d(z, x) + d(z, y)$.

3 The All-Pairs Representative LCA Problem

In this section we briefly revisit the ALL-PAIRS REPRESENTATIVE LCA problem. All algorithms for this problem we are aware of exploit the following.

Proposition 1. [7,15] *Let $G = (V, E)$ be a dag and let N be a topological ordering. Furthermore, let $x, y \in V$ be vertices with a non-empty set of CAs. If $z \in V$ is the rightmost vertex in $\mathrm{CA}(x, y)$, then z is an LCA of x and y.*

Algorithm 1 solves ALL-PAIRS REPRESENTATIVE LCA in time $O(nm)$. We modify this algorithm later to obtain dynamic programming solutions for our other problems. The correctness of this algorithm follows readily from the following observation.

Observation 2. *Let $G = (V, E)$ be a dag and let $x, y \in V$ be any pair of vertices. Furthermore, let z be the rightmost LCA of x and y.*

1. *If $(x, y) \in \mathrm{TC}(G)$ then $z = x$.*
2. *If $(x, y) \notin \mathrm{TC}(G)$ then the following holds: let $x_1, \ldots, x_k$ be the parents of x. Let $z_1, \ldots, z_k$ be the rightmost LCAs of the pairs $(x_1, y), \ldots, (x_k, y)$. Then z is the rightmost vertex in $\{z_1, \ldots, z_k\}$.*

Theorem 3. *Algorithm 1 solves* ALL-PAIRS REPRESENTATIVE LCA *in time $O(nm)$.*

Kowaluk et al. [15] observed that rightmost LCAs are found by computing maximum witnesses for boolean matrix multiplication. Let A, B, and C be boolean $n \times n$ matrices

Algorithm 1. ALL-PAIRS REPRESENTATIVE LCA

Input: A dag $G = (V, E)$
Output: An array R of size $n \times n$ where $R[x, y]$ is an LCA of x and y

```
1 begin
2     Initialize R[x, y] ← NIL
3     Compute the transitive closure TC(G) and a topological ordering N of G
4     foreach v ∈ V in ascending order of N(v) do
5         foreach (v, x) ∈ E do
6             foreach y ∈ V with N(y) ≥ N(v) do
7                 if (x, y) ∈ TC(G) then R[x, y] ← x
8                 else if N(R[v, y]) > N(R[x, y]) then R[x, y] ← R[v, y]
9 end
```

such that $C = A \cdot B$. We have $C[i, j] = 1$ if and only if there exists $1 \leq k \leq n$ such that $A[i, k] = 1 = B[k, j]$; in this case, we call k a witness for $C[i, j]$. We call k a maximum witness for $C[i, j]$ if k is the maximum one among all $C[i, j]$-witnesses.

Proposition 4. [15] *Let $G = (V, E)$ be a dag and let N be a topological ordering. Let A be the adjacency matrix of the transitive closure $\mathrm{TC}(G)$ such that the vertices are ordered corresponding to N. Then z is a rightmost ancestor of (x, y) if and only if z is a maximum witness for $C[x, y]$, where $C = A^T \cdot A$.*

We briefly describe how to improve the approach taken in [15] by using fast rectangular matrix multiplication [11] for computing the maximum witnesses. In the following, we assume that we have already computed the adjacency matrix A of the transitive closure $\mathrm{TC}(G)$ of G in time $O(\min\{nm, n^\omega\})$. Also, we assume implicitly that the rows and columns in A are ordered according to a topological order of G's vertex set.

Let $\mu \in [0; 1]$ be a parameter. We divide V into equal-sized sets $V_1, \ldots, V_r$ of consecutive vertices (with respect to the topological ordering), where $r = \lceil n^{1-\mu} \rceil$. Thus, the size of the sets is $O(n^\mu)$. Maximum witnesses are found in two steps:

1. For each pair (x, y), determine l such that the maximum witness of (x, y) is in V_l.
2. For each (x, y) and l, search V_l to find the maximum witness.

The implementation of these two steps is straightforward: for a vertex set V_l, let $A^T_{*V_l}$ denote the matrix A^T restricted to the columns corresponding to vertices in V_l. Let $M^{(l)} = A^T_{*V_l} \cdot (A^T_{*V_l})^T$.

Observation 5. *A pair (x, y) of vertices has a common ancestor $z \in V_l$ if and only if $M^{(l)}[x, y] = 1$.*

Hence for $l \in \{1, \ldots r\}$, we compute the $O(n^{1-\mu})$ (rectangular) matrix products $M^{(l)}$ and choose for each pair the maximum index l such that $M^{(l)}[x, y] = 1$. This takes time $O(n^{1-\mu+\omega(1,\mu,1)})$. Recall that $\omega(1, \mu, 1)$ is the exponent of the algebraic matrix multiplication of an $n \times n^\mu$ with an $n^\mu \times n$ matrix. For the second step, we simply search for each pair (x, y) and the corresponding index l of the set V_l manually, that is, for each $z \in V_l$ in descending order until we find z with both (z, x) and (z, y) are in $\mathrm{TC}(G)$. This takes time $O(n^2 \cdot |V_l|) = O(n^{2+\mu})$. For μ satisfying $1 - \mu + \omega(1, \mu, 1) = 2 + \mu$

we get the optimal complexity. Currently, the best known upper bounds for rectangular matrix multiplication are [11]:

$$\omega(1,r,1) = \begin{cases} 2+o(1), & 0 \le r \le 0.294 = \alpha \\ \frac{2(1-r)+(r-\alpha)\omega}{1-\alpha}, & 0.294 < r \le 1. \end{cases}$$

Together, this implies $\mu < 0.575$.

Theorem 6. ALL-PAIRS REPRESENTATIVE LCA *can be solved in time* $O(n^{2+\mu})$, *where* μ *satisfies* $1 + 2\mu = \omega(1, \mu, 1)$.

4 The All-Pairs All LCA Problem

A trivial lower bound for the ALL-PAIRS ALL LCA problem is $\Omega(n^3)$, even for sparse dags with $m = O(n)$ [4]. The definition of LCAs immediately yields several $O(n^2 m)$ algorithms. A rather trivial one, first, computes the transitive closure of G in $O(nm)$. Then, for every vertex z and every pair (x, y), determine in time $O(\text{out-deg}(z))$ if z is an LCA of (x, y). Amazingly, this trivial algorithm is optimal on sparse dags with $m = O(n)$.

We proceed by giving an $O(n^2 m)$ dynamic programming approach which is more suitable for later use. This algorithm adopts ideas from Algorithm 1. Recall that $z \in \mathrm{CA}(x, y)$ is an LCA of x and y if there is no other vertex $z' \in \mathrm{CA}(x, y)$ such that $(z, z') \in \mathrm{TC}(G)$. The following observation generalizes Observation 2.

Observation 7. *Let* $G = (V, E)$ *be a dag. Let* x *and* y *be vertices of* G. *Let* $x_1, \ldots, x_k$ *be the parents of* x *and* S *the union of the sets* $\mathrm{LCA}(x_\ell, y)$ *for all* $1 \le \ell \le k$.

1. *If* $(x, y) \in \mathrm{TC}(G)$ *then* $\mathrm{LCA}(x, y) = \{x\}$.
2. *If* $(x, y) \notin \mathrm{TC}(G)$ *then* $\mathrm{LCA}(x, y) \subset S$. *More specifically, for all* $v \in V$ *it holds that* $v \in \mathrm{LCA}(x, y)$ *if and only if* $v \in S$ *and for all* $v' \in S$, $(v, v') \notin \mathrm{TC}(G)$.

Observation 7 is implemented by Algorithm 2 in the following way. The set $\mathrm{LCA}(x, y)$ is iteratively constructed by merging the sets $\mathrm{LCA}(x_\ell, y)$. In the merging steps, all those vertices which are predecessors of some other vertices in the set are discarded. Since the vertices are visited in increasing order with respect to a topological ordering N, the sets $A[v, y]$ are finally determined by the time that v is visited. All parents of v are visited before x. This establishes the correctness of Algorithm 2.

Proposition 8. *Let* $t_{\mathrm{merge}}(n_1, n_2)$ *be an upper bound for the time needed by one merge operation on sets of sizes* n_1 *and* n_2. *Then, Algorithm 2 takes time* $O(nm\, t_{\mathrm{merge}}(n, n))$.

Naively, two sets S_1 and S_2 can be merged in time $O(\|S_1\| \cdot \|S_2\|)$. The proof of the following lemma can be found in [4]. It is based on keeping track of forbidden vertices for each pair of vertices.

Lemma 9. *In Line 9 of Algorithm 2, merging can be implemented to run in time* $O(n)$.

Corollary 10. *Algorithm 2 using refined merging solves* ALL-PAIRS ALL LCA *in time* $O(n^2 m)$.

Algorithm 2. ALL-PAIRS ALL LCA

```
   Input: A dag G = (V, E)
   Output: An array A of size n × n where A[x, y] is the set of all LCAs of x and y
1  begin
2      Compute the transitive closure TC(G) and a topological ordering N of G
3      foreach v ∈ V in ascending order of N(v) do
4          foreach y ∈ V with N(v) < N(y) do
5              if (v, y) ∈ TC(G) then A[v, y] ← {v}
6          foreach (v, x) ∈ E do
7              foreach y ∈ V with N(x) < N(y) do
8                  if (x, y) ∈ TC(G) then A[x, y] ← {x}
9                  else A[x, y] ← Merge(A[v, y], A[x, y])
10 end
```

However, if the size k of the sets $\mathrm{LCA}(x, y)$ is small, i.e., $k = o(\sqrt{n})$, the dynamic programming algorithm with naive merging is faster.

Corollary 11. *Algorithm 2 can be modified such that it solves* ALL-PAIRS ALL LCA *in time* $O\left(nmk^2\right)$ *where* k *is the maximum cardinality of LCA sets.*

Note that if we do not know k in advance, we can decide online which merging strategy to use without changing the asymptotical run-time: start Algorithm 2 with naive merging until a vertex is reached in Line 4 having more LCAs with some neighbor vertex (Line 9) than prescribed by some threshold. If this happens, start the algorithm anew with refined merging.

As an immediate consequence we obtain fast algorithms for testing lattice-theoretic properties of posets represented by dags.

Corollary 12. *Testing whether a given dag is a lower semilattice, an upper semilattice, or a lattice can be done in time* $O(nm)$.

Scaling with maximum antichains. Let $G = (V, E)$ be a dag. Let V' be a subset of V. We call V' an *antichain* if no vertex in V' is reachable from another vertex in V'. That means that no two vertices of V' are comparable with respect to the partial order imposed by a dag. A maximum antichain of G is a set $V' \subseteq V$ such that V' is an antichain of maximal cardinality. The *width* of a dag G, denoted by $\mathrm{w}(G)$, is the size of a maximum antichain in dag G. Observe that a maximum antichain is a maximum independent set of the transitive closure. Moreover, the sets $\mathrm{LCA}(x, y)$ are antichains by definition. In particular, their sizes are bounded from above by $\mathrm{w}(G)$.

In contrast to Algorithm 2, which scales quadratically with the size of LCA sets, we give an algorithm for ALL-PAIRS ALL LCA that scales linearly with the width of dags. It is based on solutions for ALL-PAIRS REPRESENTATIVE LCA. We outline our approach.

Suppose we are given a vertex z and want to determine all pairs (x, y) for which z is an LCA. To this end, we employ an ALL-PAIRS REPRESENTATIVE LCA algorithm on G. Obviously, if z is a representative LCA of (x, y) then $z \in \mathrm{LCA}(x, y)$. Thus, if we could force the ALL-PAIRS REPRESENTATIVE LCA algorithm to return z as a representative LCA for (x, y) whenever z is an LCA of x and y, we could answer the above question by solving the representative LCA problem. This can be done as follows.

Algorithm 3. ALL-PAIRS ALL LCA using LCA representatives

Input: A dag $G = (V, E)$
Output: An array A of size $n \times n$ where $A[x, y]$ is the set of all LCAs of x and y

begin
  foreach $z \in V$ **do**
    Compute a topological ordering N such that $N(z)$ is maximal
    Solve ALL-PAIRS REPRESENTATIVE LCA using any algorithm that returns the LCA with highest topological number as representative and get array R
    foreach (x, y) *with* $R[x, y] = z$ **do** $A[x, y] \leftarrow A[x, y] \cup \{z\}$ (by multiset-union)
  Remove elements of multiplicity greater than one from $A[x, y]$ for all $x, y \in V$
end

Algorithm 4. ALL-PAIRS ALL LCA using LCA representatives (improved)

Input: A dag $G = (V, E)$
Output: An array A of size $n \times n$ where $A[x, y]$ is the set of all LCAs of x and y

begin
  Compute a transitive closure $\mathrm{TC}(G)$ and a minimal path cover $\mathcal{P}$ of G
  foreach $p \in \mathcal{P}$ **do**
    Compute a topological ordering N such that $N(z)$ is maximal for all vertices of p
    Solve ALL-PAIRS REPRESENTATIVE LCA with respect to N and get array R
    foreach (x, y) *with* $R[x, y] = z$ *and* $z \in P$ **do**
      $A[x, y] \leftarrow A[x, y] \cup \{z\}$ (by multiset-union)
  Remove elements of multiplicity greater than one from $A[x, y]$ for all $x, y \in V$
end

For a dag $G = (V, E)$ and a vertex $z \in V$, let $N^*(z)$ denote the maximal number of z in any topological ordering of G. It is easily seen that a topological ordering N satisfies $N(z) = N^*(z)$ if and only if for all $x \in V$ such that $N(x) \geq N(z)$, x is reachable from z. This immediately implies a linear-time algorithm to find a corresponding ordering.

Proposition 13. *A topological ordering realizing $N^*(z)$ for any vertex z in a dag can be computed in time $O(n + m)$.*

If we fix a vertex z's number maximizing topological ordering, then z is the rightmost CA of all vertex pairs (x, y) such that $z \in \mathrm{LCA}(x, y)$. Now clearly, our strategy is to iterate for each $x \in V$ over the orderings that maximize $N^*(x)$. Note that the algorithms in Sect. 3 and in [7,15] naturally return the vertex z with the highest number $N(z)$ (for a fixed topological ordering) among all LCAs of any pair (x, y). This leads to Algorithm 3 and Theorem 14.

Theorem 14. *Algorithm 3 solves* ALL-PAIRS ALL LCA *in time $O(\min\{n^{3+\mu}, n^2 m\})$.*

Again, since we have $\mu < 0.575$, this yields an $O(n^{3.575})$ algorithm on dense dags. A key observation is that an algorithm for ALL-PAIRS REPRESENTATIVE LCA that outputs, with respect to a fixed topological ordering N, the vertex with the highest number as a representative LCA, does it for all z with $N(z) = N^*(z)$ in parallel. We aim at maximizing topological numbers simultaneously for as many vertices as possible. This can easily be reached for vertices in paths.

Proposition 15. *A topological ordering maximizing $N^*(z)$ for all vertices z in any path p of a dag G simultaneously, can be computed in time $O(n + m)$. Moreover, this can be done for all vertex subsets of the given path p.*

This proposition implies that given such an ordering, it is possible to process a path p in only one iteration of the algorithm, i.e., only one call of an algorithm for the ALL-PAIRS REPRESENTATIVE LCA problem is needed for the vertices in p. Thus, we can reduce the running time if we minimize the number of paths to be processed.

For a dag $G = (V, E)$, a *path cover* $\mathcal{P}$ of G is a set of paths in G such for every $v \in V$ there exists at least one path $p \in \mathcal{P}$ such that v lies on p. A minimum path cover is a path cover $\mathcal{P}$ such that $\|\mathcal{P}\|$ is minimized. It is known that a minimum path cover can be computed in time $O(\min\{nm, n^{2.5}\})$ [5]. This suggests Algorithm 4.

Actually, Algorithm 4 is an improvement over Algorithm 3 if we know that, on the one hand, the size of a minimal path cover is an upper bound on LCA set-sizes, and at most n on the other hand. Fortunately, the famous Dilworth's theorem does exactly this.

Lemma 16 (Dilworth). *For each dag G, $\mathrm{w}(G)$ equals the size of a minimum path cover of G.*

Indeed, we need both directions of Dilworth's theorem to obtain the following.

Theorem 17. *Algorithm 4 solves* ALL-PAIRS ALL LCA *in time* $O\big(\min\{n^{2+\mu} \cdot \mathrm{w}(G), nm \cdot \mathrm{w}(G)\}\big)$.

Algorithm 4 elegantly scales with dag widths automatically without saying which algorithmic branch should be used as it is necessary for scaling with maximum LCA set-sizes. However, Theorem 17 does not yield as many benefits as may be expected. For instance, rooted binary trees can be viewed as dags. The width of a tree is the number of its leaves which is in fully binary trees $O(n)$. In contrast to this, each pair has exactly one LCA. As another example, in the experimental setting of Internet dags mentioned in the introductory section, we obtained a width of 9,604 (i.e., there is an antichain containing around 85% of all vertices), a maximum LCA set-size of 27, and an average LCA set-size of 9.66. All this shows that improving our algorithms towards a linear-scaling behavior with respect to LCA set-sizes is essential.

5 Shortest Distance Common Ancestor Problems

Again, we start by giving a dynamic programming algorithm that solves the ALL-PAIRS SHORTEST DISTANCE CA problem in time $O(nm)$, which is optimal on sparse dags. The $O(nm)$ bound follows readily. The correctness is based on Observation 18.

Observation 18. *Let $G = (V, E)$ be a dag and let x, y be two vertices that have at least one CA. Furthermore, let $x_1, \ldots, x_k$ be the parents of x and let $Z = \{z_1, \ldots, z_k\}$ be the set of the corresponding shortest distance CAs of (x_i, y) for $1 \leq i \leq k$. Then, for the shortest distance CA z of x and y it holds that $z = \mathrm{argmin}_{z' \in Z \cup \{x\}} d(z', x) + d(z', y)$.*

Theorem 19. *Algorithm 5 solves the* ALL-PAIRS SHORTEST DISTANCE CA *problem in time $O(nm)$.*

The following theorem generalizes results from Bender et al. [7] to weighted dags and to computing shortest distance ancestors in addition to shortest ancestral distances. We reach this result by modifying the natural reduction of shortest ancestral distances to shortest distances in dags [7] and using sampling techniques already used for computing witnesses for shortest paths [20, 24]. A detailed description can be found in [4].

Algorithm 5. ALL-PAIRS SHORTEST DISTANCE CA

Input: A dag $G = (V, E)$ with a weight function $w : E \to \mathbb{R}$
Output: An array M of size $n \times n$ where $M[x, y]$ is a shortest distance CA of x and y

```
1 begin
2     Compute the all-pairs shortest distance matrix D and a a topological ordering N of G
3     foreach (x, y) with D[x, y] < ∞ do M[x, y] ← x
4     foreach v ∈ V in ascending order of N(v) do
5         foreach (v, x) ∈ E do
6             foreach y ∈ V with N(y) ≥ N(v) do
7                 if D[M[v, y], x] + D[M[v, y], y] < D[M[x, y], y] + D[M[x, y], y] then
                      M[x, y] ← M[v, y]
8 end
```

Theorem 20. *Let $\mathcal{A}$ be any* APSD *algorithm with running time $t_{\mathcal{A}}(n, m)$ on weighted dags with n vertices and m edges. Then, there is an algorithm for* ALL-PAIRS SHORTEST DISTANCE CA *with running time $\tilde{O}(t_{\mathcal{A}}(n, m)+n^2)$. Here, $t_{\mathcal{A}}$ is required to satisfy $t_{\mathcal{A}}(O(n), O(m)) = O(t_{\mathcal{A}}(n, m))$.*

We turn our attention to the ALL-PAIRS SHORTEST DISTANCE LCA problem. It seems to be inherently difficult to compute shortest distance lowest common ancestors or even shortest distances directly. Nevertheless, we obtain non-trivial upper bounds by using solutions for the ALL-PAIRS ALL LCA problem. A generic solution for finding shortest distance LCAs looks as follows (supposed that LCA sets are stored as lists):

1. Solve APSD on dag G (understood as a weighted dag).
2. Solve ALL-PAIRS ALL LCA on dag G (understood as an unweighted dag).
3. For each pair (x, y) choose in time $O\big(\|\mathrm{LCA}(x, y)\|\big)$ the vertex $z \in \mathrm{LCA}(x, y)$ which minimizes the ancestral distance $d(z, x) + d(z, y)$ and set $L[x, y] = z$.

Theorem 21. *Let $\mathcal{A}$ be an algorithm solving* ALL-PAIRS ALL LCA *in time $t_{\mathcal{A}}(n, m)$. Then,* ALL-PAIRS SHORTEST DISTANCE LCA *can be solved in time $O(t_{\mathcal{A}}(n, m))$.*

The following corollaries follow immediately from the above theorem, Theorems 14 and 17 and Corollary 11.

Corollary 22. *The* ALL-PAIRS SHORTEST DISTANCE LCA *problem can be solved in time $O(\min\{n^2 m, nmk^2\})$ where k is the maximum cardinality of all LCA sets.*

Corollary 23. *The* ALL-PAIRS SHORTEST DISTANCE LCA *problem can be solved in time $O\big(\min\{n^{2+\mu} \cdot \mathrm{w}(G), nm \cdot \mathrm{w}(G)\}\big)$. For the first bound, i.e., on dense dags, edge weights are limited to be integer constants.*

6 Conclusion and Open Problems

Figure 1 shows an overview (and the according interpretation) of the relations between several problems with output complexity $\Omega(n^2)$. For example, Bender et al. [6] have shown how to solve ALL-PAIRS REPRESENTATIVE LCA using APSD and how to solve TRANSITIVE CLOSURE using ALL-PAIRS REPRESENTATIVE LCA, whereas Kowaluk et al. [15] have figured out the relation indicated between ALL-PAIRS REPRESENTATIVE LCA and MAXIMUM BOOLEAN WITNESSES. Our contributions are

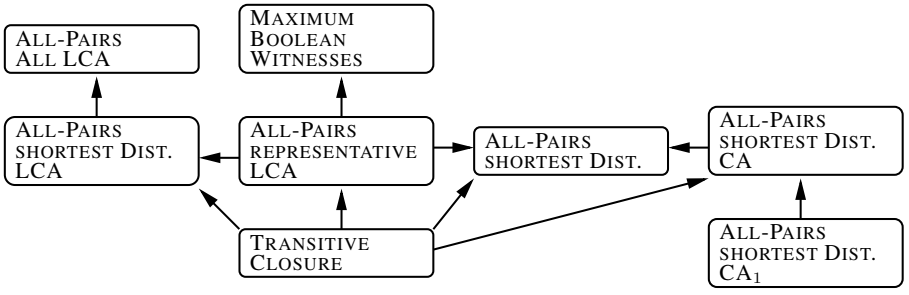


Fig. 1. Structural overview of problem complexities. The interpretation is as follows: let A and B be two problems and I_A and I_B two instances with solutions $S(I_A)$, $S(I_B)$ respectively. An arrow from A to B means that $S(I_A)$ can be *directly read* from $S(I_B)$, where I_B is constructed from I_A in linear time. More formally, $I_B = g(I_A)$ for a function g linear-time computable in $|I_A|$ and $S(I_A) = f(I_A, S(g(I_A)))$, where $I_B = g(I_A)$ f is linear-time computable in $|I_A| + |S(I_B)|$.

also integrated. Note that the arrow between TRANSITIVE CLOSURE and ALL-PAIRS SHORTEST DISTANCE CA restricted to instances with constant edge weights is intentionally missing. It seems that a solution to the first problem cannot be directly read from a solution of the latter, if the latter one's instance is restricted to constant edge weights.

In this paper, we have described and efficiently solved all-pairs ancestor problems on dags both for sparse and dense instances. Our solutions for ALL-PAIRS ALL LCA exhibit nice scaling properties. Moreover, upper bounds for the scaling factors beautifully coincide with Dilworth's theorem.

ALL-PAIRS SHORTEST DISTANCE CA is widely understood. On sparse graphs our solution is optimal, and on dense graphs the gap between APSD and ALL-PAIRS SHORTEST DISTANCE CA is shown to be at most polylogarithmic. On the other hand, our algorithms for ALL-PAIRS SHORTEST DISTANCE LCA rely fully on the solution of the ALL-PAIRS ALL LCA problem. We are left with intriguing open questions:

References

1. Aho, A., Hopcroft, J., Ullman, J.: On finding lowest common ancestors in trees. SIAM J. Comput. 5(1), 115–132 (1976)
2. Aït-Kaci, H., Boyer, R., Lincoln, P., Nasr, R.: Efficient implementation of lattice operations. ACM Trans. Program. Lang. Syst. 11(1), 115–146 (1989)
3. Alon, N., Naor, M.: Derandomization, witnesses for boolean matrix multiplication and construction of perfect hash functions. Algorithmica 16(4–5), 434–449 (1996)
4. Baumgart, M., Eckhardt, S., Griebsch, J., Kosub, S., Nowak, J.: All-pairs common-ancestor problems in weighted dags. Technical Report TUM-I0606, Institut für Informatik, TU München (April 2006)
5. Benczúr, A., Förster, J., Király, Z.: Dilworth's theorem and its application for path systems of a cycle - implementation and analysis. In: Nešetřil, J. (ed.) ESA 1999. LNCS, vol. 1643, pp. 498–509. Springer, Heidelberg (1999)

6. Bender, M., Pemmasani, G., Skiena, S., Sumazin, P.: Finding least common ancestors in directed acyclic graphs. In: Proc. 12th Annual Symposium on Discrete Algorithms (SODA'01), pp. 845–854 (2001)
7. Bender, M., Farach-Colton, M., Pemmasani, G., Skiena, S., Sumazin, P.: Lowest common ancestors in trees and directed acyclic graphs. J. Algorithms 57(2), 75–94 (2005)
8. Berkman, O., Vishkin, U.: Finding level-ancestors in trees. J. Comput. Syst. Sci. 48(2), 214–230 (1994)
9. Cole, R., Hariharan, R.: Dynamic LCA queries on trees. SIAM J. Comput. 34(4), 894–923 (2005)
10. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. J. Symbolic Computation 9(3), 251–280 (1990)
11. Coppersmith, D.: Rectangular matrix multiplication revisited. J. Complexity 13(1), 42–49 (1997)
12. Czumaj, A., Kowaluk, M., Lingas, A.: Faster algorithms for finding lowest common ancestors in directed acyclic graphs. Electronic Colloquium on Computational Complexity (ECCC), TR06-111 (2006)
13. Gao, L.: On inferring autonomous system relationships in the Internet. IEEE/ACM Trans. Networking 9(6), 733–745 (2001)
14. Harel, D., Tarjan, R.: Fast algorithms for finding nearest common ancestors. SIAM J. Comput. 13(2), 338–355 (1984)
15. Kowaluk, M., Lingas, A.: LCA queries in directed acyclic graphs. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 241–248. Springer, Heidelberg (2005)
16. Moret, B., Nakhleh, L., Warnow, T., Linder, C., Tholse, A., Padolina, A., Sun, J., Timme, R.: Phylogenetic networks: Modeling, reconstructibility, and accuracy. IEEE/ACM Trans. Comput. Biology Bioinform. 1(1), 13–23 (2004)
17. Nakhleh, L., Wang, L.: Phylogenetic networks: Properties and relationship to trees and clusters. In: Priami, C., Zelikovsky, A. (eds.) Transactions on Computational Systems Biology II. LNCS (LNBI), vol. 3680, pp. 82–99. Springer, Heidelberg (2005)
18. Nykänen, M., Ukkonen, E.: Finding lowest common ancestors in arbitrarily directed trees. Inf. Process. Lett. 50(1), 307–310 (1994)
19. Schieber, B., Vishkin, U.: On finding lowest common ancestors: Simplification and parallelization. SIAM J. Comput. 17(6), 1253–1262 (1988)
20. Seidel, R.: On the all-pairs-shortest-path problem in unweighted undirected graphs. J. Comput. Syst. Sci. 51(3), 400–403 (1995)
21. Tarjan, R.: Applications of path compression on balanced trees. J. ACM 26(4), 690–715 (1979)
22. Wang, B., Tsai, J., Chuang, Y.: The lowest common ancestor problem on a tree with an unfixed root. Inf. Sci. 119(1–2), 125–130 (1999)
23. Wen, Z.: New algorithms for the LCA problem and the binary tree reconstruction problem. Inf. Process. Lett. 51(1), 11–16 (1994)
24. Zwick, U.: All pairs shortest paths using bridging sets and rectangular matrix multiplication. J. ACM 49(3), 289–317 (2002)

Streaming Algorithms for Data in Motion

M. Hoffmann[1], S. Muthukrishnan[2,⋆], and Rajeev Raman[1]

[1] Department of Computer Science, University of Leicester, Leicester LE1 7RH, UK
{m.hoffmann,r.raman}@cs.le.ac.uk
[2] Division of Computer and Information Sciences, Rutgers University, Piscataway, NJ 08854-8019, USA
muthu@cs.rutgers.edu

Abstract. We propose two new data stream models: the *reset* model and the *delta* model, motivated by applications to databases, and to tracking the location of spatial points.

We present algorithms for several problems that fit within the stream constraint of polylogarithmic space and time. These include tracking the "extent" of the points and L_p sampling.

1 Introduction

The area of data stream algorithms is the study of algorithmic problems in applications where the data arrives at extremely high speed. Further, the memory available is quite small, so the entire data can not be stored. Thus algorithms have to solve data analyses problems to the best extent they can with sublinear—often polylogarithmic—space, and fast per-item processing. Under these constraints, computing most significant functions on the data stream is provably impossible. So, algorithms typically tend to produce an approximation and are often randomized. Such algorithms have been developed for a number of data analyses problems including estimation of various norms [8,28], clustering [23,11,24], histograms [21], quantiles [22], heavy-hitters [33] etc.

The quintessential application where this arises is in processing internet traffic logs such as the headers on IP packets and IP sessions. Since packets are forwarded at blistering speeds in the internet, these traffic logs are generated at very high speed; memory working at this speed within the routing elements is quite scarce, and the logs have to be processed in real time to solve number of queries involved in monitoring the internet for security breaches, accounting, billing etc. This application has become fairly well developed, with tutorials and a number of research papers in networking conferences [38,16,32], specialized database systems aimed at such applications discussed in workshops, tutorials and a number of papers in database conferences [1,37,9], and fundamental algorithmic and complexity-theoretic issues being studied in the theoretical computer science community [8,6,35].

Recently, there is growing interest in *other* data streams. In particular, an emerging area is one of *spatial* streams. These are location-specific streams. For

⋆ Supported by NSF EIQ 0087022, NSF ITR 0220280 and NSF EIA 02-05116.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 294–304, 2007.

example, consider a vast collection of moving objects. Over time, their locations change and one wants to track various functions of their locations. GPS (Global Positioning System) enables tracking locations of devices over large scale. Wireless service providers are looking to use this or other infrastructure to enable location-aware services, which is a burgeoning industry [3]. Other industries are actively engaged in developing applications based on tracking the location of various objects: delivery trucks (eg. [2]), University campus buses [4], cars, stocks of goods, biosensors [5], astronomical studies via satellite and ground-based systems [36], etc.

We propose two new models. Both models have relevance to existing scenarios (e.g. databases) as well as spatial streams. We study a fundamental problems, e.g., accurately estimating the "extent" of moving objects, under both models and design space- and time-efficient algorithms for these problems. As in the study of standard data streams [9,35], exact calculation of most functions is impossible in our new model, and algorithms tend to be probabilistic and approximate. Our algorithms rely on new versions of techniques used commonly in streaming problems.

1.1 The Reset Model

The first model is the *reset* model, which is as follows. There is a vector $\boldsymbol{d} = (d_1, \ldots, d_n)$ in $\Re^n$, where n is assumed to be very large. We assume that $d_i = 0$ for $i = 1, \ldots, n$ initially, and consider the following model of updates:

Reset model: The updates appear in a stream (i, x), implying that $d_i \leftarrow x$.

For spatial data, each element of the vector may be a point in $\Re^d$, and an update may reset one or more coordinates of a given point.

Remark 1. In contrast, in existing data stream models (e.g. the *turnstile* model [35, p9]), an update (i, x) implies $d_i \leftarrow d_i + x$. So, instead of increments as in previous models, an update resets data values.

The reset model also differs from the *dynamic geometric* model considered in e.g. [27] processes a stream of intermixed operations $INS(p)$ and $DEL(q)$, for points p and q. One cannot reduce the reset model to the dynamic geometric model, as simulating a reset by a DEL followed by an INS gives information about the previous location of the point that is not readily available to the algorithm in the reset model (recall that $\boldsymbol{x}$ and $\boldsymbol{y}$ are not stored explicity).

Motivations. The reset model has two independent motivations. First, there is a selftuning approach to selectivity estimation in databases [7]. The query processor has a "summary" that describes some (not necessarily the current) state of the data distribution in the database. It selects query plans for execution using the summary; but when a query is executed, as a by-product, it gets to know the actual values the query intersects. The query optimizer needs to use this feedback to update its knowledge of the data distribution. Each time it knows an actual value, it corresponds to a reset. The goal is for the query optimizer to estimate

various parameters of interest on the data distribution under this stream of resets. This precisely corresponds to the reset model of data streams. The second motivation arises in streams of locations. For example, in traffic sensor information systems [34], the stream corresponds to location-dependent queries from clients that are passed on to one of several servers by mediators. Each server gets a subset of clients; a given client may go to multiple servers before being assigned to one of them again. Servers need to track statistics on the locations of clients assigned to them. At each server, a new location-dependent query for a client is precisely a reset operation and they need to tracking statistics on their location corresponds using "summaries". We have provided only high level view of how reset model of streaming fits both these applications. More details are in [7,34].

We now describe our results in the reset and delta models. In what follows, an (ϵ, δ)-estimator of a quantity x is a random variable $\hat{x}$ such that $\Pr[(1-\epsilon)x \leq \hat{x} \leq (1+\epsilon)x] \geq 1-\delta$.

Results. The reset model is significantly more restrictive than known data stream models, so it presents significant algorithmic challenges. Indeed, for the reset model, only very basic techniques appear to be useful, such as random sampling, using which several problems such as approximate quantiles or the 1-median problem can be solved easily (see e.g. [29]). We obtain results in this model by assuming that the update stream is *monotone*, i.e. an update always causes d_i to increase.

Let $\boldsymbol{d}$ in $\Re^n$ be updated as in the reset model. We consider two fundamental problems—that of estimating $||\boldsymbol{d}||_p$ for some $p \geq 0$, and L_p sampling, i.e. choosing an element from $\{1, \ldots, n\}$ such that i is chosen with probability proportional to $|d_i|^p$. If $p = 0$ then the problem reduces to calculating the L_0 norm of a stream which can be solved as in [12]. In fact, we can assume that $p = 1$, and that d_i is always non-negative, without loss of generality. Specifically, let $1 \geq \delta, \epsilon > 0$ be two constants specified by the user. We give an algorithm that, if the if the update stream is monotone:

- for any k, $1 \leq k \leq n$, returns on request an (ϵ, δ) estimator for $\boldsymbol{d}^{(k)}$, the sum of the k largest elements in $\boldsymbol{d}$. In particular, when $k = n$, it returns an (ϵ, δ) estimator for $||\boldsymbol{d}||_1$,
- returns on request an integer i from $\{1, \ldots, n\}$ with probability $\tilde{p}_i$ where $\frac{|d_i|}{||\boldsymbol{d}||_1(1+\epsilon)} - \epsilon/n \leq \tilde{p}_i \leq \frac{(1+\epsilon)|d_i|}{||\boldsymbol{d}||_1}$.

In case the algorithm is presented with a non-monotone update sequence, it (in essence) ignores any non-monotone updates, and may therefore return an incorrect answer. However, in case the update sequence is 'significantly' non-monotone, this can be detected, in the following sense:

- Let d_i^* be the maximum value attained by d_i over the sequence of updates, and let $\boldsymbol{d^*} = (d_1^*, \ldots, d_n^*)$. If the update stream is non-monotone, with probability $1 - \delta$, we detect non-monotonicity before $||\boldsymbol{d}||_1(1+\epsilon)^3 \leq ||\boldsymbol{d^*}||_1$.

The algorithm uses polylog(n) space, and polylog(n) time to process each update and query. The approach is similar to one used in [14], but we consider a wider range of problems than were considered there.

We make a few remarks about the results. The problem of computing the L_∞ norm of $\boldsymbol{d}$ (the maximum element) is known to be hard in the turnstile model, even if values are increasing monotonically. However, this problem is trivial in the monotone reset model. Conversely, the problem of computing the sum of the values in $\boldsymbol{d}$ is trivial in the turnstile model, but requires a little effort in the reset model. It is not known how to solve the problem of L_p sampling, or to estimate the sum of the k largest elements, for any k, in the turnstile model.

1.2 Delta Model

Let $\boldsymbol{x}, \boldsymbol{y} \in \Re^n$; initially, $\boldsymbol{x} = \boldsymbol{y} = 0$. The vectors are updated as follows:

Delta model: The updates appear in a stream $(i, \Delta x, \Delta y)$, implying that $x_i \leftarrow x_i + \Delta x$ and $y_i \leftarrow y_i + \Delta y$.

Again, $\boldsymbol{x}, \boldsymbol{y}$ hold the coordinates of n points in $\Re^2$. This model is a direct generalisation of the classical turnstile model to two dimensions. As in Remark 1, this model is different from the geometric stream models considered in the literature. Although we are aware of no "physically" spatial application where the delta model is currently appropriate[1] this model is also motivated by database contexts, where several of the myriad motivations of the (one-dimensional) turnstile model carry over.

Results. We consider the problem of computing the "extent" of the point set S given by $\boldsymbol{x}$ and $\boldsymbol{y}$. Let $\boldsymbol{d}$ be the vector of L_p distances of S relative to some centre $c = (c_x, c_y)$, which is specified as part of the query. The objective is to be able to estimate $||\boldsymbol{d}||_q$ for some q. If $p = q$, the problem is equivalent to that of estimating $||\boldsymbol{x} - c_x||_p + ||\boldsymbol{y} - c_y||_p$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are the vectors of x- and y- coordinates respectively. Thus, the problem factors trivially into two 1-D L_p norm estimations. The truly interesting case is when $p = 2$ and $q = 1$; this corresponds to computing the sum of the (Euclidean) distance of the points from a given centre. We show:

- For any fixed $\epsilon > 0$ and $\delta > 0$, we give a data structure that, given any $c = (c_x, c_y)$, returns an (ϵ, δ)-estimator of $||\boldsymbol{d}||_1$, where $\boldsymbol{d}$ is the vector of L_2 distances of points in S from c.

The result assumes that the choice of c is independent of the random choices made by the algorithm (more precisely, that the operation sequence is generated by an oblivious adversary). The algorithm uses polylog(n) space, and polylog(n) time to process each update and query.

In the remainder of the paper, we give the results on the reset and delta model in Section 2 and Section 3, and conclude with some open problems.

[1] However, one can easily conceive of devices that can report the change in position since the last report, without being aware of their current position.

2 Reset Model

Let $\boldsymbol{d}$ in $\Re^n$ be updated as in the reset model. Recall that an update stream is *monotone* if an update always causes d_i to increase. We show:

Lemma 1. *For any constants $1 \geq \delta, \epsilon > 0$, there is an algorithm that returns on request an (ϵ, δ) an (ϵ, δ)-estimator $\tilde{d}$ of $\boldsymbol{d}^{(k)}$, for any given integer k, $1 \leq k \leq n$, provided that the update stream is monotone. The algorithm ignores non-monotone operations.*

The algorithm takes $O(\tau s)$ time to process an update and uses $O(\sigma s)$ space, where τ and σ are the time and space required to return an $(\epsilon', \delta/(2(s+1)))$-estimator of the number of distinct elements in a stream of items from $\{1, \ldots, n\}$, where $\epsilon' = \epsilon/3$ and $s = \lceil 1 + \log_{1+\epsilon'}(n/(\epsilon')^2) \rceil$.

Proof. We begin with the case $k = n$, i.e., an estimator for $||\boldsymbol{d}||_1$. We conceptually create an infinite sequence of *buckets*, $\ldots, B_{-1}, B_0, B_1, \ldots$, where the i-th bucket has *weight* $w_i = (1+\epsilon')^i$ and $\epsilon' = \epsilon/3$. At any given time, only a contiguous sequence of buckets $B_{r-s}, \ldots, B_r$ is *active*. Let n_i denote the number of distinct elements from $\{1, \ldots, n\}$ that are placed into B_i. Associated with each active bucket B_i is the assumed data structure that returns an $(\epsilon', \delta/(2(s+1)))$-estimator $\tilde{n}_i$ of the n_i. Activating a bucket means allocating memory for such a data structure and appropriately initializing it. For any d, we let $\rho(d)$ be the integer k such that $\sum_{j=-\infty}^{k-1} w_j < d \leq \sum_{j=-\infty}^{k} w_j$.

To process an update (i, d), we determine $r_i = \rho(d)$ and distinguish three cases:

- If $r_i < r - s$ we update no buckets.
- If $r - s \leq r_i \leq r$ we place the integer i into *all* buckets $B_{r-s}, \ldots, B_{r_i}$.
- If $r_i > r$, we deactivate buckets $B_{r-s}, \ldots, B_{r_i-s-1}$, activate any inactive buckets among $B_{r_i-s}, \ldots, B_r$, place the integer i into all these buckets, and finally set $r := r_i$.

To estimate $||\boldsymbol{d}||_1$, we proceed as follows. For $j = r-s, \ldots, r$ we obtain an estimate $\tilde{n}_j$ of the number of distinct values placed in B_j and return $\tilde{d} = \sum_{j=r-s}^{r} \tilde{n}_j \cdot w_j$. We now calculate the error in this estimate.

First note that by definition, $\sum_{j=-\infty}^{r_i-1} w_i \leq d_i$, and so $\sum_{j=-\infty}^{r_i} w_j \leq (1+\epsilon')d_i$. Since $\tilde{n}_i \leq (1+\epsilon')n_i$ for all $i = r-s, \ldots, r$, it follows that with probability at least $1 - \delta/2$:

$$\begin{aligned}
\tilde{d} &= \textstyle\sum_{j=r-s}^{r} \tilde{n}_j \cdot w_j \leq (1+\epsilon') \sum_{j=r-s}^{r} n_j \cdot w_j \\
&\leq (1+\epsilon')^2 \sum_{j=r-s}^{r} d_i \leq (1+\epsilon/3)^2 ||\boldsymbol{d}||_1 \\
&\leq (1+\epsilon)||\boldsymbol{d}||_1
\end{aligned}$$

To lower-bound the estimate, take $s = \lceil 1 + \log_{1+\epsilon'}(n/(\epsilon')^2) \rceil$, and note that:

$$\sum_{j=r-s}^{r_i} w_j \geq d_i - \sum_{j=-\infty}^{r-s-1} w_j \geq d_i - d_{max}/(1+\epsilon')^s \geq d_i - (\epsilon')^2 d_{max}/(n(1+\epsilon'))$$

Summing over i, we get that:

$$\sum_{j=r-s}^{r} n_j \cdot w_j \geq \sum_{i=1}^{n} d_i - (\epsilon')^2 d_{max}/(1+\epsilon') \geq (1-\epsilon') \sum_{i=1}^{n} d_i$$

As with the upper bound, we note that the probability that $\tilde{n}_i$ is not a significant underestimate is at least $(1-\delta/2)$. This gives the desired probability bound.

We now describe the general case. We omit the analysis of the failure probability, which is as above. For integer x, $r-s \leq x \leq r$, define $f(x) = k\sum_{i=r-s}^{x} w_i + \sum_{i=x+1}^{r} n_i w_i$. Let t be the largest index i such that $n_i \geq k$. Arguing as above, $(1-\epsilon')\boldsymbol{d}^{(k)} \leq f(t) \leq (1+\epsilon')\boldsymbol{d}^{(k)}$, as $f(t)$ is precisely how $\boldsymbol{d}^{(k)}$ is represented within the buckets. Now let $\tilde{f}(x) = k\sum_{i=r-s}^{x} w_i + \sum_{i=x+1}^{r} \tilde{n}_i w_i$. The algorithm chooses the largest index t' such that $\tilde{n}_{t'} \geq k$ and returns $\tilde{f}(t')$ as the estimator. We now bound the error $|f(t) - \tilde{f}(t')|$, considering first the case $t' \geq t$.

Since $\tilde{n}_{t'} \geq k$, except with negligible probability, we have $n_{t'} \geq k/(1+\epsilon')$. Thus, $k/(1+\epsilon) \leq n_{t'} \leq \ldots \leq n_{t+1} < k$, and:

$$\begin{aligned} |f(t) - \tilde{f}(t')| &\leq \sum_{i=t+1}^{r} w_i|\tilde{n}_i - n_i| + \sum_{i=t+1}^{t'} w_i(k - n_i) \\ &\leq \sum_{i=t+1}^{r} \epsilon' n_i w_i + \sum_{i=t+1}^{t'} \epsilon' w_i n_i \leq \epsilon' f(t) \end{aligned}$$

If $t > t'$ then for $i = t'+1, \ldots, t$, $n_i \geq k > \tilde{n}_i \geq (1-\epsilon')n_i$. Thus:

$$\begin{aligned} |f(t) - \tilde{f}(t')| &\leq \sum_{i=t'+1}^{r} w_i|\tilde{n}_i - k| + \sum_{i=t'}^{t} w_i(k - \tilde{n}_i) \\ &\leq \sum_{i=t'+1}^{r} \epsilon' n_i w_i + \sum_{i=t+1}^{t'} \epsilon' w_i n_i \leq \epsilon' f(t) \end{aligned}$$

Thus, we have that $(1-\epsilon')^2\boldsymbol{d}^{(k)} \leq \tilde{f}(t') \leq (1+\epsilon')^2\boldsymbol{d}^{(k)}$, and the result follows as above. □

For non-monotone sequences, let d_i^* be the maximum value attained by d_i over the sequence of updates, and let $\boldsymbol{d^*} = (d_1^*, \ldots, d_n^*)$. We now show:

Theorem 1. *For any constants $1 \geq \delta, \epsilon > 0$ there is an algorithm that, if the update stream is monotone:*

(a) returns on request an (ϵ, δ)-estimator of $||\boldsymbol{d}||_1$,
(b) for any $k, 1 \leq k \leq n$, returns on request an (ϵ, δ)-estimator of $\boldsymbol{d}^{(k)}$ and
(c) returns on request an element i from $\{1, \ldots, n\}$ with probability $\tilde{p}_i$ where $\frac{|d_i|}{||\boldsymbol{d}||_1(1+\epsilon)} - \epsilon/n \leq \tilde{p}_i \leq \frac{(1+\epsilon)|d_i|}{||\boldsymbol{d}||_1}$.

Non-monotone updates are ignored by the algorithm. Furthermore,

(d) if the update stream is non-monotone, the algorithm, with probability at least $1-\delta$ *detects the non-monotonicity before* $||\boldsymbol{d}||_1(1+\epsilon)^3 \leq ||\boldsymbol{d^*}||_1$*, where* $\boldsymbol{d^*}$ *is as above.*

The algorithm uses polylog(n) space and time to process an update.

Proof. (Sketch) Lemma 1 essentially proves (a) and (b); it only remains to note that either the Flajolet-Martin algorithm or its refinements [18,10] or the algorithm of [12, Theorem 4] estimates the number of distinct items in a stream in at most polylog(n) space and time.

We now outline a solution to (c). Ideally, for every active bucket B_i we would choose a random representative, such that each (distinct) element in B_i is chosen with probability $1/n_i$ (and update the representative when a bucket is updated). When asked for a sample from $\boldsymbol{d}$, we would return one of the representatives, choosing the representative of bucket i with probability proportional to $n_i w_i$. Then, the probability of selecting i is (taking $r_i = \rho(d_i)$):

$$\sum_{j=r-s}^{r_i} \frac{1}{n_j} \frac{n_j w_j}{\sum_{j=r-s}^{r} n_j w_j} = \frac{\sum_{j=r-s}^{r_i} w_j}{\sum_{j=r-s}^{r} n_j w_j}.$$

As before, the numerator is essentially d_i and the denominator is essentially $||\boldsymbol{d}||_1$. Other sources of errors are that we are able only to sample a bucket representative with a distribution that is arbitrarily close to uniform (using approximately min-wise independent permutations, see [15]), and that we must work with probabilistic estimates $\tilde{n}_j$ rather than n_j. These are handled as in Lemma 1.

Finally, we consider (d), and indicate how to detect non-monotonicity. After a non-monotonic update, the updated element may continue to exist incorrectly in a (higher-numbered) bucket. We let n_i^* and n_i denote the number of elements placed in B_i by the algorithm, and the number of elements that 'ought' to be in B_i, respectively. If $||\boldsymbol{d}||_1(1+\epsilon)^3 < ||\boldsymbol{d^*}||_1$, where $\boldsymbol{d^*}$ is as above, then $(1+\epsilon)\sum_{j=r-s}^{r} n_i w_i \leq \sum_{j=r-s}^{r} n_i^* w_i$. Thus, there is at least one bucket B_i for which $(1+\epsilon)n_i w_i \leq n_i^* w_i$. This can only happen if at least $\epsilon n_i^*/(1+\epsilon)$ of the n_i^* elements that were ever in B_i were updated non-monotonically. This can be detected with the required probability by keeping a test sample of $m = O(\log \delta/(1-\epsilon/(1+2\epsilon)))$ elements chosen approximately uniformly at random from the n_i^* elements in B_i, together with their current values, and checking to see if any of the elements in the test sample are updated non-monotonically. (The sampling is done as in [15].) □

Remark 2. In contrast, the problem under the general case when updates are not monotonic is provably impossible to solve under streaming constraints (e.g. polylog space). We can reduce the problem to that of estimating $|A| - |A \cap B|$ for two sets A and B given in standard turnstile model, which in turn, can be shown to have a nearly linear space lower bound for $1+\epsilon$ approximation by known lower bounds in Communication Complexity. We omit showing this process here because it is quite standard once the problem of estimating $|A| - |A \cap B|$ is given as the key.

3 Delta Model

Recall that in this model, a set S of n 2-D points is represented by two vectors $\boldsymbol{x}, \boldsymbol{y} \in \Re^n$. We consider the following problem:

- Let $\boldsymbol{d}$ be the vector of L_2 distances of the points of S relative to some centre $c = (c_x, c_y)$ (the centre is specified as part of the query). The objective is to be able to estimate $||\boldsymbol{d}||_1$.

We begin by considering a simple 1-D version of the problem. Consider a vector $\boldsymbol{x}$ of size n that contains x-coordinates, and suppose that we process a sequence of updates $(i, \Delta x)$, which sets $x_i \leftarrow x_i + \Delta x$. Then, we have that:

Proposition 1. *For any fixed $\epsilon > 0$ and $\delta > 0$, there is a data structure that processes each update in poly-log time, uses poly-log space, and in poly-log time, given any c, returns an (ϵ, δ)-estimator of $\sum_{i=1}^{n} |x_i - c|$.*

Proof. We essentially observe that the standard algorithm [28] for estimating $||\boldsymbol{x}||_1$ also computes the above quantity. The algorithm maintains a *sketch* of $\boldsymbol{x}$, which is simply a sequence of values $\boldsymbol{x} \cdot \boldsymbol{r_i}$, for $i = 1, \ldots, k$, and where r_i is a pseudo-random vector drawn from an *1-stable* distribution. It can be shown that the (normalised) median of $\boldsymbol{x} \cdot \boldsymbol{r_i}$, for $i = 1, \ldots, k$, for sufficiently large k is an (ϵ, δ)-estimator of $||\boldsymbol{x}||_1$. In order to estimate $||\boldsymbol{x} - \boldsymbol{c}||_1$, where $\boldsymbol{c} = (c, c, \ldots, c)$, we simply compute a sketch of $\boldsymbol{x} - \boldsymbol{c}$, by computing $\boldsymbol{x} \cdot \boldsymbol{r_i} - \boldsymbol{c} \cdot \boldsymbol{r_i}$, for $i = 1, \ldots, k$. This can be done in $O(1)$ time per sketch element provided the sum of the values in $\boldsymbol{r_i}$ is known. Since $\boldsymbol{r_i}$ is not stored explicitly, this is not trivial, but can be done by using so-called *range-summable* random variables [21, Lemma 2].

The 2-D version of the problem can be reduced to two 1-D problems by considering the projections of the points e.g. on to orthogonal axes. Estimating the extent based on one pair of axes, however, would yield a $(\sqrt{2}+\epsilon)$-approximation in the worst case. Considering the projections of the points along several pairs of orthogonal axes allows us to greatly improve the accuracy:

Lemma 2. *Let ℓ be a line segment of length x. Consider a collection $\mathcal{C}$ of $k \geq 1$ lines passing through the origin, at angles $(\pi i)/k$ for $i = 0, \ldots, k-1$. Then, if s is the sum of the lengths of the projections of l on all lines in $\mathcal{C}$, then $x(1 - \Theta(1/k)) \leq s\pi/(2k) \leq x(1 + \Theta(1/k))$.*

Proof. Viewing the question as a continuous one, we can approximate the sum by $x \int_0^{\pi} |\cos(\theta)| d\theta = 2x$. As cos has bounded slope, we have that $|s\pi/k - 2x| = O(x/k)$ (see Figure 1). The lemma follows.

We conclude:

Theorem 2. *For any fixed $\epsilon > 0$ and $\delta > 0$, there is a data structure that processes each update in poly-log time, uses poly-log space, and in poly-log time, given any $c = (c_x, c_y)$, returns an (ϵ, δ)-estimator of $||\boldsymbol{d}||_1$, where $\boldsymbol{d}$ contains the L_2 distances from c to points in S.*

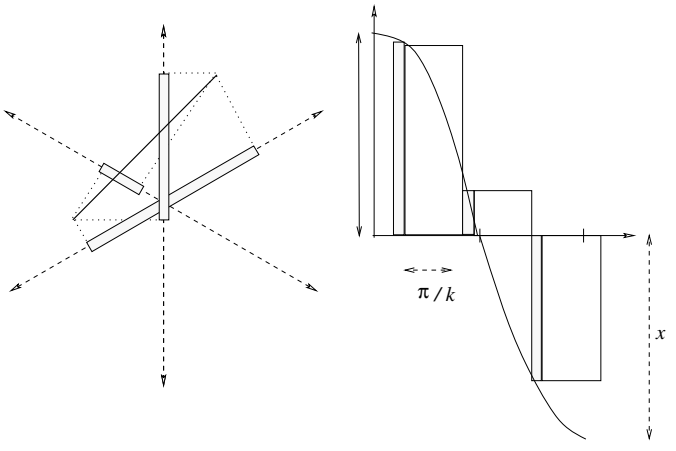


Fig. 1. Approximation of a line by its projections

Proof. For k a sufficiently large constant, we consider a collection of k lines through the origin as above. Let ℓ be a line in this collection, and project all the points (x_i, y_i) onto ℓ. Let t_i be the distance of the projected point from the origin. We use Proposition 1 to maintain a sketch of the vector $(t_1, \ldots, t_n)$; this allows us also to project a given centre c on to ℓ and use the sketch to estimate the total distance $\tau_\ell(c)$ from the projection of the centre. But $\tau_\ell(c)$ is nothing but the sum of the projections of the distance from c to the points; by Lemma 2, we can sum all estimates of $\tau_\ell(c)$ over ℓ, multiply this by $\pi/2k$, and obtain the desired (ϵ, δ) estimator.

4 Conclusions and Open Problems

We have proposed a new model for processing high speed location streams, and presented algorithms for the problem of tracking the "extent", i.e., l_p norm of the distances of the points from a center. For the reset model, the assumption is that the input sequence is *monotone*; we obtain further algorithms for L_p sampling and computing the sum of the k largest elements. All algorithms work under 'streaming' constraints, namely, they use poly-logarithmic space and time.

The problem of processing location streams does not have the immediacy of applications that processing IP network streams has, at this point. But the earliest streaming algorithms were developed around 1996 [8] when the immediacy of IP networking application was not clear. Our model is challenging and we hope it will prove to be a rich source of problems as new applications with location streams become prevalent. Many natural open problems are obvious in the reset models, including the fundamental geometric problems such as estimating convex hulls, etc. Particularly interesting is the formulation of suitable notions of monotonicity that may be needed to make these problems tractable.

Acknowledgements. We thank Graham Cormode for several enlightening discussions, and to Christian Söhler for bringing [29] to our attention.

References

1. DIMACS Workshop on Managing and Processing Data Streams, FCRC (2003), http://www.research.att.com/conf/mpds2003/
2. http://www.interfleet.com/
3. http://www.lbszone.com/, http://www.lbszone.com/, http://www.lbszone.com/
4. http://www.whereismybus.com/
5. http://dimacs.rutgers.edu/Workshops/WGDeliberate/FinalReport5-20-02.doc
6. DIMACS Working Group on Streaming Data Analysis, http://dimacs.rutgers.edu/Workshops/StreamingII/
7. Abounaga, A., Chaudhuri, S.: Self-tuning histograms: Building histograms without looking at the data. In: Proc. SIGMOD (1999)
8. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. In: Proc. ACM STOC, pp. 20–29 (1996)
9. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. ACM PODS, pp. 1–16 (2002)
10. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D., Trevisan, L.: Counting distinct elements in a data stream. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 1–10. Springer, Heidelberg (2002)
11. Charikar, M., O'Callaghan, L., Panigrahy, R.: Better streaming algorithms for clustering problems. ACM STOC (2003)
12. Cormode, G., Datar, M., Indyk, P., Muthukrishnan, S.: Comparing data streams using Hamming norms (How to zero in). IEEE Trans. Knowledge and Data Engineering 15, 529–541 (2003)
13. Cormode, G., Muthukrishnan, S.: Radial Histograms. DIMACS TR 2003-11.
14. Cormode, G., Muthukrishnan, S.: Estimating dominance norms of multiple data streams. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 148–160. Springer, Heidelberg (2003)
15. Datar, M., Muthukrishnan, S.: Estimating Rarity and Similarity over Data Stream Windows. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 323–334. Springer, Heidelberg (2002)
16. Estan, C., Savage, S., Varghese, G.: Automatically inferring patterns of resource consumption in network traffic. SIGCOMM (2003)
17. Feigenbaum, J., Kannan, S., Ziang, J.: Computing diameter in the streaming and sliding window models. Manuscript (2002)
18. Flajolet, P., Martin, G.: Probabilistic counting algorithms for database applications. JCSS 31, 182–209 (1985)
19. Gibbons, P., Matias, Y.: Synopsis data structures. In: Proc. SODA, pp. 909–910 (1999)
20. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: Surfing wavelets on streams: One pass summaries for approximate aggregate queris. VLDB Journal, 79–88 (2001)
21. Gilbert, A.C., Guha, S., Indyk, P., Indyk, P., Kotidis, Y., Muthukrishnan, S., Strauss, M.J.: Fast, small-space algorithms for approximate histogram maintenance. In: Proceedings 34th ACM STOC, pp. 389–398 (2002)

22. Greenwald, M., Khanna, S.: Space-efficient online computation of quantile summaries. In: Proc. ACM SIGMOD (2001)
23. Guha, S., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering data streams. IEEE FOCS, pp. 359–366 (2000)
24. Har-Peled, S., Mazumdar, S.: On Coresets for k-Means and k-Median Clustering. In: Proc. 36th ACM STOC, pp. 291–300 (2004)
25. Henzinger, M., Raghavan, P., Rajagopalan, S.: Computing on data stream. Technical Note 1998-011. Digital systems research center, Palo Alto (May 1998)
26. Hershberger, J., Suri, S.: Convex hulls and related problems on data streams. In: Proc. MPDS (2003)
27. Indyk, P.: Algorithms for dynamic geometric problems over data streams. In: Proc. Annual ACM Symposium on Theory of Computing (STOC), pp. 373–380 (2004)
28. Indyk, P.: Stable distributions, pseudorandom generators, embeddings and data stream computation. IEEE FOCS, pp. 189–197 (2000)
29. Indyk, P., Thorup, M.: Unpublished manuscript (2001)
30. Jana, R., Johnson, T., Muthukrishnan, S., Vitaletti, A.: Location based services in a wireless WAN using cellular digital packet data (CDPD). MobiDE 2001: 74–80
31. Korn, F., Muthukrishnan, S., Srivastava, D.: Reverse nearest neighbor aggregates over data streams. In: Proc. VLDB (2002)
32. Krishnamurthy, B., Sen, S., Zhang, Y., Chen, Y.: Sketch-based change detection: methods, evaluation and applications. In: Proc. Internet Measurement Conference (IMC) (2003)
33. Manku, G., Motwani, R.: Approximate frequency counts over data streams. In: Proc. VLDB, pp. 346–357 (2002)
34. Madden, S., Franklin, M.: Fjording the stream: An architecture for queryies over streaming sensor data. In: Proc. ICDE (2002)
35. Muthukrishnan, S.: Data Streams: Algorithms and Applications. The Foundations and Trends in Theoretical Computer Science series, Now Publishers (2005)
36. Bates, J.: Talk at NAS meeting on Statistics and Massive Data, http://www7.nationalacademies.org/bms/Massive_Data_Workshop.html
37. Querying and mining data streams: you only get one look. Tutorial at SIGMOD (2002) VLDB 2002 etc. See http://www.bell-labs.com/user/minos/tutorial.html
38. Varghese, G.: Detecting packet patterns at high speeds. Tutorial at SIGCOMM (2002)

A Scheduling Problem with One Producer and the Bargaining Counterpart with Two Producers

Xiaobing Gan[1], Yanhong Gu[2], George L. Vairaktarakis[3], Xiaoqiang Cai[4], and Quanle Chen[4,⋆]

[1] Department of Information and System Management, Shenzhen University, Shenzhen 518060, China
[2] Department of Applied Mathematics, Shenzhen University, Shenzhen 518060, China
[3] Department of Operations, Case Western Reserve University, Cleveland, OH 44106-7235, USA
[4] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., HK
smacq@nus.edu.sg, qlchen@se.cuhk.edu.hk

Abstract. First this paper considers a Common Due Window (**CDW**) scheduling problem of n jobs on a single machine to minimize the sum of common weighted earliness and weighted number of tardy jobs when only one manufacturer processes these jobs. Two dynamic algorithms are designed for two cases respectively and each case is proved to be ordinary NP-hard. Successively the scenario, where two manufacturers jointly process these jobs due to the insufficient production facilities or techniques of each party, is investigated. A novel dynamic programming algorithm is proposed to obtain an existing reasonable set of processing utility distributions on the bi-partition of these jobs.

1 Introduction

Since job-processing schedules are important for large-scale manufacturing projects, the development of algorithms for scheduling problems is always attracting many researchers in the domain of optimization. Moreover if the cooperation among several manufacturers for a manufacturing project is considered, bargaining models on partitioning jobs are naturally striking.

In this paper we first address a scheduling problem of n jobs with a **CDW** where the time window is from an earliest due date to a latest due date. It may be convincing that the original research result on due window appears in [1]. Successively, during about twelve years, a main thrust of study in this scheduling area has been directed to the problem type where deterministic combinatorial optimization algorithms can be developed. These include [2]∼[11]. But in recent five years, some effort has been taken to obtain approximate solutions of those more complicated **DW** scheduling problems by stochastic adaptive schemes. The representative papers are [12] and [13]. Such a phenomenon may overwhelmingly

⋆ Corresponding author.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 305–316, 2007.

foresee that it is not proper for us to pursue deterministic algorithms and heuristics to solve tough problems in this field. Then in this paper we focus on one special case of the problem investigated in [4]. We develop some properties to obtain a dynamic programming algorithm for the unrestricted case, where the position of **CDW** is a decision variable and a similar dynamic programming algorithm for the restricted case, in which the position of **CDW** is a given parameter.

Before we introduce the other part of our results in this paper, we should first describe the new discrete bargaining model involving the partitions and schedules of jobs in [26]. Suppose there are two manufacturers who are interested in a manufacturing project with n jobs to be processed. Any of them has not enough production facilities or techniques for this project. Finally they decide to cooperate to bid for this project. Thus before they take part in that bid for the processing of these jobs and win finally, they should first negotiate to partition these n jobs in order to derive a reasonable processing profit distribution accepted by each of these two parties.

Such a cooperation game model on the partition of a beneficial object is first studied by Nash [16]. In his original *Nash Bargaining Model* (**NBM**) Nash supposes that two parties need to divide a beneficial object. When a partition of that beneficial object is given, a unique utility is determined for each party. Thus each party has a utility function on the division of that object. If the Pareto efficient boundary of the set of all the feasible utility pairs is continous, we call it Pareto efficient frontier. If the Pareto efficient frontier are concave, Nash shows that the bargaining solution (utility pair) of his **NBM** is unique, and furthermore it is also the unique solution satisfying four *Nash Axioms*. That is his famous *Nash Bargaining Solution* (**NBS**).

Many subsequent research results on this model can be found in [17]-[25]. The results in [17]-[22] still investigate the situation with a concave Pareto efficient frontier, while the situation only with finite feasible utility pairs, where the Pareto efficient subset possesses some properties similar with the concavity of its continuous counterpart (Pareto efficient frontier), is addressed in [23]-[25]. Although Chen [26] also investigates this discrete situation, he does not assume any convenient property on the set of feasible utility pairs. But Chen assumes the beneficial object is a set of non-preemptive jobs to be processed. The processing time of any job is an integer. A feasible division of this object is a bi-partition of these jobs. All the given parameters of any job such as processing time, due date, weighted penalty factors, and so on, are also non-negative integers. Then naturally chen also lets the two utility function be integer-valued. Besides these general assumptions Chen is concerned with the following special scheduling scenario: when a bi-partition of jobs is given, any party's utility of processing those jobs assigned to him depends on an optimal schedule of those jobs. This optimal schedule minimizes a cost (penalty) function on some operation, inventory or transportation expenditure. Each utility function on jobs assigned to a party comprises two terms and is defined as the first term minus the second. The first term represents the integer gross profit proportional to the total processing times

of jobs assigned to him from a job bi-partition. The second is the minimal value of the objective (cost) function of a scheduling problem. For each of five specific scheduling problems Chen designs a novel dynamic programming algorithm to derive his bargaining solution set initiated for his complicated and practical discrete bargaining model. In section 4 we will concisely expose why Chen needs to revise the unique solution concept highlighted in the NBM to his solution set concept initiated in [26].

In this paper we render a more general gross processing profit structure on the jobs assigned to each party, in which the profit of one processing time unit of different jobs can be different. For the scheduling case addressed in this paper we also propose a dynamic programming algorithm to obtain Chen's solution set.

2 Problem Formulation and Notation

Let $\overline{\mathbf{P}}$ denote the problem studied in [4]. Some basic assumptions about **CDW** are very similar with those of $\overline{\mathbf{P}}$, and other particular notation for our special case should be indicated. These two parts are as follows.

There are n jobs from a customer to be processed on a single machine. All these jobs have been ready when the machine is available, and any preemption is prohibited.
$N = \{1, 2, \cdots, n\}$, the set of these n jobs J_i = job i in N
p_i = processing time of J_i
C_i = completion time of J_i
d_1 = common earliest delivery date for any job in N
d_2 = common latest delivery date for any job in N
$[d_1, d_2]$ = common due window
$d = d_2 - d_1$ = given delivery window length
E = set of jobs in N subject to $C_i < d_1$
W = set of jobs in N such that $C_i \in [d_1, d_2]$
T = set of jobs in N satisfying $C_i > d_2$
$I(x) = \begin{cases} 1, x > 0, \\ 0, x \leq 0. \end{cases}$

In our problem, we also assume that there only exists one common due window, $[d_1, d_2]$, for each job. If J_i is in W, no penalties are incurred. If J_i is in E, it is called an early job, and the producer has to pay the inventory cost, $\alpha(d_1 - C_i)$, where α is defined as the common earliness penalty of each time unit for any job in our situation. It is one simplified constraint against the relevant assumption in [4]. On the other hand, when the completion time of J_i is after d_2, it is defined as a tardy job, and the producer should pay an extra fee, cp_i, in which c is the common cost of one time unit of any job. It is also a simplification of the assumption on the weighted number of a tardy job in [4]. Moreover in this paper we are interested in the following two cases. In the so-called unrestricted case, d_1 or d_2 is a decision variable, and the restricted case refers to a given d_1 or d_2.

Consequently we face two problems, $\mathbf{P_1}$ and $\mathbf{P_2}$.

$\mathbf{P_1}$: $\min\limits_{(\sigma,d_1)} \sum\limits_{i=1}^{n} \alpha(d_1 - C_i)I(d_1 - C_i) + cp_iI(C_i - d_2).$

$\mathbf{P_2}$: $\min\limits_{\sigma} \sum\limits_{i=1}^{n} \alpha(d_1 - C_i)I(d_1 - C_i) + cp_iI(C_i - d_2).$

$\mathbf{P_1}$ is for the unrestricted case, and $\mathbf{P_2}$ is for the restricted case.

Now we present the definitions and notation of the discrete bargaining problems for $\mathbf{P_1}$ and $\mathbf{P_2}$ respectively. For $\mathbf{P_1}$ and $\mathbf{P_2}$ we assume that each party ($\mathbf{A_i}$) has only one machine to process the jobs assigned to him. Those are denoted by $\mathbf{MA_1}$ and $\mathbf{MA_2}$ respectively. D_i denotes $\mathbf{A_i}$'s continuous machine time interval available for this task, and let p_j be the common processing time of J_j for $\mathbf{MA_1}$ and $\mathbf{MA_2}$. The following are the definitions for the utility functions on $\mathbf{P_1}$ and $\mathbf{P_2}$ respectively.

$X = (X_1, X_2)$ = bi-partition of N to $\mathbf{A_1}$ and $\mathbf{A_2}$ such that $X_1 \cap X_2 = \emptyset$ and $X_1 \cup X_2 = N$

$p = \sum\limits_{j=1}^{n} p_j$ = total processing times of these n jobs

$U_i(X_i)$ = integer utility of $\mathbf{A_i}$'s processing the jobs assigned to him
$= \sum\limits_{j \in X_i} b_j - \min\limits_{(\sigma_i, d_1)} f(X_i, \sigma_i, d_1)$ (or $= \sum\limits_{j \in X_i} b_j - \min\limits_{\sigma_i} f(X_i, \sigma_i)$)

where b_j denotes the common processing profit of these two parties for J_j, $f(X_i, \sigma_i, d_1)$ is for $\mathbf{P_1}$, $f(X_i, \sigma_i)$ is for $\mathbf{P_2}$, and $f(X_i, \sigma_i, d_1)$ (or $f(X_i, \sigma_i)$) is $\sum_{j \in X_i} \alpha(d_1 - C_j)I(d_1 - C_j) + cp_iI(C_j - d_2)$.
$e = (e_1, e_2)$ = disagreement point satisfying $e_i \geq U(\emptyset)$, $i = 1, 2$

Actually e_i is defined as the profit obtained from another project by $\mathbf{A_i}$, if these two parties fail to form a union for the project of these n jobs.
$(u_1, u_2) = (U_1(X_1), U_2(X_2))$
$V_i(X_i) = U_i(X_i) - e_i$ = net utility of $\mathbf{A_i}$'s processing the jobs in X_i
$(v_1, v_2) = (V_1(X_1), V_2(X_2))$

According to the analysis in [26], we should treat a series of discrete optimization problems as follows:

$$\begin{aligned} \mathbf{P_1(r)} \quad & \max_X : r_1V_1(X_1)V_2(X_2) + r_2 \min\{V_1^2(X_1), V_2^2(X_2)\} \\ & \text{s.t. } 0 < V_1(X_1) \leq V_2(X_2), \\ & \quad X \text{ is feasible,} \end{aligned}$$

$$\begin{aligned} \mathbf{P_2(r)} \quad & \max_X : r_1V_1(X_1)V_2(X_2) + r_2 \min\{V_1^2(X_1), V_2^2(X_2)\} \\ & \text{s.t. } V_1(X_1) > V_2(X_2) > 0, \\ & \quad X \text{ is feasible,} \end{aligned}$$

where (r_1, r_2) denotes the common weight factor pair for $\mathbf{A_1}$ and $\mathbf{A_2}$ initiated in Chen's bargaining mechanisms, $H = \{1, 2, \cdots, 9\}$, and $r = (r_1, r_2) \in H \times H \cup \{(1, 0), (0, 1)\}$. The initial maximization problem formulated by Nash is max : v_1v_2. The solution of this problem is just the **NBS** of his **NBM**. In

[26] Chen develops one dynamic programming algorithm to solve his $\mathbf{P_i}(\mathbf{r})$ and obtain his original solution set from the solutions of $\mathbf{P_i}(\mathbf{r})$ for each of five specific scheduling cases. In section 4, we will concisely justify the revisions on Nash's formulation and design the algorithms for our scheduling cases.

3 Algorithms for $\mathbf{P_1}$ and $\mathbf{P_2}$

In [4], the NP-hardiness of the studied problem is proved. Here if we revise and add some parameters in that proof as follows,

$$\alpha_i \equiv \alpha = p_{2n+1} = (\sum_{i=1}^{2n} p_i)^3 \ , \ i = 1, 2, \cdots, 2n, 2n+1, \quad c = 1,$$

we can also prove that $\mathbf{P_i}$ is NP-hard; see the details in section 4 of [4].

Property 1. There exists an optimal schedule for $\mathbf{P_1}$ so that the processing of these n jobs begin from time point zero and no machine idle time appears between the zero point and the completion time of the last job. ∎

Property 2. There exists an optimal schedule to $\mathbf{P_1}$, where one job starts (or completes) at d_2. ∎

Property 3. There exists an optimal schedule for $\mathbf{P_1}$, where jobs in E are in LPT order. ∎

Property 4. For $\mathbf{P_1}$, for any given set W and any given set T, the subsequences on these two sets can be arbitrary for any optimal sequence of all the n jobs. ∎

Among these four properties, Property 1,2,4 is very similar with those in [4], and Property 3 simplifies the counterpart in [4] owing to our special assumptions on the problem structure. But the following properties are original for $\mathbf{P_1}$; and furthermore, without generality, from now on we assume that the jobs in N has been numbered in LPT sequence.

Property 5. There exists an optimal schedule, where the processing time of the first completed job, J_m, in W is the longest among those in W (see, figure 1). ∎

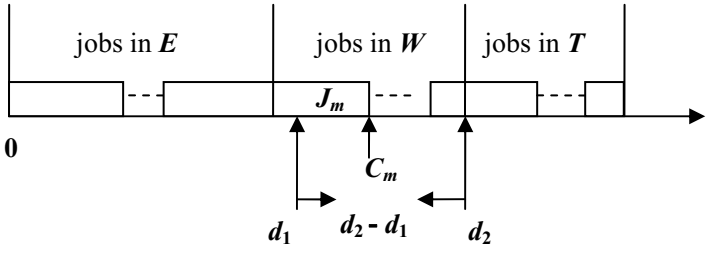


Fig. 1. Illustration of Property 5

Property 6. For any given J_m, there exists a local optimal partition in which
(i) $J_i \in P_m^1 = \{J_i | i > m\} \Rightarrow J_i \in W$ or $J_i \in T$,
(ii) $J_i \in P_m^2 = \{J_i | i < m\} \Rightarrow J_i \in E$ or $J_i \in T$. ∎

This two properties can be proved by the traditional adjacent pairwise interchange method, thereby the details are omitted here. It is not difficult to know that Property 2-6 still hold for $\mathbf{P_2}$, and Property 1 can be revised to the following corollary.

Corollary 1. There exists an optimal schedule for $\mathbf{P_2}$ so that no machine idle time appears between the beginning time point of processing the first job and the completion time of the last job. ∎

Property 7. For $\mathbf{P_1}$ and $\mathbf{P_2}$, there exists such an optimal schedule where the jobs in E and W are in LPT order. ∎

In fact it is a corollary of Property 4-6. In order to design the dynamic programming algorithm of $\mathbf{P_1}$, we should reschedule these n jobs in SPT order. The state variables are k and x. Variable k represents the stage k where the first k jobs in N is used. Variable x refers to the total processing times of the jobs in $E \cup W$ (see Figure 2). $f(k, x)$ is the minimal value of the objective function for a given pair (k, x).

Algorithm 1 (Al1)

Recursive Relationships

$$f(k,x) = \min \begin{cases} f(k-1, x-p_k) \\ +\alpha I(x-p_k-(d_2-d_1))(x-p_k-(d_2-d_1))), \\ f(k-1,x) + cp_k, \end{cases}$$

$2 \le k \le n$, $0 \le x \le p$.

Boundary Conditions

$$f(1,x) = \begin{cases} cp_1, x = 0, \\ 0, x = p_1, \\ +\infty, \text{otherwise}, \end{cases}$$

$0 \le x \le p$;
$f(k,x) = +\infty$, $k \ge 1$, $x < 0$. ∎

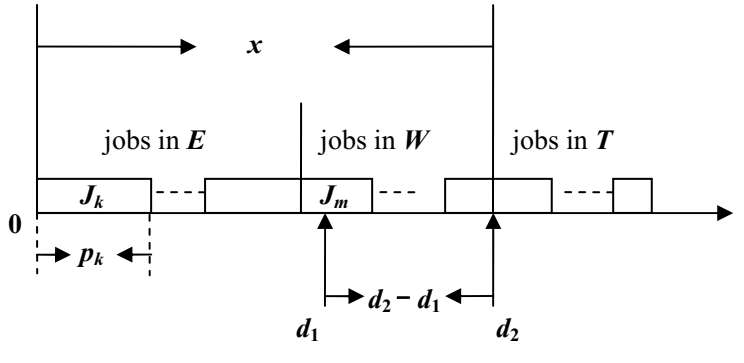


Fig. 2. Illustration of Algorithm 1

It is not difficult to know that (1) the only necessary change is to replace p with d_2 if we consider the restricted case ($\mathbf{P_2}$); (2) the only necessary change is to replace cp_k with β_k if we consider a more general definition on the weighted number for each tardy job, where let β_k denote the common penalty factor for a tardy job J_k. Obviously the complexity of **Al1** is $O(np)$, and that of the revised algorithm for $\mathbf{P_2}$ is $O(nd_2)$.

4 Dynamic Programming Algorithms for $\mathbf{P_i(r)}$

Before we design the dynamic programming algorithms for our scheduling cases on $\mathbf{P_1}$ and $\mathbf{P_2}$, we should first present a concise justification of the key ideas utilized in Chen's bargaining mechanisms in [26] for his discrete bargaining situation. In [26] Chen gives enough computational experiments and some critical properties to expose the challenges confronted when the utility function of each party is integer-valued and the Pareto efficient subset does not possess any similar concave property for the Pareto efficient frontier. A minor trouble is that, in Chen's discrete bargaining model, it is a usual scenario that both (6,9) and (9,6) maximize v_1v_2. Hence it is easy to know, for any given r, not only $\mathbf{P_1(r)}$ but also $\mathbf{P_2(r)}$ is necessary. One main trouble is that maximizing v_1v_2 and maximizing $\min\{v_1^2, v_2^2\}$ is contradictive frequently even if those two parties have the same utility function. For example, in [26] Chen present an instance of a scheduling case, in which $(v_1^*, v_2^*) = (1300, 1350)$ is the optimal utility distribution of $\max : v_1v_2$, but (1310,1333) maximizes $\min\{v_1^2, v_2^2\}$. Actually besides both $\max : v_1v_2$ and $\max : \min\{v_1^2, v_2^2\}$ ensure the Pareto efficiency of the bargaining solutions, the first criterion highlights a more total utility of those two parties, $v_1 + v_2$, in some sense while the second pursues the least absolute difference between v_1 and v_2, i.e. absolutely fair utility allocation. Obviously in the real world which criterion is more important is subjective. Consequently any proper bargaining solution scheme should at least include the solutions of these two maximization problems. Moreover Chen introduces Satty's famous 1-9 rule initiated for the Analytic Hierarchy Processes (AHP) into his bargaining solution concept. If a weight pair (r_1, r_2) for those two criteria is given by those two parties together, it means that they feel the importance of $\max : v_1v_2$ versus $\max : \min\{v_1^2, v_2^2\}$ is $r_1 : r_2$. Thus $\mathbf{P_i}((\mathbf{r_1}, \mathbf{r_2}))$ can be used to offer the solutions needed by those two parties. After some further investigations in [26], Chen finds only an essential part of all the solutions of $\mathbf{P_i(r)}$ is useful to construct his final bargaining solution set which can be utilized to axiomatize his results on his new discrete cooperation game model.

Now we will present the algorithm for our unrestricted scheduling case to solve the corresponding $\mathbf{P_i(r)}$. Let $d_{i2} - d_{i1}$ denote $\mathbf{A_i}$'s given common due window, and assume that $D_1 \leq D_2$. Here we also reschedule these n jobs in SPT order first. The state variables are k, x_1, y_1, y_2, v_1. Variable k represents the stage k where the first k jobs in N are allocated. Variable x_1 is defined as the total processing time of jobs assigned to $\mathbf{A_1}$. Let Variable y_i be the total processing time of jobs which are not tardy on $\mathbf{MA_i}$. v_1 is the net profit earned by $\mathbf{A_1}$'s

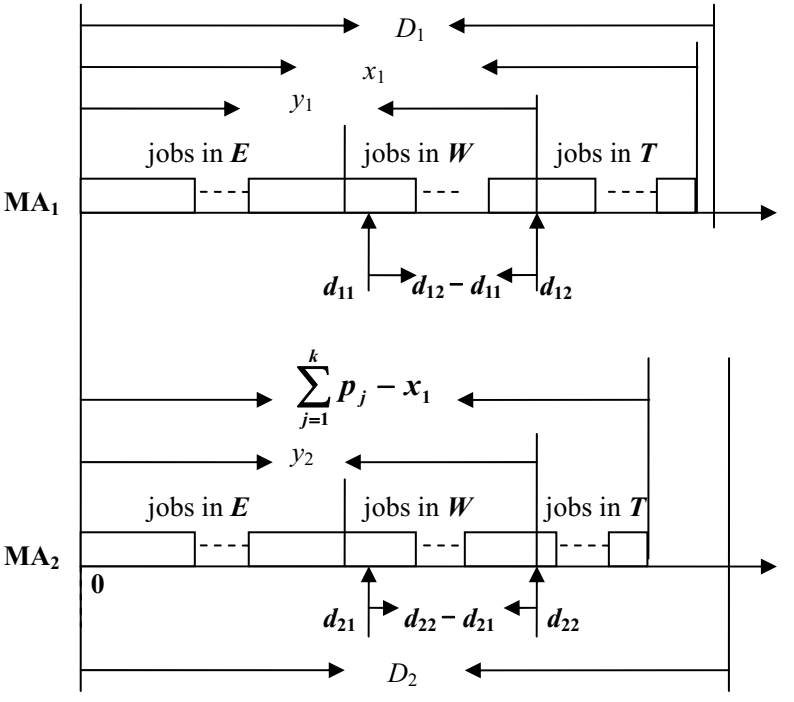


Fig. 3. Illustration of Algorithm 2

processing the jobs assigned to him. $v_1(x_1, y_1, y_2, v_1)$ denotes the maximal profit of $\mathbf{A_2}$ w.r.t. a given vector (x_1, y_1, y_2, v_1).

Algorithm 2 (Al2) (see Figure 3)

Recursive Relationships

$$v_2(k, x_1, y_1, y_2, v_1)$$
$$= \max \begin{cases} v_2(k-1, x_1 - p_k, y_1 - p_k, y_2, v_1 - b_k \\ +\alpha I(y_1 - p_k - (d_{12} - d_{11}))(y_1 - p_k - (d_{12} - d_{11}))), \\ v_2(k-1, x_1 - p_k, y_1, y_2, v_1 - b_k + cp_k), \\ v_2(k-1, x_1, y_1, y_2 - p_k, v_1) + b_k \\ -\alpha I(y_2 - p_k - (d_{22} - d_{21}))(y_2 - p_k - (d_{22} - d_{21})), \\ v_2(k-1, x_1, y_1, y_2, v_1) + b_k - cp_k, \end{cases}$$
$$2 \le k \le n, \max\{\sum_{j=1}^{k} p_j - D_2, 0\} \le x \le \min\{D_1, p\},$$
$$0 \le y_i \le \min\{D_i, p\},$$
$$-\max\{c, (n-1)\alpha\}(p - p_1) - e_1 \le v_1 \le \sum_{j=1}^{n} b_j - e_1.$$

Boundary Conditions

$$v_2(1, x_1, y_1, y_2, v_1)$$
$$= \begin{cases} -e_2, \ x_1 = p_1 \le D_1, \\ \quad y_1 = p_1, y_2 = 0, v_1 = b_1 - e_1, \\ \quad \text{or} \\ \quad x_1 = p_1 \le D_1, \\ \quad y_1 = 0, y_2 = 0, v_1 = b_1 - \beta_1 - e_1, \\ b_1 - e_2, \ x_1 = y_1 = 0, v_1 = -e_1, y_2 = p_1 \le D_2, \\ b_1 - \beta_1 - e_2, \ x_1 = y_1 = 0, v_1 = -e_1, y_2 = 0, p_1 \le D_2, \\ -\infty, \text{otherwise}, \end{cases}$$
$$\max\{\sum_{j=1}^{k} p_j - D_2, 0\} \le x \le \min\{D_1, p\},$$
$$0 \le y_i \le \min\{D_i, p\},$$
$$-\max\{c, (n-1)\alpha\}(p - p_1) - e_1 \le v_1 \le \sum_{j=1}^{n} b_j - e_1;$$
$$v_2(k, x_1, y_1, y_2, v_1) = -\infty, \ k \ge 1,$$
$$x_1 < \max\{\sum_{j=1}^{k} p_j - D_2, 0\},$$
or
$$y_i < 0,$$
or
$$y_i > \min\{D_i, p\},$$
or
$$v_1 < -\max\{c, (n-1)\alpha\}(p - p_1) - e_1,$$
or
$$v_1 > \sum_{j=1}^{n} b_j - e_1.$$ ∎

$$(v_{1r1}^*, v_{1r2}^*) = \arg \max_{0 < v_1 \le v_2(n, x_1, v_1)} \{r_1 v_1 v_2(n, x_1, v_1) + r_2 \min\{v_1^2, v_2^2(n, x_1, v_1)\}$$

is an optimal solution of $\mathbf{P_1}(\mathbf{r})$. The optimal profit distribution structure on $\mathbf{P_2}(\mathbf{r})$ is similar. Obviously the complexity of Algorithm 2 is $O(n(\min\{D_1, p\})^3 (\sum_{j=1}^{n} b_j + \max\{c, (n-1)\alpha\}(p - p_1)))$. Hence this bargaining problem is ordinary NP-hard. Similarly if we make some slight changes on **Al2**, it can be used to treat the restricted case and the cases in which the tardy penalty per time unit of different jobs may be different.

5 Conclusion

In this paper we develop some pseudo-polynomial algorithms for a scheduling problem with a common due window, and furthermore for the situation where two producers jointly process these jobs, we present a dynamic programming

algorithm to derive some reasonable bi-partitions of jobs and the relevant processing distributions and then implement the bargaining mechanisms in [26].

It is interesting to develop heuristics or adaptive algorithms for those more complicated scheduling cases. Topics on this domain has been in our further investigation.

6 Appendix

This appendix is only prepared for referees of this paper (Paper 189). We will present some content according to some referees' comments about our paper. If this revised version is accepted finally, this appendix can be deleted.

In this revised paper, we simplify the old title, modify the section of introduction, and add something detailed on the justification of Chen's bargaining model. Besides it we finally decide to cancel the content on the Algorithm 1 in the old version. The reasons are as follows:

1. There is a strict paper limit.
2. This algorithm is only useful for the case with the more special assumption, cp_i, on the weighted number of a tardy job.
3. For a conference paper with page limit, it may be more important to render readers new models and new ideas. Since the bargaining part may be a promising research domain of discrete optimization on game theory, we want to present those details on the old Algorithm 1, which is a technical breakthrough on traditional scheduling problems, in some future complete paper.
4. The index function $I(\cdot)$, emerging in the Algorithm 1 in this revised version, is also novel for dynamic programming structures.

Finally we will offer an explanation on the proof of NP-hardness of $\mathbf{P_1}$ and $\mathbf{P_2}$.

The partition problem (**PP**)

Given a finite set A of $2n$ integers, $a_1, a_2, \cdots, a_{2n}$, where each $a_i \in Z^+$, does there exist a subset $B \subseteq A$ such that $\sum_{a_i \in B} a_i = \sum_{a_i \in A-B} a_i$?

The corresponding instance of $\mathbf{P_1}$ can be constructed as follows:
number of jobs: $2n+1$;
window size: $d = (\sum_{i=1}^{2n} a_i)/2 = S/2$;
processing times: $p_i = a_i,\ i = 1, 2, \cdots, 2n,\ p_{2n+1} = S^2$;
earliness weights: $\alpha_i \equiv \alpha = p_{2n+1} = S^3,\ i = 1, 2, \cdots, 2n, 2n+1$;
assumption on weighted number of a tardy job: $c = 1$.

Then it is not difficult to know that there exists an optimal sequence, where the processing of J_{2n+1} should be completed on d_1. Hence due to this optimal schedule structure we can implement the polynomial reduction from **PP** to $\mathbf{P_1}$. Since $\mathbf{P_1}$ is NP-hard, so is $\mathbf{P_2}$. For the referees' convenience, we also offer the reference [4] with the file name, liman1994.pdf, when we upload our final version to ESCAPE2007.

References

1. Anger, F.D., Lee, C.-Y., Martin-Vega, L.A.: Single Machine Scheduling with Tight Windows. Research Report, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FloridaTransportation Science, pp. 86–16 (1986)
2. Kramer, F.-J., Lee, C.-Y.: Common Due Window Scheduling. Production and Operations Management 6, 262–275 (1997)
3. Lee, C.-Y.: Earliness-Tardiness Scheduling Problems with Constant Size of Due Date Window. Research Report, Department of Industrial and System Engineering, University of Florida, Gainesville, Florida, pp. 91–17 (1991)
4. Liman, S.D., Ramaswamy, S.: Earliness-Tardiness Scheduling Problems with a Common Delivery Window. Operations Research Letters 15, 195–203 (1994)
5. Liman, S.D., Panwlker, S.S., Thongmee, S.: Determination of Common Due Window Location in a Singer Machine Scheduling Problem. European Journal of Operational Research 93, 68–74 (1996)
6. Koulamas, C.: Single-Machine Scheduling with Time Window and Earliness/Tardiness Penalties. European Journal of Operational Research 91, 190–202 (1996)
7. Ventura, J.A., Weng, M.X.: Single Machine Scheduling with a Common Delivery Window. Journal of the Operational Research Society 47, 424–434 (1996)
8. Koulamas, C.: Maximizing the Weighted Number of On-Time Jobs in Single Machine Scheduling with Time Windows. Math. Comput. Modelling 25, 57–62 (1997)
9. Liman, S.D., Panwalkar, S.S., Thongmee, S.: Common Due Window Size and Location Determination in a Single Machine Scheduling Problem. Journal of the Operational Research Society 49, 1007–1010 (1998)
10. Liman, S.D., Panwalkar, S.S., Thongmee, S.: Scheduling Common Due Window Problems with Controllable Processing Times. Annals of Operations Research 70, 145–154 (1997)
11. Chen, Q.-L., Sun, S.-J.: An Earliness and Tardiness Problem in Single Machine Scheduling wiht a Common Due Window. Applied Mathematics- a Journal of Chinese universities SerA 15, 440–448 (2000)
12. Chen, Z.-L., Lee, C.-Y.: A Column Generation Algorithm for Parallel Machine Common Due Window Scheduling. European Journal of Operational Research 136, 512–527 (2002)
13. Yen, B., Wan, G.: Tabu Search for Total Weighted Earliness and Tardiness Minimizing on Single Machine with Distinct Due Windows. European Journal of Operational Research 142, 271–281 (2002)
14. Baker, K.R., Scudder, G.D.: Sequencing with Earliness and Tardiness Penalties: A Review. Oper. Res. 38, 22–36 (1990)
15. Lawler, E.L.: Fast Approximation Algorithms for Knapsack Problems. In: Proc. 18th Annual Symposium on Foundation of Computer Science, pp. 206–213. IEEE Computer Society, Long Beach, CA (1977)
16. Nash, J.: Two Person Cooperative Games. Econometrica 21, 128–140 (1953)
17. Muthoo, A.: Bargaining Theory with Application. Cambridge University Press, Cambridge (1999)
18. Zhang, D.: A logical Model of Nash Bargaining Solution. In: Proceeding of IJCAI, pp. 983–990 (2005)
19. Trockel, W.: Integrating the Nash Program into Mechanism Theory. Review of Economic Design 7, 27–43 (2002)

20. Trockel, W.: Core-equivalence for the Nash Bargaining Solution. Economic Theory 25, 255–263 (2005)
21. Dagan, N., Volij, O., Winter, E.: A Characterization of the Nash Bargaining Solution. Social Choice and Welfare 19, 811–823 (2002)
22. Touati, C., Altman, E., Galtier, J.: Generalized Nash Bargaining Solution for Bandwidth Allocation. Computer Networks, 2006 (in press)
23. Nagahisa, R., Tanaka, M.: An axiomatization of the Kalai-Smorodinsky Solution when the Feasible Sets can be Finite. Social Choice and Welfare 19, 751–761 (2002)
24. Lahiri, S.: Axiomatic Characterization of the Nash and Kalai-Smorodinsky Solutions for Discrete Bargaining Problems. PU.M.A 14, 207–220 (2004)
25. Mariotti, M.: Nash Bargaining Theory when the Number of Alternatives can be Finite. Social Choice and Welfare 15, 413–421 (1998)
26. Chen, Q.-L.: A New Discrete Bargaining Model on Job Partition Between Two Manufacturers. PhD Dissertation, The Chinese University of Hongkong (2006)

Phrase-Based Statistical Language Modeling from Bilingual Parallel Corpus

Jun Mao, Gang Cheng, and Yanxiang He

Computer School, Wuhan University, Wuhan, China

Abstract. Phrase-based models and class-based models are both variants of classical n-gram models. In this paper, we propose an approach by merging phrase-based models and class-based models together. In the phrase-based part, we use bilingual parallel corpus to extract phrases with a method deriving from phrase-based translation models. Then we partition these phrases into phrase classes by minimizing the loss of the average mutual information with the aid of a count matrix. Our experimental results suggest that phrase-based models can capture more key information than word-based models and class-based models can capture the relationship among similar words or phrases and thus solve the problem of data sparseness in some sense.

1 Introduction

Statistical language models have been widely used in many domains, especially in automatic speech recognition and statistical machine translation. Language models help our outputted sentences to be natural and grammatical with regard to a certain language.

Mathematically, a language model is to estimate the prior probability of word sequences as its task is to predict the next word given previous words. Words are usually taken as the basic linguistic units in standard language models. If the prior probability of some word sequences in a sentence has a high value, we can probably conclude that this sentence is more like a good sentence than low-valued ones according to the criterion of a given language.

Obviously, however, there are a large number of phrases which occur very often and have high frequencies. If we search by Google, results show that 2,230,000,000 pages are found for New York while 2,270,000,000 for York. The word York mostly appears in the phrase New York since P($New\ York|York$) is as high as 98.2%. In this case, such highly frequent phrases can be regarded as single units in language models. Our experimental results show that taking phrases as basic linguistic units can greatly improve the performance of language models. Phrased-based language models are capable of dividing and converging sentences more naturally and capturing more key information.

Sparseness of data is an inherent property of any given real text. It is a big problem that will happen when collecting frequency statistics on word sequences from a corpus of finite size, even very large corpus. When we improve the basic units from words to phrases, the data sparseness problem are getting more

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 317–328, 2007.

serious. Many methods to smooth data have been proposed to provide better estimates of the more infrequent or unseen events, such as discounting and back-off or interpolated estimation. But those methods are limited in the capacity of solving the sparseness problem particularly to phrased-based language models. In this paper, we use class-based models which assign words to classes based on the frequency of their co-occurrence with other words. Experimental results also show that perplexity is notably decreased.

This paper is organized as follows: we review in section 2 the basic language models and related work. In section 3, we introduce our approaches to extract phrases from the bilingual parallel corpus based on Koehn's method. In section 4, we partition phrases into classes with a enhanced method based on Brown's. In section 5, we present the results of our main experiments. Finally, we give in section 6 a conclusion and future work.

2 Language Models

2.1 N-Gram Language Models

Basic language model is to compute the probability of a word sequence or usually a sentence by computing every conditional probability of each word given previous words and then multiplying them together. A sequence of m words is denoted as $w_1 \dots w_m$ or w_1^m. For the joint probability of each word in a sentence, we use $P(w_1, w_2, \dots, w_m)$ or $P(w_1^m)$. Then, using the chain rule of probability, $P(w_1^m)$ can be formulated as follows[1]:

$$P(w_1^m) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)\dots P(w_m|w_1\dots w_{m-1}) \tag{1}$$

However, with limited corpus, it is too difficult to compute the exact probability of any word given a long sequence of preceding words since language is so creative and changeful that there are numerous possible N-grams given the last word.

According to the Markov assumption, this problem can be simplified to n-grams models. N-gram models are word prediction probabilistic models, which predict the next word of previous n-1 words instead of all the histories. Most common n-grams are bigram, trigram, four-gram models. In trigram models, instead of computing P(*departure*|*Rhett Bulter is back from London after a long*), we compute P(*departure*|*a long*). Generally, using n-gram models, the conditional probability of the next word can be computed as follows:

$$P(w_i|w_1^{i-1}) \approx P(w_i|w_{i-n+1}^{i-1}) \tag{2}$$

A simple way to estimate the above equation is called *maximum likelihood estimation*(MLE). We get the MLE estimate for the parameters of an n-gram model by taking the counts from corpus. If *C(w)* is the number of times that w occurs in the corpus, then:

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{C(w_{i-n+1}^{i-1}w_i)}{C(w_{i-n+1}^{i-1})} \tag{3}$$

2.2 Related Work on Language Models

In 1992, IBM's Brown et al. discussed n-gram models based on classes of words[2]. Classed are extracted automatically in such a way that overall perplexity of the corpus is minimized.

Extending the previous work of Goodman in 2001, Microsoft's Jianfeng Gao made an empirical study of clustering for Asian language (Chinese & Japanese) modeling. Clustering is used to reduce the perplexity as well as to compress language models[3]. Instead of the way to get best clusters, they concentrated on making best use of clusters, including predictive clustering, conditional clustering and combined clustering.

From a linguistically tagged corpus, Imed Zitouni et al. proposed a statistical language model based on variable-length sequences[4]. A mutual information threshold and the minimum occurrence of word sequences are predefined at the beginning. Compute all the consecutive couples from the corpus. If the mutual information and the occurrence of a couple are greater than the threshold, this couple can be selected as a candidate of a word sequence. Compute the perplexity on a development corpus and iterate until perplexity stops decreasing.

3 Extract Phrases

3.1 Overview of Our Method

Based on Koehn's method[5], we learn phrase alignments from a parallel corpus that has been word-aligned by Giza++[6]. Then we extract phrases based on the alignment points. In the following of this section we will go into more details of the process of extracting phrases.

3.2 Giza++

Giza++ is a part of the SMT toolkit Egypt which extracts linguistic information from a bilingual corpus, and it is based on IBM models and algorithms described in [2]. Giza++ is usually to output translation tables and some files after training the corpus. Here we are only interested in the word alignment file derived from Giza++.

3.3 Extract Alignment Points

According to IBM models, one word aligns to multiple words, but ideal alignment phrases we need should be multiple words which aligns to multiple words. In order to get many-many alignment words, both an English-Chinese alignment file and a Chinese-English one are generated by bidirectional training. From bidirectional alignment files, we extract alignment points based on a heuristic approach. Our heuristic approach is similar to Koehn's [5], but we extend his approach by removing some deficiencies. Here, the algorithm for our approach:

Algorithm 1

Input `an English-Chinese alignment file`
`a Chinese-English alignment file`

Output `alignment points`

1. Get the intersection of the two alignment files, intersect $(e2c, c2e)$. And Get the union of the two alignment files, union $(e2c, c2e)$.
2. Take intersect $(e2c, c2e)$ as the initial alignment points set.
3. Add neighboring points to the alignment points set: For all the English words and Chinese words, if an English word e aligns to a Chinese word c, $< e, c > \in (e2c, c2e)$, check all the neighbors of $< e, c >$ by starting from the left neighbors then going through all other neighbors clockwise and a neighboring point $< x, y >$ is added to the alignment points set if it satisfies:
   (a) In present alignment points set, there is no alignment point like $< x, * >$ or $< *, y >$. Here, $*$ denotes any word, x is an English word, y is a Chinese word.
   (b) $< x, y >$ is in union $(e2c, c2e)$.
4. Loop step 3 until no new alignment point is found.
5. Add outliers to the alignment points set: For all the English words and Chinese words, check all the outliers which are neither present alignment points nor neighbors of any present alignment point. An outlier $< p, q >$ is added to the alignment points set if it satisfies:
   (a) In present alignment points set, there is no alignment point like $< p, * >$ and $< *, q >$. Likewise, $*$ denotes any word, p is an English word, q is a Chinese word.
   (b) $< p, q >$ is in union $(e2c, c2e)$.
6. Collect all the appropriate alignment points derived from the intersection, the neighboring points set and the outliers set, we obtain the final alignment points set.

3.4 Extract Phrases

In this section, we are going to extract phrases by merging the alignment points. In Koehn's approach, consistent and a BP constraint are defined. For a phrase alignment to be consistent with the word alignment, all alignment points for rows and columns that are touched by the box have to in the box, and not outside.

Based on BP constraint, we present a simple approach to find all the valid phrases. We define *direct neighboring* is neighboring on the left, right, up, down side and define *diagonal neighboring* is neighboring the word on the four diagonal directions. Direct neighboring points are required to appear together in the same phrases. At the beginning, neighboring points are merged into one phrase. For example, b_1 and b_2 are direct neighboring, so b_1 and b_2 can be regarded as one alignment point. The initial phrases set are extracted after all the neighboring points are merged with their direct neighbors. Then we merge all the adjacent

phrases into longer phrases and remove those contradict to BP constraint. Actually in our approach we are not interested in removing invalid phrases but focusing only on those valid ones. Here, the details of our algorithm:

Algorithm 2

Input `Alignment Points`

Output `Phrases`

1. Get initial phrases set by merging direct neighboring points into phrases.
2. Sort all the phrases from the initial phrases set by their sequence in the Chinese sentence(X-axis). Denote these sorted phrases as $\overline{w}_1, \overline{w}_2, \ldots, \overline{w}_n$. For a Chinese phrase $\overline{w}_i$, the x coordinate of the start point is denoted as x_{si}, the end point, x_{ti}. Thus the range of the X coordinate of C_i is $[x_{si}, x_{ti}]$. For all $i \geq j, x_{si} \leq x_{ti}$ and $x_{ti} \leq x_{si}$.
3. Merge phrase $\overline{w}_i$ and $\overline{w}_{i+1}$ as a phrase candidate, the range of X coordinate of the candidate is $[x_{si}, x_{t(i+1)}]$. If C_i aligns to an English phrase E'_i is $[y'_{si}, y'_{ti}]$. So the range of Y coordinate of the candidate is $[y'_{si}, y'_{t(i+1)}]$ or $[y'_{s(i+1)}, y'_{ti}]$.
4. $[y'_{si}, y'_{t(i+1)}]$ or $[y'_{s(i+1)}, y'_{ti}]$ can be denote as $[y_{sm}, y_{tn}]$, $n \geq m$. If $n - m = 1$, then take the phrase candidate as a valid phrase and add it to the phrase set.
5. Reorder present phrases based on their range of X coordinate. Go back to Step2 and iterate until no new phrase candidate can be found.

The final result is that the whole sentence becomes a single long phrase.

4 Extract Classes

4.1 Class-Based Models

Obviously, some phrases are similar syntactically, semantically and functionally. We always say *good morning*, *good afternoon* or *good night*. These three phrases all have a meaning of speakers warm greeting to the listener and they all have a structure of the word *good* plus a noun. Intuitively, we can assign the three phrases into one distinct class. And it is not difficult to be aware that *good evening* also belongs to the class. Suppose that a word w_i can only be uniquely mapped to its own class c_i for the sake of simplicity.

As described in section 1, we use class-based models to resolve the problem of data sparseness. Phrases are more sparsely distributed than words with regard to a large corpus. We also explore the potential of class-based models to minimize perplexity by making local optimizations. Even if class-based models offer no perplexity reduction in comparison to traditional models, it is beneficial to smooth the data via back-off methods since the counts of most linguistic units are increased and classes of high-order grams occur more frequently than original grams in corpora.

4.2 MLE of the Parameters for the Phrase and Class Model

Extended from word-based n-gram models, phrase and class models use the frequency of sequences of classes to produce a more knowledgeable estimate of the probability of sentences. The basic class-based model defines the conditional probability of a word w_i based on its history as the product of the two factors: the probability of a particular word w_k given the class c_k, and the probability of the class c_k given the preceding classes $c_1, c_2, \ldots, c_{k-1}$[2]. For example, in traditional class-based n-gram models, we have

$$P(w_k|w_1^{k-1}) = P(w_k|c_k)P(c_k|c_1^{k-1}) \tag{4}$$

Thus, our phrase and class trigram language model is computed as:

$$\begin{aligned} P(\overline{w}_1^n) &= P(\overline{w}_1)P(\overline{w}_2|\overline{w}_1)P(\overline{w}_3|\overline{w}_1\overline{w}_2)\cdots P(\overline{w}_n|\overline{w}_{n-2}\overline{w}_{n-1}) \\ &= P(\overline{w}_1|c_1)P(c_1)P(\overline{w}_2|c_2)P(c_2|c_1)\cdots P(\overline{w}_n|c_n)P(c_n|c_{n-2}c_{n-1}) \\ &= P(c_1^n)\cdot\prod_{i=1}^{n} P(\overline{w}_i|c_i) \end{aligned} \tag{5}$$

Similar to IBM model, the maximum likelihood estimates of the parameters of a trigram phrase and class model are:

$$P(\overline{w}_i|c_i) = \frac{C(\overline{w}_i)}{C(c_i)}, \tag{6}$$

and

$$P(c_i|c_{i-2}c_{i-1}) = \frac{C(c_{i-2}c_{i-1}c_i)}{C(c_{i-2}c_{i-1})}, \tag{7}$$

and for unigram:

$$P(c_i) = \frac{C(c_i)}{T}, \tag{8}$$

and for bigram:

$$P(c_i|c_{i-1}) = \frac{C(c_{i-1}c_i)}{C(c_{i-1})}, \tag{9}$$

Combine estimates (6)$\sim$(9), we obtain:

$$P(\overline{w}_1^n) = \frac{1}{T}\cdot\prod_{i=1}^{n}\frac{C(\overline{w}_i)}{C(c_i)}\cdot\frac{\prod_{i=3}^{n}C(c_{i-2}c_{i-1}c_i)}{\prod_{i=3}^{n-1}C(c_{i-1}c_i)}. \tag{10}$$

4.3 Basic Algorithm

The algorithm is based on the idea that phrases are partitioned into classes in such a way that the overall perplexity of the corpus is minimized.Brown[2]made such a conclusion in the following:

$$L(\pi) = \Sigma_w Pr(w)logPr(w) + \Sigma_{c_1c_2}Pr(c_1c_2)log\frac{Pr(c_2|c_1)}{Pr(c_2)} = -H(w) + I(c_1, c_2) \tag{11}$$

Here, $L(\pi)$ is the negative of perplexity. $H(w)$ is the entropy of the 1-gram word distribution. It is a constant given a corpus. $I(c_1, c_2)$ is the average mutual information of adjacent classes. Because $L(\pi)$ depends only on π the average mutual information, the partition that maximizes $L(\pi)$ is the partition that maximizes the average mutual information of adjacent classes.

Algorithm 3

Input `Phrases`

Output `Phrase Classes`

1. List all the phrases in an order of frequency with the most frequent phrases first.
2. Assign each of the first K phrases to its own as distinct class.
3. Assign the $(K+1)^{st}$ most probable phrase to a new class and merge that pair among the resulting $K+1$ classes for which the loss in average mutual information is minimal.
4. Do step 3 iterative for $V-K$ times until each of the phrases in the phrase list will have been assigned to one of K classes. Here, V is the total number of phrases.

4.4 Classifying Phrases Based on Count Matrix

The basic algorithm of classifying phrases is required to find the minimal loss in average mutual information. In this subsection, we will go further to discuss the approach based on a count matrix.

We are going to construct a $(K+1)*(K+1)$ matrix to store all the counts of neighboring phrase classes occurring in the corpus. The count matrix is denoted as $A(i,j)$. Then we have

$$A(i,j) = \sum_{a \in c(i), b \in c(j)} count(ab) \tag{12}$$

Table 1. Table of count matrix

A(1,1)	A(1,2)	· · · · · · ·	A(1,K+1)
A(2,1)	A(2,2)	· · · · · · ·	A(2,K+1)
·	·	· · · · · · ·	·
·	·	· · · · · · ·	·
·	·	· · · · · · ·	·
·	·	· · · · · · ·	·
·	·	· · · · · · ·	·
A(K+1,1)	A(K+1,2)	· · · · · · ·	A(K+1,K+1)

Based on some equations to compute the average mutual information in [2], we propose our detailed approach to compute average mutual information using count matrix. Suppose at the beginning every single phrase belongs to a distinct class and thus there are V distinct classes. Then each time we merge two classes into one and we get K classes after $V-K$ merges. Now we are going to investigate the merge $C_k(i)$ with $C_k(j)$ for $1 \leq i \leq j \leq k$. Let $p_k(l,m) = P_r(C_k(l), C_k(m))$, i.e. ,the probability that a word in class $C_k(m)$ follows a word in class $C_k(l)$. We also introduce $Pl_k(l)$, $Pr_k(m)$. Then $P_k(l,m)$, $Pl_k(l)$, $Pr_k(m)$ can be computed as follows:

$$P_k(l,m) = P_r(C_k(l), C_k(m)) = \frac{A(l,m)}{\sum_{1\leq i\leq K+1, 1\leq j\leq K+1} A(i,j)} \tag{13}$$

$$Pl_k(l) = \sum_m P_k(l,m) = \frac{\sum_{1\leq m\leq K+1} A(l,m)}{\sum_{1\leq i\leq K+1, 1\leq j\leq K+1} A(i,j)} \tag{14}$$

$$Pr_k(m) = \sum_l P_k(l,m) = \frac{\sum_{1\leq l\leq K+1} A(l,m)}{\sum_{1\leq i\leq K+1, 1\leq j\leq K+1} A(i,j)} \tag{15}$$

The average mutual information remaining after $V-K$ merges is

$$I_k = \sum_{l,m} q_k(l,m), \tag{16}$$

and

$$q_k(l,m) = P_k(l,m) \log \frac{P_k(l,m)}{Pl_k(l) \times Pr_k(m)}. \tag{17}$$

Then the average mutual information remaining after we merge $C_k(i)$ and $C_k(j)$ is

$$\begin{aligned} I_k(i,j) = I_k &- s_k(i) - s_k(j) + q_k(i,j) + q_k(j,i) + q_k(i+j, i+j) \\ &+ \sum_{l\neq i,j} q_k(l, i+j) + \sum_{m\neq i,j} q_k(i+j, m) \end{aligned} \tag{18}$$

where

$$s_k(i) = \sum_l q_k(l,i) + \sum_m q_k(i,m) - q_k(i,i) \tag{19}$$

According to the equations above, to minimize the loss of the average mutual information after mergence is to find the least $I_k - I_k(i,j)$. In this way, the two candidate classes $i, j (i < j)$ to be merged are eventually found. Then we wish to update the count matrix by adding the new class denoted as $(i+j)$ and remove the candidate classes i, j.

Firstly, count matrix A can be modified by adding row j to row i and adding column j to column i,
$A(i,m) := A(i,m) + A(j,m)$ $(1 \leq m \leq K+1)$
$A(l,i) := A(l,i) + A(l,j)$ $(1 \leq l \leq K+1)$

That is, we use the space of storing i to store the new class $(i + j)$, and then add a new candidate class into the space of previous class j. The new candidate class is the present $(K + 1)^{st}$ phrase after recent mergence. After iterations, the final K classes can be obtained.

5 Experiment

5.1 Data Description

We have built an English-Chinese parallel corpus to make experiments on our language model. The corpora contain 221,020 sentences both in English and Chinese, covering English textbooks, news from several websites, laws of China, movie scripts, novels and etc. Our corpora are balanced and representative. In our experiments, sentences of length 5-25 were reserved for testing while others are cut-off.

5.2 Data Smooth

In order to go further with resolving the problem of data sparseness and get better experimental results, we also compute the Turing-discounting and backing-off coefficients of n-grams[7]. In our experiment, we focus on trigrams, bigrams and unigrams because they are most frequently implemented whether in speech recognition systems or in machine translation systems. Moreover, out-of- vocabulary(OOV) words which appear in a given sentence may cause errors[8]. Then the problem of computing the probabilities of OOVs is solved in the unigram part. Results are shown in 5.3.

5.3 Results of Probabilities and Backing-Off Coefficients

In our experiment, we input prepared corpora and output 2 table files, one table of English language model parameters, and one table of Chinese language model parameters. Both the tables are represented as $< n\text{-}gram, Probability, \alpha >$. Here, column *n-gram* represents the *n-gram* word sequences. The results of computing *n-grams* probabilities are shown in the column probability. And column *alpha* includes the backing-off coefficients of all bigrams and unigrams. All the probabilities are stored in logarithm to simplify the computation.

Only 3-grams, 2-grams and 1-grams are considered in our experiment. For the OOVs, $P(w_1) = 0.0315601$ for the English experiment and $P(w_1) = 0.0408239$ for the Chinese experiment.

5.4 Results of Phrase Classes

It is not easy to predict the value of C although there are some off-the-shelf methods to predict C automatically. As it is not what we concern in this paper, we empirically take $C = 2,000$ in our experiment, similarly to Brown's experiment.

Table 2. Some examples of results on English corpus

n-gram	Probability	α
offensive	-5.122058	-0.1836201
room	-3.177584	-0.8335384
preservation	-5.343901	-0.241665
as dangerous	-2.766028	-0.5649664
Sunday afternoon	-1.567137	-0.0663463
Because of	-0.9120448	-0.4944892
sick and tired	-0.6643708	
get angry with	-1.062311	
talking angrily to	-0.6643708	
had any breakfast	-1.363341	

Table 3. Some examples of phrase classes

Mr. John ||| Ms. Julia
there's ||| that is
indifficulty ||| indebt
The Morgan factory ||| steel mill
the Michigan ||| New York City ||| Washington DC
Song writers ||| singer star ||| movie star ||| football fans
thirteen feet ||| the 14th floor ||| ten kilometres
are discussing ||| was listening to ||| 's in London ||| am reading a
South Korea ||| People's Republic of China ||| United States ||| United Nation ||| Middle East
tell anyone ||| tell you ||| tell you ||| informed him ||| let him know
Chief Executive Officer ||| Chief Information Officer ||| General Manager ||| Marketing Director
The year ||| What year ||| Which department do ||| What sort of ||| Which station ||| Which flight

5.5 Perplexity

Perplexity is the most common value to evaluate the performance language models. It is defined as 2^H where

$$H = -\frac{1}{L-n+1}\sum_{i=n}^{i=L}\log_2 P(w_i|w_{i-n+1}^{i-1}) \tag{20}$$

Here, $w_1, \ldots, w_L$ is the test word sequence and $n = 2, 3$ for a 2-gram and 3-gram model. The following is the test perplexity of different language models.

Results show that the introduction of phrases and classes has greatly improved the performance of language models. The introduction of phrases alone decreases the perplexity of more than 20 percent. We also noticed that the introduction of classes has obviously reduce the perplexity by nearly 2 more points than using phrase-based model alone. Moreover, trigram models outperform bigram models by approximate 14 points. The results prove that higher-order n-gram models are better than lower ones. However, it is very difficult to computer n-grams($n \geq 4$) for severe data sparseness problem.

Table 4. Results of perplexity with different language models

No.	Type of LM	Perplexity	Improvement	Compared with
1.	Bigram	136.72		
2.	Phrase-based Bigram	99.53	27.2%	1
3.	Phrase&Class based Bigram	96.29	29.6%	1
4.	Trigram	118.01	13.7%	1
5.	Phrase-based Trigram	92.50	21.6%	4
6.	Phrase&Class based Trigram	90.37	23.4%	4

6 Conclusion and Future Work

We have presented in this paper a novel approach to improve the performance of traditional n-gram language models by introducing phrases and classes as the basic elements of language models. From our English-Chinese parallel corpus, we extract phrases by merging the alignment points deriving from the alignment template which is first introduced in statistical machine translation. We propose our own algorithm to extract phrases which is much easier than checking BP constraints all the time. When we discuss on class-based models, we follow the traditional idea of minimizing the loss of average mutual information and propose a simplified approach to classify phrases based on a count matrix.

We have compared our phrase and class based approach with basic n-gram model. Experimental results show that we have gained an improvement by reducing perplexity more than 20 percent both in bigram model and trigram model. Phrase and class based trigram model shows the best performance in our experiment.

Actually, there are many good methods of classification or clustering. In the future work, we are going to explore these methods and compare them with the one presented in this paper. And we will focus on investigating other approaches to extract phrases not only from bilingual corpora but large monolingual corpora.

References

1. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An introduction to speech recognition, computational linguistics and natural language processing. Prentice-Hall, Englewood Cliffs (2006)
2. Brown, P.F., DellaPietra, V.J., deSouza, P.V., Lai, J.C., Mercer, R.L.: Class-based n-gram models of natural language. Computational Linguistics 18(4), 467–479 (1992)
3. Gao, J., Goodman, J.T., Miao, J.: The Use of Clustering Techniques for Language Modeling - Application to Asian Language. Computational Linguistics and Chinese Language Processing 6(1) (2001)
4. Zitouni, I., Smaili, K., Haton, J.-P.: Statistical Language Modeling Based on Variable-Length Sequences. Computer Speech and Language. 17, 27–41 (2003)

5. Koehn, P., Och, F., Marcu, D.: Statistical Phrase-Based Translation. In: Proceedings of. the Conference on Human Language Technology, pp. 127–133 (2003)
6. Och, F.J., Ney, H.: A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics 29(1), 19–51 (2003)
7. Katz, S.M.: Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. IEEE Transactions on Acoustics, Speech and Signal Processing ASSP-35, 400–401 (1987)
8. Clakson, P., Rosenfeld, R.: Statistical Language Modeling using the CMU-Cambridge Toolkit. In: Proceedings of Euro-speech, vol 5 (1997)

Optimal Commodity Distribution for a Vehicle with Fixed Capacity Under Vendor Managed Inventory*

Xiaolin Xu[1,2,**], Xiaoqiang Cai[2], Chunlin Liu[1], and Chikit To[2]

[1] School of Business, Nanjing University, Nanjing 210093, China
xlxu@se.cuhk.edu.hk
[2] Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, N.T., HK

Abstract. Under vendor managed inventory (VMI), vendors have the obligation to maintain the retail stores they serve with sufficient inventory levels. In this paper, specifically, we consider an oil tanker which visits and replenishes multiple retail stations daily. However, due to the fixed capacity of vehicle, at each station the vendor faces such a trade-off, replenishing the current station to a higher level or reserving more capacity for the left stations that may take more profits in future. We address this problem by two mathematical models, corresponding to situations with deterministic and stochastic station demand respectively. A greedy algorithm is developed and proved optimal for the case with deterministic demand. For the case with stochastic demand, we find the optimal replenishment policy by dynamic programming, which selects the median among three critical numbers. Numerical examples are also presented, which shed light on managerial insights partially.

1 Introduction

In typical business model, each party involved in the supply chain manages his own inventory independently, resulting in inefficient inventory control due to factors like bullwhip effect. However, nowadays a new business model, Vendor Managed Inventory (VMI), has been adopted in many industries, under which the upstream distributor has full control on replenishing inventory of the downstream retailers. It is being widely used in many areas and especially found successful in franchising business like "Seven-eleven", "Circle-K" and "ParknShop", a retailer chain store in Hong Kong. Under VMI, as inventory is managed by a central supply chain department, the vendor can maintain lower total inventories and provide better service levels to the customers due to economic of scale and complete information collection among different retailers.

Our work is partially motivated by such an industrial practice: In some region, a gasoline manufacturer serves multiple retail stations by a tanker daily. In the

* Research supported by Natural Science Foundation of China (70671054) and the Research Grant Council of Hong Kong under Project no. CUHK442/05.
** Corresponding author.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 329–339, 2007.

morning, say 8:am, the tanker driver will receive a list of stations to be visited, with delivery route fixed. Upon arriving at each station, he needs to decide how much inventory should be replenished to it. However, due to the limited capacity of the tanker and different profit margins at different stations, the driver faces the trade-off between replenishing the current station to a higher inventory level and reserving more capacity that may take higher profit later.

Corresponding to different realistic situations, we use two mathematical models to address this problem. In the first model, we assume the demand at each station is deterministic and predictable. It corresponds to such a situation, the demand rate at each station is stationary and the transportation time between any two adjacent stations on the route is stable. We develop a greedy algorithm to find the optimal solution. In the second one, however, we consider the stochastic demand at each station, a more practical situation in reality. We formulate this model by dynamic programming, taking into account the demand information updating at each station. Optimal distribution policies are then derived.

The problem studied here is mainly related to inventory routing problem, which has been studied since 70s. For example, Beltrami and Dobin [5] modelled a network and routing problem for waste collection. As stated by Campbell et al. [6], there are three main issues in the inventory and routing problem: 1) when to serve; 2) how much to deliver; and 3) which route to use. In our problem, we focus on the second one, i.e., how much capacity should be allocated to each station on the delivery route.

There are two streams in the inventory routing literature, based on the type of demand (deterministic or stochastic). For the case with deterministic demand, Bell et al. [4] adopted integer programming to solve an inventory routing problem occurred at air products and chemicals. Christiansen ([7], [8], and [9]) modelled shipment planning using mixed integer programming techniques with deterministic parameters.

Stochastic inventory routing problems have gained increasing attention in recent years, due to their practical importance. For a complete survey, see Anton et al. [1]. Barnes and Bassok [2] studied the single depot multi-retailer system with stochastic demands, linear inventory costs, and backlogging cost over infinite horizon and used simulation to analyze the effectiveness of the proposed strategy. Singer et al. [11] described an application of a vehicle routing model with random demands for the distribution of liquefied petroleum gas. However, neither of the above two papers considered demand information updating, hence the replenishment decisions are made before the actual delivery arrival.

The most closely related to our model is Berman and Larson [10], which considered a gas tanker visiting n customers daily and solved the problem by dynamic programming. They considered a linear cost function, a key difference from ours, adopting a piecewise linear cost function. Bassok [3] also considered a problem similar to ours. In their model, the realized demand at each station is an exact number, under which a penalty cost will be charged. Contrast to their setting, the demand at each station in our model is actually a range between the safety level and capacity of each station. If the replenishing quantity is lower

than the safety level, it will incur a penalty cost. Under their setting, they proved the optimal policy is of threshold, however, under ours, the optimal policy is a median among three critical numbers.

The remainder of the paper is organized as follows. In Section 2, we introduce notations and the basic model of our problem. A greedy algorithm for the deterministic demand problem is developed and proved optimal in Section 3. Subsequently, we discuss the problem with stochastic demand in Section 4. Two different settings for the problem are investigated. Some numerical examples are provided in Section 5. Finally, we summarize and suggest some possible extensions in Section 6.

2 Modelling and Notations

Our problem can be described as follows. A tanker with limited capacity K^0 is assigned to deliver oil from the depot to n stations in a chain, with the delivery route fixed. The initial inventory level at station i, say l_i^0, is known before the vehicle leaves the depot. The vehicle visits all the stations sequentially. The oil vendor has a long term contract with the retail station i specifying the unit wholesale price γ_i per unit kiloliter of oil, which is not necessarily the same for all stations. At each station, if the inventory level is lower than a particular threshold when the vehicle departs, the retailer will face a safety problem. On the other hand, each station has its own capacity limit. Let a_i and b_i denote the safety level and capacity respectively, for station $i = 1, ..., n$. If the tanker cannot fill station i to its safety level a_i, a unit penalty α_i will be charged for each unmet unit.

The delivery tour can be regarded as a sequence of decision processes. Before leaving the depot, the initial information about the station inventory levels (i.e., $l_i^0, i = 1, ..., n$) are known to the vendor. However, before the delivery arrival of the vehicle, the inventory level is continuous decreasing because of external customers' demand. After the vehicle's arrival, the actual inventory level at station i is changed to l_i, the difference to the initial level is denoted by I_i. Upon arriving at a particular station, say station i, the driver/vendor needs to decide a replenishing quantity x_i to it. Considering the limited capacity left in the tanker, if he delivers too much to the current station, the driver faces the risk of not having enough oil to replenish the left stations which may make more revenue. However, if he delivers too little, there may be oil leftover with a low unit salvage value δ. We summarize all the notations in Table 1.

In order to preclude some trivial cases, we make the following assumptions.

Assumption 1. *The unit salvage value is smaller than contract sales price at each station, i.e., $\delta < \gamma_i$, for $i = 1, \ldots, n$.*

Assumption 2. *The vehicle capacity is no larger than the sum of all stations' capacities, i.e., $K^0 \leq \sum_{i=1}^{n} b_i$.*

Assumption 1 precludes such a case, if the unit salvage value is larger than the sales price at a particular station, say i, the replenishing amount at this

Table 1. Notations

Type	Symbol	Description
Parameters	δ	the salvage value per unit
	α_i	the penalty cost per unit of unmet demand
	γ_i	the unit sales price
	K^0	vehicle capacity
	l_i^0	initial inventory in station i before vehicle leaves the depot
	l_i	inventory of station i when the vehicle arrives
	I_i	demand in station i before the vehicle arrival, i.e. $I_i = l_i^0 - l_i$
	b_i	capacity of station i
	a_i	safety level of station i
Decision variable	x_i	Replenishment amount for station i

station must be zero since reserving the capacity could take higher revenue. If Assumption 2 is broken, i.e., the vehicle capacity is larger than the total capacity of all stations, the optimal policy is very trivial, just to replenish each station to its capacity limit no matter what its current inventory level is.

At station i, if the inventory level is l_i upon the vehicle's arrival and the replenishing amount is x_i, the profit earned by the vendor is

$$\gamma_i x_i - \alpha_i (a_i - l_i - x_i)^+,$$

which is a piecewise linear function with respect to x_i. We use Figure 1 to depict the profit made in station i against the inventory filled in that station. According to Figure 1, there is a turning point when the replenishing amount x_i filled in station i is equal to $a_i - l_i$, while below which the marginal profit is $\alpha_i + \gamma_i$, and above which, the marginal profit is γ_i.

Our objective is to maximize the total net profit that can be earned from all stations, which is formulated as follows,

$$\max_{x_i} \sum_{i=1}^{n} \{\gamma_i x_i - \alpha_i (a_i - x_i - l_i^0)^+\} + \delta(K^0 - \sum_{i=1}^{n} x_i)$$

subject to

$$\sum_{i=1}^{n} x_i \leq K^0, \tag{1}$$

$$x_i + l_i \leq b_i, \quad \text{for } i = 1, \ldots, n, \tag{2}$$

$$x_i \in Z^+, \text{ for } i = 1, \ldots, n, \tag{3}$$

where $l_i = l_i^0 + I_i, i = 1, \ldots, n$. There are three constraints in the problem: (1) the capacity constraint of the vehicle, i.e., the total inventory allocated to all stations should not be larger than the vehicle capacity; (2) the capacity constraint at

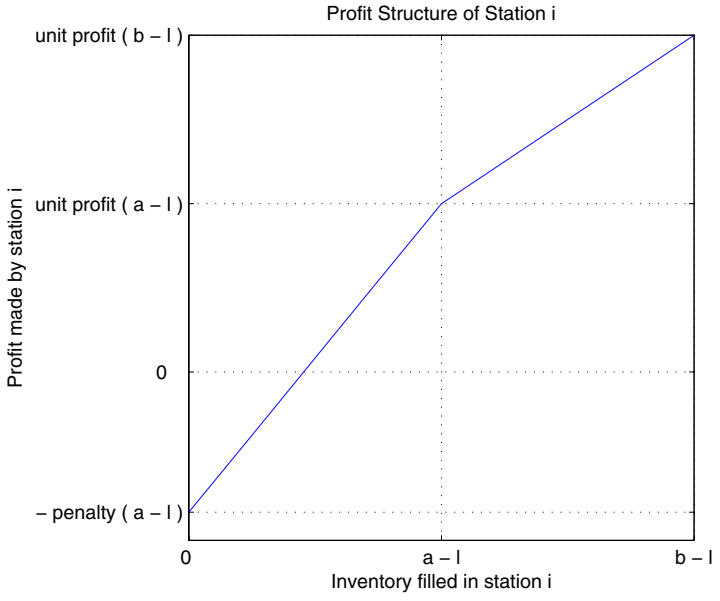


Fig. 1. Profit against inventory filled in station i

each station, i.e., the inventory level after replenishment at each station cannot exceed its capacity limit; and (3) the decision variables $x_i, i = 1, \ldots, n$, are positive integers, which can be easily relaxed to continuous case.

3 Stations with Deterministic Demand

As mentioned in the previous section, the initial inventory levels of all stations $l_i^0, i = 1, \ldots, n$ is known before the vehicle leaves the depot. However, when the vehicle travels from the depot to the stations, it will take time on the road. During this time, the inventory in each stations depletes. In this section, we assume the depletion/consuming rate is stationary. As a result, the inventory level of station i is completely predictable and known. We refer to this as the deterministic demand problem. Then, the inventory level of station i upon the vehicle's arrival is $l_i = l_i^0 - I_i$, where I_i is deterministic.

In this model, since the actual inventory level is completely predictable before the vehicle leaves the depot. The optimal decision remains same at each station, whether it is made before or after the arrival of the vehicle.

We sort all the parameters $\alpha_i + \gamma_i$ and γ_i, $i = 1, \ldots, n$ in a descend order, which can then be further classified into sets $\{J_i^u, J_i^l\}_{i=1}^n$, where J_i^u includes a consecutive chain of parameter(s) with form $\alpha+\gamma$, and J_i^l includes a consecutive chain of parameter(s) with form γ.

We use the following Greedy Algorithm to find a solution to the problem described above.

Greedy Algorithm

step 1. Let $J_i^u = \alpha_i + \gamma_i$ and $J_i^l = \gamma_i$, $i = 1, \ldots, n$; Sort the set $\{J_i^u, J_i^l\}_{i=1}^n$ in a descend order; Let $h = 1$;
step 2. Fill the stations associated with J_h^u up to safety level a_i and the stations associated with J_h^l up to capacity limit, according to the order sorted in step (1). If the oil contained in the vehicle is used up amid one of the stations, terminate; $h := h + 1$;
step 3. If $h \leq 2n$, go to step 2; otherwise, terminate.

Theorem 1. *The complexity of the Greedy Algorithm is $O(n)$. Denote the solution found by Greedy Algorithm as $\{\overline{x}_i\}_{i=1}^n$, it is optimal.*

Corollary 1. *If $\max\{\gamma_k\} \leq \min\{\alpha_h + \gamma_h\}$, then*

(a) If there exists station i with $x_i > a_i - l_i$, then for all station $j \neq i$ $x_j \geq a_j - l_j$;
(b) Assume at station j, $l_j < a_j$, then
(b.1) if $x_j < a_j - l_j$,
$\gamma_j + \alpha_j > \gamma_i + \alpha_i$, then $x_i = 0$;
$\gamma_j + \alpha_j < \gamma_i + \alpha_i$, then $l_i + x_i = a_i$.
(b.2) if $b_j - l_j > x_j > a_j - l_j$,
$\gamma_j > \gamma_i$, then $x_i + l_i = a_i$;
$\gamma_j < \gamma_i$, then $x_i + l_i = b_i$.

Next, we illustrate how the Greedy Algorithm works through a simple example. Assume that we have three stations to serve. The detailed data of parameters are specified in the following Table 2.

Table 2. Parameters of the example

Parameters	station 1	station 2	station 3
γ_i	50	60	70
α_i	40	20	15
$\gamma_i + \alpha_i$	90	80	85
a_i	30	30	30
b_i	60	60	60

According to the above Greedy Algorithm, at the first step, we sort the stations in descend order of $\alpha_i + \gamma_i$, associated with which is the sequence 1, 3, 2 of stations. Hence, we first fill station 1 to its safety level and then stations 3 and 2. After all stations are filled up to the safety level, if there is still inventory available, we will fill the station in descend order of γ_i. Therefore we will have another sequence 3, 2 1 for the stations. If the vehicle capacity is 100 units of oil, then the optimal solution is assign 30 units to station 1, 30 units to station 2, and 40 units to station 3.

Although it is not practical to assume deterministic demand in the real world, this model can provide some insights in the priority of replenishing stations.

Intuitively, one may think that having higher sales price will have the higher priority in filling inventory. However, based on the Greedy algorithm, we can see that there are 2 criteria in selecting the station to be replenished first. The first one is the order of $\gamma_i + \alpha_i$. Once a station is filled to the safety level, its priority will depend on the order of γ_i. This provides a rough idea of the importance of a specific station.

4 Stations with Stochastic Demand

In the section, we assume that the actual inventory level of each station is unknown until after the vehicle arrives at it. We assume that the vehicle will have preliminary information about the initial inventory level and the distribution of demand of each station. Let ϕ_i, Φ_i be the probability density function and the cumulative distribution function of I_i, respectively.

Before proceeding further, we first present the following lemma to be used later.

Lemma 1. *Define $\Delta f(x) = f(x) - f(x-1)$. Let f, g be increasing concave function and s be a constant. For the following optimization problem,*

$$\max_{r \le x \le s} g(x) + f(s-1-x),$$

its optimal solution x^* satisfies the following:

(a) $x^* = s$ if $\Delta g(s) - \Delta f(0) > 0$,
(b) $x^* = r$ if $\Delta g(r) - \Delta f(s-r) < 0$,
(c) $\Delta g(x^*) = \Delta f(s - x^*)\}$, otherwise.

4.1 Decision Made Before Delivery Arrival

In order to evaluate the value of information updating on demand, in this subsection, we assume that the vehicle will make its replenishment decision before the realization of station demand.

Let $\hat{V}_i(K)$ be the expected profit completing a tour from station i and after, given the current capacity of vehicle is K units. We formulate the optimal dynamic equation below:

$$\hat{V}_i(K) = \max_{0 \le x \le K} \{E_{l_i}[\gamma_i \min(x, b_i - l_i) - \alpha_i(a_i - l_i - x)^+] + \hat{V}_{i+1}(K - x)\},$$

with terminal function

$$\hat{V}_{n+1}(K) = \delta K.$$

Lemma 2. *Let $g_i(x) = E_{l_i}[\gamma_i \min(x, b_i - l_i) - \alpha_i(a_i - l_i - x)^+]$, it is increasing and concave.*

Theorem 2. *$\hat{V}_i(K)$ is increasing concave for all $i = 1, \ldots, n$, the optimal value of which is obtained at*

$$x_i^* = \begin{cases} 0, \ \textit{if } \Delta\hat{V}_{i+1}(1) > \gamma_i\Phi(b_i - 1) + \alpha_i\Phi(a_i - 1); \\ K, \ \textit{if } \Delta\hat{V}_{i+1}(K) < \gamma_i\Phi(b_i - K) + \alpha_i\Phi(a_i - K); \\ max\{x \in \{0, 1, \ldots, K\} | \Delta\hat{V}_{i+1}(x) \geq \gamma_i\Phi(b_i - x) + \alpha_i\Phi(a_i - x)\}, \ \textit{o.w.}. \end{cases}$$

In this model, we don't make use of the latest information about the station inventory status. As a result, we risk that our decision for a station i exceeds the available capacity at that station and for potential profits that can be earned by that excess amount. Apparently, the decision made is less optimal, even much worse, than that made after the vehicle's arrival.

4.2 Decision Made After Delivery Arrival

In this subsection, we assume that the decision is made after the vehicle arrives at each station. Consequently, the updated information about the inventory level of current station is available to the vendor before making his decision. In addition, the travel time between any two adjacent stations may not be anticipated. We can model the effect of the transportation time into the demand variation of the stations. For ease of presentation, we assume that the travel time between any two adjacent stations is deterministic and the demand at each station follows a poisson process.

The vehicle will visit all the assigned stations in a predetermined route. Denote the travel time from the depot to station 1 by t_1 and from station i to $i + 1$ by t_{i+1}. Each station has demand that follows poisson distribution with demand rate λ_i units kiloliters per unit time. Therefore, the demand of station i, i.e., I_i, follows a poisson distribution with mean $(t_1 + \ldots + t_i)\lambda_i$.

The recursive dynamic equation is as follows,

$$V_i(K; l_i) = \max_{0 \leq x \leq \min(K, b_i - l_i)} [\gamma_i x - \alpha_i(a_i - l_i - x)^+ + f_{i+1}(K - x)],$$

with terminal function

$$V_{n+1}(K; \emptyset) = \delta K,$$

where $f_{i+1}(z) = E_{I_{i+1}, \ldots, I_n}[V_{i+1}(z; (l_{i+1}^0 - I_{i+1})^+]$ and $f_{n+1}(z) = \delta z$.

Given levels $l_{i+1}, ..., l_i$ define the critical numbers z_i^1 and z_i^2 as follows. Let $\Delta f_{i+1}(z) = f_{i+1}(z) - f_{i+1}(z - 1)$, if $\Delta f_{i+1}(1) < \gamma_i + \alpha_i$, $z_i^1 = 0$, otherwise $z_i^1 = \max\{z \in \{0, ..., K^0\} | \Delta f_{i+1}(z) \geq \gamma_i + \alpha_i\}$; if $\Delta f_{i+1}(1) < \gamma_i$, $z_i^2 = 0$, otherwise $z_i^2 = \max\{z \in \{0, ..., K^0\} | \Delta f_{i+1}(z) \geq \gamma_i\}$.

Theorem 3. *For each $i = 0, ..., n$,$V_i(K; l_i)$ is increasing and concave with respect to $0 \leq K \leq K^0$, for any given l_i. Furthermore, at each stage i, the optimal replenishment policy is as follows:*

(a) if $l_i \geq a_i$, $x^* = mid(K - z_i^2, 0, b_i - l_i)$;

(*b*) *if* $l_i < a_i$, $x^* = mid(\min(K - z_i^1, b_i - l_i), \max(K - z_i^2, 0), a_i - l_i)$;

where $mid(u, v, w) = v$ *if* $u \leq v \leq w$.

In the vast existing literatures on supply chain management and revenue management, the dynamic policy quite common to see is of threshold, however, the optimal policy we obtain is a middle point among three critical numbers, an interesting result. Bassok [3] also considered a problem similar to ours, which proved the optimal policy is of threshold. The main reason causing this difference is in our setting, in addition to the capacity limit at each station, the safety level requirement also affects the decision.

The optimal policy expressed in Theorem 3 is computationally implementable. Upon the vehicle's arrival at a particular station, the optimal solution can be computed by inputting the state of the art information on inventory levels at current station and subsequent stations to be visited.

5 Numerical Example

In this section, we use a numerical example to discuss the value of information updating. We consider a sample with $n = 3$ stations with given route $1, 2, 3$,and a vehicle capacity $K = 20$ units. The detailed data with respect to corresponding parameters are as follows.

Station capacity: $b_1 = 20$, $b_2 = 20$, $b_3 = 24$
Station safety level: $a_1 = 11$, $a_2 = 10$, $a_3 = 12$
Station unit profit: $\gamma_1 = 30$, $\gamma_2 = 38$, $\gamma_3 = 40$
Station unit penalty: $\alpha_1 = 25$, $\alpha_2 = 20$, $\alpha_3 = 15$

We want to compare the difference in the expected return of using the two stochastic models discussed in Section 4. According to the analysis in last section, we can easily calculate the optimal decision made before the delivery arrival, i.e.,

$$\hat{x} = arg \max_x \{E_{l_i}[\gamma_i min(x, b_i - l_i) - \alpha_i(a_i - l_i - x)^+ + \hat{V}_{i+1}(K - x)]\}.$$

In order to focus on the evaluation of information updating at a particular station, we assume after that station, the decisions made will take into account demand updating, which is expressed below

$$F_i(x; K, l_i) = \gamma_i x - \alpha_i(a_i - l_i - x)^+ + f_{i+1}(K - x). \tag{4}$$

Equation (4) is actually an upper bound of the expected return without information updating. We compare it with the optimal profit considering information updating.

We use the following performance measure to evaluate the value of information for a specific realization of random demand (i.e. l_i),

$$\frac{V_i(K; l_i) - F_i(\hat{x}; K, l_i)}{V_i(K; l_i)} * 100\%.$$

For the case with 3 stations described above, we compare the expected value before and after the driver arrives at station 1. The specific results are listed in Table 3. According to Table 3, we can see from an extreme case, $l_1 = 20$, near 30% profit loses due to the lack of information updating.

Table 3. Comparing the result from two stochastic models

Remain inventory in station i (l_i)	Expected Return by model in (4.1)	Expected Return by model in (4.2)	Percentage difference
0	686.0148	627.0712	8.6005
1	697.4380	652.0712	6.4562
2	708.1999	677.0712	4.3785
3	718.2566	702.0712	2.2284
4	727.0712	727.0712	0
5	735.3859	727.0712	1.0884
6	743.5551	727.0712	2.4161
7	751.6111	727.0712	3.1957
8	759.5181	727.0712	4.2161
9	767.1328	727.0712	5.2151
10	774.1670	727.0712	6.0724
11	780.1702	727.0712	6.7949
12	780.1702	727.0712	6.7949
13	780.1702	727.0712	6.7949
14	780.1702	727.0712	6.7949
15	780.1702	727.0712	6.7949
16	780.1702	697.0712	10.6410
17	780.1702	667.0712	14.4871
18	780.1702	637.0712	18.7179
19	780.1702	607.0712	22.1795
20	780.1702	577.0712	26.0256

6 Concluding Remarks

In this paper, we considered a commodity distribution problem with fixed capacity under the framework of VMI. Specifically, we considered an gasoline vendor, who distributes oil to multiple retail stations by a tanker. At each station, there is a contracted safety inventory level, not satisfying which will be penalized by the station owner. Given the specified delivery route of the tanker, the vendor needs to decide a replenishment quantity at each station, facing the trade-off of replenishing more inventory at current station or reserving more capacity for later stations.

We modelled this problem in two ways, corresponding to different realistic situations. For the problem with stationary demand, we developed a Greedy Algorithm, which is proved optimal, to solve a constrained nonlinear objective function. For the random demand case, we formulated the problem by dynamic programming. The optimal policy obtained is a median among three critical

numbers, different from previous similar researches. Investigating valuable extensions of our model includes the following: (1) As the vehicle route is fixed in our current work, incorporating the routing problem should be challenged; (2) In our work, the commodity to be distributed is a single product, it is more interesting to consider multi-commodity distribution problem which will involve the initial space allocation among different commodities.

References

1. Anton, J.K., Nori, V.S., Martin, S.: The Stochastic Inventory Routing Problem with Direct Deliveries. Transportation Science 36(1), 94–118 (2002)
2. Barnes, S.D., Bassok, Y.: Direct Shipping and the Dynamic Single-depot/Multi-retailer Inventory System. Elsevier Science 101(3), 509–518 (1997)
3. Bassok, Y.: Dynamic Allocations for Multi-Product Distribution. Transportation Science 29(3), 256–266 (1995)
4. Bell, W.J., Dalberto, L.M., Fisher, M.L., Greenfield, A.J., Jaikumar, J., Kedia, P., Mack, R.G., Prutzman, P.J.: Improving the distribution of industrial gases with on-line computerized routing and scheduling optimizer. Interfaces 13(1), 4–23 (1983)
5. Beltrami, E., Bodin, L.: Networks and Vehicle Routing for Municipal Waste Collection. Networks 4, 65–94 (1974)
6. Champell, A., Clarke, L., Anton, K., Martin, S.: The Vehicle Routing Problem, Chapter 12 (2002)
7. Christiansen, M., Nygreen, B.: A Method for Solving Ship Routing Problems with Inventory Constraints. Ann. Oper. Res. 81, 357–378 (1998a)
8. Christiansen, M.: Modelling Path Flows for a Combined Ship Routing and Inventory Management Problem. Ann. Oper. Res. 82, 391–412 (1998b)
9. Christiansen, M.: Decomposition of a Combined Inventory and Time Constrained Ship Routing Problem. Transportation Science, 33(1), 3–16 (1999)
10. Berman, O., Larson, R.C.: Deliveries in Inventory/Routing Problem Using Stochastic Dynamic Programming. Transportation Science 35(2), 192–213 (2001)
11. Singer, M., Donoso, P., Jara, S.: Fleet Configuration Subject to stochastic demand: An Application in The Distribution of Liquefied Petroleum Gas. Journal of The Operational Research Society 53, 961–971 (2002)

On-Line Bin Packing with Arbitrary Release Times

Yongqiang Shi[1] and Deshi Ye[2,⋆]

[1] College of Economics, Zhejiang University, Hangzhou 310027, China
[2] College of Computer Science, Zhejiang University, Hangzhou 310027, China

Abstract. We study a new variant of on-line bin packing problem, in which each item a_i is associated with a size a_i and also a release time r_i so that it must be placed at least r_i above from the bottom of a bin. Items arrive in turn and must be assigned without any knowledge of subsequence items. The goal is to pack all items into unit-size bins using the minimum number of bins. We study the problem with all items have equal size. First, we show that the ANY FIT algorithm cannot be approximated within any constant. Then we present a best possible on-line algorithm with asymptotic competitive ratio of 2.

Keywords: Bin packing, Arbitrary release time, On-Line algorithm.

1 Introduction

Bin packing is one of the basic problems since the early 70's in the theoretical computer science and combinatorial optimization literature and different variants of the problem continue to attract researchers' attentions. In this problem, one is given a list L of items $a_1, a_2, \ldots, a_n$, each item $a_i \in (0, 1]$, and the goal is to pack this sequence of items into unit-size bins using the minimum number of bins. A packing algorithm is called *on-line* if it packs item one by one without any information on subsequence items. We call a packing algorithm *off-line*, if the algorithm knows all the information of items.

In this paper we study a new variant of on-line bin packing problem, which is called *on-line bin packing with arbitrary release time*. In this variant, each bin has a time axis $[0, 1]$ from the bottom to top, jobs arrive in turn and must be assigned without information of future items. Each item a_i is associated with a size a_i and a release time r_i so that it must be placed at least r_i above from the bottom of a bin. The goal is to pack all items into unit-size bins and minimize the number of bins used. Without loss of generality, we assume that $a_i + r_i \leq 1$, otherwise, this item cannot be assigned to any bin. In this paper, we focus on the problem of *unit size items*. All items have the same size of $1/K$ for some integer $K \geq 1$.

Unlike traditional on-line models, where items arrive one by one without release times, or items arrive over time but the items appear only after their release

⋆ Research was supported by NSFC(10601048).

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 340–349, 2007.

times, the on-line model studied in this paper is that items arrive one by one with arbitrary release times. Upon the arrival of items, their sizes and release times are known to the scheduler. Items must immediately and irrevocably be assigned without any information on subsequence items. Furthermore, items should be fitted into a position after their release times.

To our best knowledge, this kind of on-line model was first proposed by Li and Huang [8]. In their paper, each job has a release time and a processing time, jobs cannot be scheduled before its release time. Jobs arrive in turn and must be scheduled without any information of subsequence jobs while their release times are independent of the order. The problem is to find a schedule of these jobs while minimizing the overall completion time, i.e. the *makespan*. They gave the tight competitive ratio $3 - 1/m$ of List Scheduling (LS) and also proposed an improved algorithm for $m \geq 2$.

The on-line bin packing problem with arbitrary release time can be regarded as a variant of on-line scheduling problem for jobs with arbitrary release time. Each job is also associated with processing time and release time, however all jobs have common due date. The goal of this scheduling problem is to schedule all the jobs before its due date while minimizing the number of machines used. This scheduling problem is motivated by air cargo import terminal problem. Cargo agents will make requests daily either by fax or through web or by phone to book for truck docks at the terminal for the cargo collection. Cargo agents will also indicate their preference time per day to come to the terminal. Since the truck docks are scarce resource, it is significant to efficiently schedule all the jobs in a day to reduce using the number of docks. Comparing to the bin packing problem, the processing time of a job can be regarded as the size of an item, the common due date can be regarded as the size of a bin. This bin packing problem arises also in many real world applications: e.g. in the real-world packing of bins, some items are fragility and cannot be assigned to a bin with too much items above.

The standard measure of algorithm's quality for on-line bin packing is the *asymptotic competitive ratio*. Given a list L of items and an on-line algorithm A, we denote by $OPT(L)$ and $A(L)$, respectively, the minimum possible number of bins used to pack items in L and the number of bins used by algorithm A on L. The *asymptotic competitive ratio* R_A^∞ of algorithm A is defined to be

$$R_A^\infty = \limsup_{n\to\infty} \max_L \{A(L)/OPT(L) | OPT(L) = n\}.$$

Throughout of this paper, we will often simply write *competitive ratio* instead of "asymptotic competitive ratio".

Related results: Our studied problem becomes the classical on-line bin packing problem when all the jobs are released at time zero. There is a long history of results for the classical bin packing problem (see the survey [3,4,5]). The bin packing is well-known to be NP-hard [6]. The classical on-line bin packing problem was first investigated by Ullman [10]. The FIRST FIT algorithm [7] was shown to have competitive ratio 17/10. A straight forward class of on-line

algorithm is the ANY FIT (AF) algorithm: the AF algorithm never puts an item a_i into an empty bin unless the item does not fit into any partially filled bin. NEXT FIT, FIRST FIT and BEST FIT algorithms belong to the class of ANY FIT. The current best known lower and upper bounds for classical on-line bin packing problem are 1.54 due to van Vliet [11] and 1.588 by Seiden [9], respectively.

A generalization of our problem is scheduling with release times and deadlines on a minimum of machines (SRDM) [2]. Each job has a release time, a processing time and a deadline, the goal is to find a non-preemptive schedule such that all jobs meet their deadlines and the number of machines needed to process all jobs is minimum. They concerned on the off-line case and showed that this problem is at least $\Theta(\log n)$-approximation, where n is the total number of jobs. For the special case with equal processing times, a 9-approximation algorithm was presented. They also presented an asymptotic 4.62-approximation algorithm with all jobs have equal release times. Our problem becomes a special case of the SRDM problem if all jobs have equal deadlines. Another similar problem called *strip packing* with release time was studied by Augustine et al. [1]. They provided an asymptotic full polynomial time approximation scheme for the off-line strip packing with release time.

Our contributions: To our best knowledge, we are the first one to study the on-line bin packing problem with release times. We assume that all items have equal size. The detail of our results are given as follows:

1) There is no constant competitive ratio for the ANY FIT algorithm.
2) We show that no on-line algorithms can achieve competitive ratio less than 2.
3) We present an on-line algorithm with competitive ratio 2.

The rest of this paper is organized as follows. Section 2 analyzes the lower bound of ANY FIT algorithms. Section 3 shows the general lower bound 2 and presents an on-line algorithm with competitive ratio 2. The conclusions will be given in Section 4.

2 Lower Bound of the ANY FIT Algorithm

In this section we show an unbounded lower bound for the ANY FIT algorithm. Note that for the AF algorithm, a new bin is opened only if the item does not fit into any non-empty bin.

Theorem 1. *There is no constant competitive ratio for the ANY FIT algorithm.*

Proof. We construct an instance I_0 to show the theorem. First we assume that all items have the same size $1/K$, where K is a sufficiently large integer. K^2 items arrive in turn : K items with release time 0, K items with release time $1/K$, K items with release time $2/K, \ldots, K$ items with release time $(K-1)/K$. This instance I_0 will also be used in the rest of this paper.

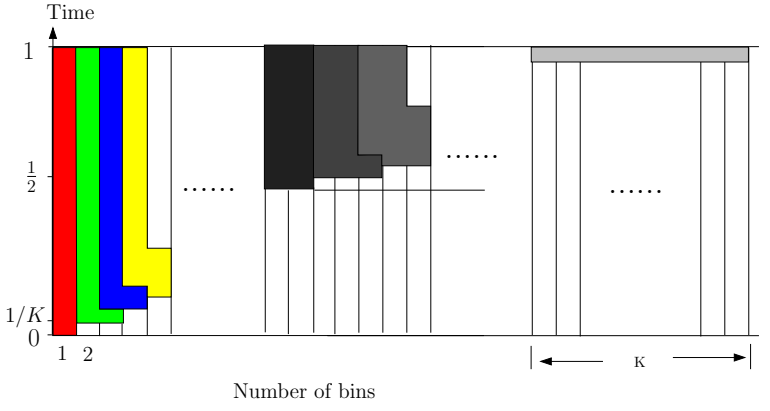


Fig. 1. ANY FIT Packing of instance I_0

Clearly, the optimal solution uses K bins, where each item is located exactly at its release time. Now we observe the assignments by the ANY FIT algorithm, see the Fig. 1 for an illustration, each color represents a set of items with the same release time.

The first K items are packed into the first bin, i.e., only one bin is used. For the second K items, $K-1$ items are in the 2nd bin and one item in the 3rd bin, i.e., use $K/(K-1)$ bins. Because of the ANY FIT algorithm, when a new bin is opened, no subsequence items can be fitted into a bin with index smaller than this opened new bin. For the K items with release time x/K, at least $\lfloor 1/(1-x/K) \rfloor$ bins are needed. For the last K items with release time $\frac{K-1}{K}$, each piece shall occupy a bin, i.e. total K new bins are needed to pack the last K items. Consequently, the number of bins used by the algorithm AF is at least $K/K+\lfloor K/(K-1)\rfloor+\cdots+\lfloor K/2\rfloor+K \geq (1/2+1/3+\cdots+1/K)K = K(\log K-1)$. Then the competitive ratio of the ANY FIT algorithm is at least

$$R_{AF} \geq \frac{AF(I)}{OPT(I)} = \frac{K(\log K-1)}{K} = \log K - 1.$$

Hence there is no constant competitive for the ANY FIT algorithm when K approaches infinity. □

Note that the above instance I_0 works also for the model with unit size. Thus one question arises: does there exists an on-line algorithm with constant competitive ratio even if the case with unit size? We will give a positive answer in the following section.

3 Optimal On-Line Algorithm

3.1 General Lower Bound

In this section we show that one cannot expect to find an on-line algorithm with asymptotic competitive ratio less than 2. Note that the classical bin packing

problem is solved in polynomial time if all items have equal size. The ANY FIT algorithm will also produce an optimal solution for the unit size model. Thus each item with additional release time brings much more trouble to our investigated problem.

Theorem 2. *For any on-line algorithm A,* $R_A^{\infty} \geq 2$.

Proof. Suppose that R is the competitive ratio of on-line algorithm A. For an integer $k \geq 3$. The sequence of items generated by the adversary is given as follows. First, n items with release time 0 and all have item size $1/k$.

After first n items, OPT is n/k, for large n divisible by k, so only Rn/k items are packed at time 0. Then for each i, n items with release time i/k are presented one by one, where $i = 1, \ldots, k-1$.

Similarly, considering the situation after in items, OPT is in/k, thus at most iRn/k items are packed starting at time $i-1$. So, summing the arithmetic sequence, the total number of packed items at the end is at most $\sum_{i=0}^{k-1}(i+1)Rn/k = R(k+1)n/2$. But this must be at least kn, i.e. $R(k+1)n/2 \geq kn$ yielding $R \geq 2k/(k+1)$.

Thus, the asymptotic competitive of any on-line algorithm is at least 2 by choosing a sufficiently large integer k. □

3.2 Upper Bound

In this section, we present an optimal on-line algorithm for the problem with unit size, i.e. all the items have the same size $1/K$ for some integer K, therefore we can assume that the release times of items are at $\{i/K | i = 0, 1, \ldots, K-1\}$.

Main idea: From the analysis of the lower bound of algorithm AF, we observe that it is not good always packing items into a non-empty bin. Thus the key point to design an efficient algorithm is to find a suitable rule when will a new bin be opened.

Before designing our algorithm, some notations are given as follows. When packing a new item a_j, suppose that the optimal solution for these first j items is $OPT_j = M$, then denote by l_i^M the number of items with release time i/K, $(i = 0, 1, ..., K-1)$. Let $L_i^M = \sum_{j=i}^{K-1} l_j^M \leq (K-i)M$. Clearly, we have the following lemma due to the capacity constraint of any optimal solution.

Lemma 3

$$M \geq \max_{0 \leq i \leq K-1} \{\frac{L_i^M}{K-i}\}.$$

Algorithm $AH_C\{\frac{1}{K}\}$**(Average Height)**

For any new item a_j with release time r_j, we pack it as follows:

Step 1. Calculate the optimal solution $OPT_j = M = \lceil \max_{0 \leq i \leq K-1}\{\frac{L_i^M}{K-i}\}\rceil$.
Step 2. If the optimal solution increases, i.e. $OPT_j > OPT_{j-1}$, open C bins.
Step 3. Find the lowest indexed bin so that the item a_j is located as low as possible in all open bins.

From the fact of our algorithm, we always first put an item a_j into a bin with lowest index so that it can be located at r_j, then it is easy to obtain the following Fact.

Fact 4. *If there is a vacancy at j/K in the bin $B^{'}$, there must be a vacancy at j/K in the non-empty bins with index larger than $B^{'}$. Furthermore if there is no vacancy at j/K in the bin $B^{'}$, there must be no vacancy at j/K in the bins with index less than $B^{'}$.*

Here we want to find the minimum constant C so that all items can be packed into $C \cdot OPT_n$ bins. Then the competitive ratio of the algorithm $AH_C\{\frac{1}{K}\}$ is C.

Lemma 5. *For any instance I, if there are two adjacent items a_j and a_{j+1} with release times $r_j > r_{j+1}$, the solution of $AH_C\{\frac{1}{K}\}$ will not be better for the instance I with interchanging a_j and a_{j+1}. Moreover, the instance I_0 generated in the proof of Theorem 1 is one of the worst instances by $AH_C\{\frac{1}{K}\}$.*

Proof. Without loss of generality, we assume that all items of the instance I can be packed into $C \cdot OPT_n$ bins. Denote I' to be the instance I after interchanging the items a_j and a_{j+1}. First of all, we study the case $OPT_j = OPT_{j-1}$. For instance I', after a_{j+1} has been assigned, the optimal solution is denoted by $OPT_j^{'}$ and the optimal solution after a_j is $OPT_{j+1}^{'}$. We consider the following cases of possible locations of items a_j and a_{j+1} in the instance I by the algorithm $AH_C\{\frac{1}{K}\}$, and also the cases of the optimal solution increases or not upon the arrival of a_{j+1} in I'.

• Case 1: The two items are both located exactly at their release times.

◦ Case 1.1: $OPT_j = OPT_{j+1}$. It implies that when a_j comes there are vacancies at r_j and r_{j+1}. For instance I', their locations will not be changed by $AH_C\{\frac{1}{K}\}$.

◦ Case 1.2: $OPT_{j+1} > OPT_j$. In this case, C bins will be opened before assigning a_{j+1}.

-Subcase 1.2.1: a_{j+1} isn't packed into these C new bins. Similar analysis as Case 1.1.

-Subcase 1.2.2: a_{j+1} is packed into the first of these C new bins. There is no vacancy just at r_{j+1} in the open bins before a_{j+1} comes. For instance I', we have $OPT_{j-1}^{'} = OPT_j^{'} = OPT_j$. From the algorithm, a_{j+1} must be located above r_{j+1}. Consider the following cases:

a): a_{j+1} is located at r^* between r_{j+1} and r_j. The detail proof will be given in full paper.

b): a_{j+1} is located at r_j. In this case, a_{j+1} occupies the position which is for a_j in the stance I. Similarly with *subcase 1.2.2.a)*, the solution will increase 1 or does not decrease.

• Case 2: Only a_j is located at its release time. In this case, $OPT_j = OPT_{j+1}$.

◦ Case 2.1: a_{j+1} is located below r_j. After interchanging of the two items a_j and a_{j+1}, their locations will not change.

◦ Case 2.2: a_{j+1} is located at or above r_j. Consider the instance I', the locations of a_j and a_{j+1} will interchange comparing to their locations in the instance I. This will not affect the locations of subsequential items.

• Case 3: a_j is located above r_j. It means there is no vacancy between r_j and its location in any open bin.

◦ Case 3.1: a_{j+1} is located exactly at its release time.

-Subcase 3.1.1: $OPT_j = OPT_{j+1}$. From the algorithm $AH_C\{\frac{1}{K}\}$, their locations will not change in the stance I'.

-Subcase 3.1.2: $OPT_j < OPT_{j+1}$. In this case C bins will be opened before assigning the item a_{j+1}.

a): a_{j+1} isn't packed into these C new bins, same analysis as *Subcase 3.1.1.*

b): a_{j+1} is packed into the first of these C new bins, similar to the analysis in *Subcase 1.2.2.*

◦ Case 3.2: a_{j+1} is located above r_{j+1}. In this case, $OPT_j = OPT_{j+1}$ still holds. The analysis is similar to *Case 2.*

-Subcase 3.2.1: a_{j+1} is located between r_{j+1} and r_j. Their locations will not change in the instance I'.

-Subcase 3.2.2: a_{j+1} is located above or at r_j. Their locations will interchange in the instance I' and keep the same locations of subsequential items as the instance I.

From the above all and the case $OPT_j > OPT_{j-1}$ (will be given in full paper), by interchanging such items a_j and a_{j+1} with $r_j > r_{j+1}$ in the stance I, we obtain that an instance with items with release times in nondecreasing order will achieve the worst solution by the algorithm $AH_C\{\frac{1}{K}\}$, i.e. for the instances with the same items and but with different arrival sequences, the instance with release times in nondecreasing order will use the largest number of bins by $AH_C\{\frac{1}{K}\}$. However, their optimal solutions are the same. Hence, the instance I_0 generated in the proof of Theorem 1 is one of the worst instances. □

Theorem 6. *The competitive ratio of algorithm $AH_C\{\frac{1}{K}\}$ is 2 if we let $C = 2$.*

Proof. By Lemma 5, the instance I_0 is one of the worst instances, and the optimal solution of I_0 is K. Firstly, we construe the configuration of solution for I_0 by $AH_C\{\frac{1}{K}\}$. Afterward, we will analyze the other worst instances with release times in nondecreasing order.

Let $C = 2$. See Fig. 2 for a demonstration of the packing instance I_0 when K is even.

The first K items are packed into 2 bins under $\frac{1}{2}$. $K-2$ items of the second K ones are packed into the $3rd$ and $4th$ bins under $\frac{1}{2}$, and each of the other 2 items of them is put into the first 2 bins at $\frac{1}{2}$. For any K items with release times $i/K (i = 1, 2, \ldots, K-1)$, consider the following cases.

a): if i is odd, suppose $i = 2h-1$, $K-2h$ items of them are packed into the $(2i+1)-th$ and $(2i+2)-th$ bins under $\frac{1}{2} + \frac{h-1}{K}$, and each of the other $2h$ of them is located at $\frac{1}{2} + \frac{h-1}{K}$ in the former $2h$ bins;

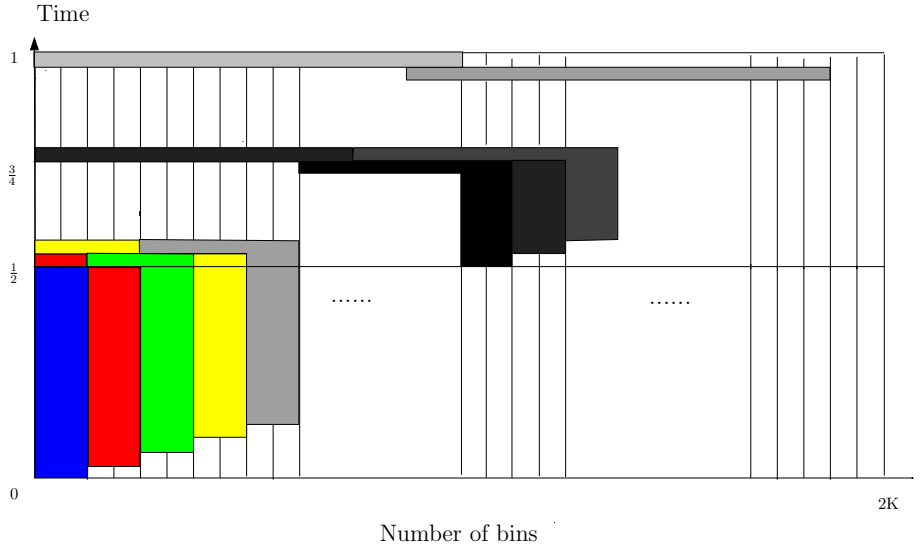


Fig. 2. $AH_2\{\frac{1}{K}\}$ Packing for Even K

b): i is even, assume $i = 2h$, $K - 2h - 2$ items of them are packed into the $(2i+1) - th$ and $(2i+2) - th$ bins under $\frac{1}{2} + \frac{h-1}{K}$, and each of the other $2h+2$ of them is located at $\frac{1}{2} + \frac{h-1}{K}$ in the later $2h+2$ bins.

For the last K items whose release times are $\frac{K-1}{K}$, $(K-1)$ is odd. These K items are just located at $\frac{K-1}{K}$ in the former K bins and total $2K$ bins are opened.

If K is odd, see the Figure 3. For the first K items, $(K+1)/2$ items are packed into the $1st$ bin and $(K-1)/2$ items are packed into the $2nd$ bin. Similar to the above analysis, when K items comes with release times $i/K(i = 1, 2, \ldots, K-1)$, we consider the following cases.

a): if i is odd, set $i = 2h - 1$, $K - 2h - 1$ items of them are packed into the $(2i+1) - th$ and $(2i+2) - th$ bins under $\frac{K-1}{2K} + \frac{h-1}{K}$, and each of the other $2h+1$ of them is located at $\frac{K-1}{2K} + \frac{h-1}{K}$ in the later $2h+1$ bins.

b): i is even, set $i = 2h$, $K - 2h - 1$ items of them are packed into the $(2i+1) - th$ and $(2i+2) - th$ bins under $\frac{K-1}{2K} + \frac{h}{K}$, and each of the other $2h+1$ of them is located at $\frac{K-1}{2K} + \frac{h}{K}$ in the former $2h+1$ bins.

For the last K items with release time $\frac{K-1}{K}$, $(K-1)$ is even. These K items are just located at $\frac{K-1}{K}$ in the former K bins of all the $2K$ ones.

From the above analysis, the last 2 bins are empty. But, if there are only first $P(P < K)$ batches of items with P different release times, the optimal solution is P while the solution of $AH_2\{\frac{1}{K}\}$ is $2P$ and none of bins is empty. Thereby the ratio is 2.

From now on we have discussed the case that the number of items with the same release time are all equal to K. If they are all equal to some multiple of K,

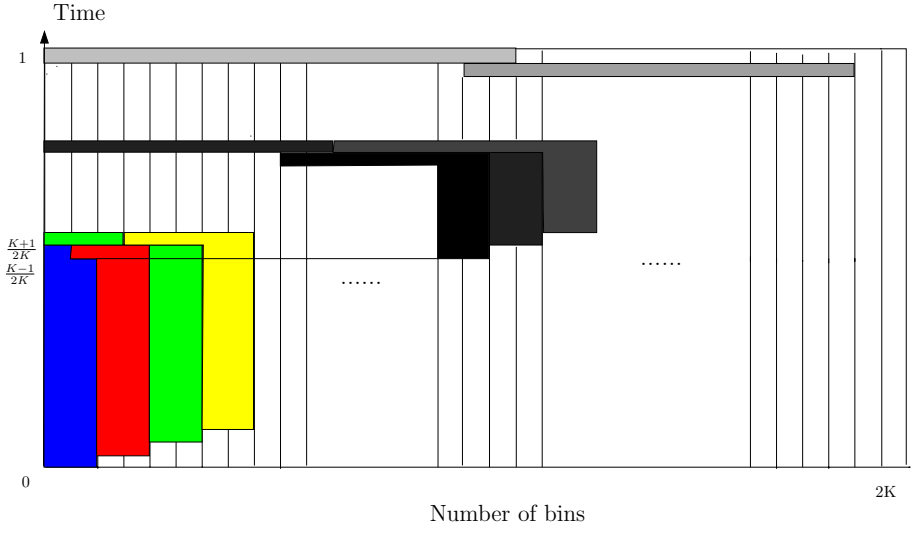


Fig. 3. $AH_2\{\frac{1}{K}\}$ Packing for Odd K

the configuration of the algorithm $AH_2\{\frac{1}{K}\}$ is alike. Furthermore, for the state that the numbers of items with same release time are different, the vacancies brought by the items whose number is not multiple of K are more than those vacancies brought by the items with number of K's multiple. With analogy analysis, the solution of algorithm $AH_2\{\frac{1}{K}\}$ is at most twice of optimal solution. This completes the proof of the theorem. □

In the following, we give an improved algorithm $MAH_C\{\frac{1}{K}\}$ with tight competitive ratio $2 - 1/K$.

Algorithm $MAH_C\{\frac{1}{K}\}$
For any new item a_j with release time r_j, we pack it as follows:

Step 1. Calculate the optimal solution $OPT_j = M = \lceil \max_{0 \le i \le K-1}\{\frac{L_i^M}{K-i}\}\rceil$.
Step 2. If $OPT_j = 1$, put the item into the first bin.
Step 3. If $OPT_j \ge 2$ and $OPT_j > OPT_{j-1}$, open new C bins.
Step 4. Find the lowest indexed bin so that the item a_j is located as low as possible in all open bins.

The analysis of above algorithm is similar to the theorem 6, but the detail proof will be given in full paper.

Theorem 7. *Let $C = 2$, the competitive ratio of algorithm $MAH_C\{\frac{1}{K}\}$ is $2-\frac{1}{K}$.*

4 Conclusions

In this paper we have studied the on-line bin packing problem with release times. For the problem with all items have equal size, we presented a general

lower bound 2 and designed an on-line algorithm with asymptotic competitive ratio 2. We also showed that the ANY FIT algorithm cannot be approximated within any constant asymptotic competitive ratio.

If we turn to the off-line case of our problem, all the classical on-line bin packing algorithms can be applied to our problem by sorting the items in the order of non-increasing release times and all items are packed to a bin as top as possible.

There are many challenging aspects of the bin packing with release time. For example, study the on-line problem with items have different sizes, study the off-line problem whether there exists a full polynomial time approximation scheme.

Acknowledgements. We would like to thank anonymous referees for many helpful suggestions to design a better general lower bound and for many suggestions to improve the presentation of this paper.

References

1. Augustine, J., Banerjee, S., Irani, S.: Strip Packing with Precedence Constraints and Strip Packing with Release Times. In: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures, pp. 180–188 (2006)
2. Cieliebak, M., Erlebach, T., Hennecke, F., Weber, B., Widmayer, P.: Scheduling with Release Times and Deadlines on a Minimum Number of Machines. IFIP TCS, 209–222 (2004)
3. Coffman Jr., E.G., Galambos, G., Martello, S., Vigo, D.: Bin packing approximation algorithms: Combinatorial analysis. In: Du, D.-Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Dordrecht (1998)
4. Coffman Jr., E.G., Garey, M.R., Johnson, D.S.: Bin packing approximation algorithms: A survey. In: Hochbaum, D.S. (ed.) Approximation algorithm for NP-hard problems, pp. 46–93. PWS (1996)
5. Csirik, J., Woeginger, G.J.: On-line packing and covering problems. In: Goto, E., Nakajima, R., Yonezawa, A., Nakata, I., Furukawa, K. (eds.) RIMS Symposium on Software Science and Engineering. LNCS, vol. 147, pp. 147–177. Springer, Heidelberg (1983)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability (A guide to the theory of NP-completeness.). W. H. Freeman and Company, San Francisco (1979)
7. Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L.: Worst-case performance bounds for simple one-dimensional packing algorithms. Siam Journal on Computing 3, 256–278 (1974)
8. Li, R., Huang, H.-C.: On-line Scheduling for Jobs with Arbitrary Release Times. Computing 73, 79–97 (2004)
9. Seiden, S.: On the online bin packing problem. Journal of the ACM 49, 640–671 (2002)
10. Ullman, J.D.: The performance of a memory allocation algorithm. Technial Report 100, Princeton University, Princeton, NJ (1971)
11. van Vliet, A.: An improved lower bound for on-line bin packing algorithms. Information processing letters 42, 277–284 (1992)

On the Complexity of the Max-Edge-Coloring Problem with Its Variants

Chang Wu Yu

Department of Computer Science and Information Engineering
Chung Hua University, Hsinchu, Taiwan 300, R.O.C
cwyu@chu.edu.tw

Abstract. The *max-edge-coloring* problem (MECP) is finding an edge colorings $\{E_1, E_2, E_3, \ldots, E_z\}$ of a weighted graph $G=(V, E)$ to minimize $\sum_{i=1}^{z} \max \{w(e_k) | e_k \in E_i\}$, where $w(e_k)$ is the weight of e_k. In the work, we discuss the complexity issues on the MECP and its variants. Specifically, we design a 2-approximmation algorithm for the max-edge-coloring problem on biplanar graphs, which is bipartite and has a biplanar drawing. Next, we show the splitting chromatic max-edge-coloring problem, a variant of MECP, is NP-complete even when the input graph is restricted to biplanar graphs. Finally, we also show that these two problems have applications in scheduling data redistribution on parallel computer systems.

Keywords: edge coloring, bipartite graphs, multi-graphs, algorithm design.

1 Introduction

The *max-edge-coloring* problem (MECP) is finding an edge colorings $\{E_1, E_2, E_3, \ldots, E_z\}$ of a weighted graph G to minimize

$$\sum_{i=1}^{z} \max \{w(e_k) | e_k \in E_i\},$$

where $w(e_k)$ is the weight of e_k. We can also define its two variants by using the minimum number of coloring as follows. The *chromatic max-edge-coloring* problem (CMECP) is finding an edge colorings $\{E_1, E_2, E_3, \ldots, E_\Delta\}$ of a weighted graph G to minimize

$$\sum_{i=1}^{\Delta} \max \{w(e_k) | e_k \in E_i\},$$

where Δ is the maximum degree of G and $w(e_k)$ is the weight of e_k. Given a weighted graph $G=(V, E)$, the *splitting chromatic max-edge-coloring* problem (SCMECP) is an decision problem whether we can add extra edges E' such that the resulting graph $G'=(V, E \cup E')$ satisfies the following four conditions:

(1) Another edge (s, t) can be added to E' only if there exists an edge (s, t) in E.
(2) The maximum degree of the resulting new graph is the same; that is, $\Delta(G)=\Delta(G')$.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 350–361, 2007.

(3) The value

$$\sum_{i=1}^{\Delta} \left\{ w(e_k) \middle| e_k = (s,t) \text{ in } E' \bigcup E \right\}$$

equals the weight of (s, t) in E, where $w(e_k)$ is the weight of e_k.

(4) There is an edge coloring $\{E_1, E_2, E_3, \ldots, E_\Delta\}$ of $G'=(V, E \cup E')$ such that

$$\sum_{i=1}^{\Delta} \max \left\{ w(e_k) \middle| e_k \in E_i \right\} \leq K,$$

where K is given and $w(e_k)$ is the weight of e_k.

To the best of our knowledge, the above three problems are defined formally for the first time. Although some heuristic algorithms have been designed for the CMECP [19, 20, 21], both the MECP and the CMECP are still open to devise a polynomial-time algorithm even when the input graph is restricted to biplanar graphs. The first contribution of the work is showing the existence of 2-approximation algorithm for the MECP when the input graph is restricted to biplanar graphs. The SCMECP is a new graph problem defined here, which have applications in shortening the overall communication time for data redistribution in parallel systems. In this work, we will show the SCMECP is NP-complete even when the input graph is restricted to biplanar graphs.

Only a similar problem called the *max-coloring problem* [28, 29] can be found in literature. The problem is to find a proper vertex coloring of input graphs whose color classes $C_1, C_2, \ldots, C_k$, minimize $\sum_{i=1}^{k} \max \left\{ w(e) \middle| e \in C_i \right\}$ where $w(e)$ is the weight of e. The max-coloring problem has been shown to be NP-hard on interval graphs [28]; however, there exist some approximation algorithms for the problem on many well-known subclasses of graphs including bipartite graphs, interval graphs, circle graphs, circular arc graphs, unit disk graph, chordal graphs, and permutation graphs [28, 29].

The rest of the paper is organized as follows. Section 2 presents necessary definitions and notations. Next, Section 3 presents a 2-approximation algorithm for the MECP when the input graph is restricted to biplanar graphs. The SCMECP is shown to be NP-complete even when the input graph is restricted to biplanar graphs in Section 4. The applications of these problems to scheduling problems for data redistribution in parallel systems are discussed in Section 5. Finally, Section 6 concludes the paper.

2 Definitions and Notations

A *graph* G consists of a finite nonempty vertex set together with an edge set. A *bipartite graph* $G=(S, T, E)$ is a graph whose vertex set can be partitioned into two subsets S and T such that each of the edges has one end in S and the other end in T. A typical convention for drawing a bipartite graph $G=(S, T, E)$ is to put the vertices of S on a line and the vertices of T on a separate parallel line and then represent edges by placing straight line segments between the vertices that determine them. In this convention, a drawing is *biplanar* if edges do not cross, and a graph G is *biplanar* if it has a biplanar drawing [27]. A graph is *connected* if there is a path joining each pair of nodes. An *acyclic* graph is one that contains no cycles. A *forest* is an acyclic graph. A *tree* is a connected acyclic graph. A *component* of a graph is a maximal connected subgraph. The number of components of G is denoted by $\omega(G)$.

Let $N(v)$ denote the set of vertices which are adjacent to v in G. The ends of an edge are said to be *incident* with the edge. Two vertices which are incident with a common edge are *adjacent*. A *multi-graph* is a graph allowing more than one edge to join two vertices. The degree $d_G(v)$ of a vertex v in G is the number of edges of G incident with v. We denote the maximum degree of vertices of G by $\Delta(G)$.

A *complete bipartite graph* $G=(S, T, E)$ is a graph such that each vertex of S is joined to each vertex of T; if $|S|=m$ and $|T|=n$, such a graph is denoted by $K_{m,n}$. An ordering of S (T) has the *adjacency property* if for each vertex $v\in T(S)$, $N(v)$ contains consecutive vertices in this ordering. The graph $G=(S, T, E)$ is called a *doubly convex-bipartite graph* if there are orderings of S and T having the adjacency property [24]. A graph is called *interval graph* if its vertex set can be represented by using a finite number of interval on a straight line and two vertices are connected by an edge when the corresponding intervals overlap at least partially.

The *line graph* $L(G)$ of a graph $G=(V, E)$ is defined so that there is a one-to-one correspondence between the vertices in $L(G)$ and the edges in G. That is, there is an edge joining two vertices in $L(G)$ when their corresponding edges in G are incident with a common vertex.

The coloring is *proper* if no two adjacent edges have the same color. An edge with identical ends is called a *loop*. A *k-edge coloring* of a loopless graph G is an assignment of k colors to the edges of G. G is *k-edge colorable* if G has a proper k-edge coloring. The *edge chromatic number* $\chi'(G)$, of a loopless graph G, is the minimum k for which G is k-edge-colorable. A subset M of E is called a *matching* in $G=(V, E)$ if its elements are links and no two are adjacent in G. Note that the each edge set with the same color in a proper edge coloring forms a matching. At last, most graph definitions used in the paper can be found in [22].

An algorithm that generates near-optimal solution is called an *approximation algorithm*. We say that an approximation algorithm has a ratio-bound of ρ, called *ρ-approximation algorithm*, if for any input of size, the cost C of the solution produced by the approximation algorithm is within a factor of ρ of the cost C^* of an optimal solution: $\max(C/C^*, C^*/C)\leq \rho$.

3 The 2-Approximation Algorithm for the Max-Edge-Coloring Problem When the Input Graph Is Restricted to Biplanar Graphs

This section gives a 2-approximation algorithm for the max-edge-coloring problem when the input graph is restricted to biplanar graphs. First, the following theorem presents additional properties of biplanar graphs.

Theorem 1: The following four statements are equivalent [25-27]:

(1) A bipartite graph G is biplanar.
(2) The graph G is a collection of disjoint caterpillars.
(3) The graph G contains no cycle and no double claw.
(4) The graph G^* that is the remainder of G after deleting all vertices of degree one, is acyclic and contains no vertices of degree at least three.

Here a *caterpillar* is a connected graph that has a path called the *backbone b* such that all vertices of degree larger than one lie on *b*; and a *double claw* graph is depicted in Fig. 1.

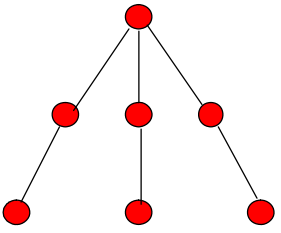


Fig. 1. A double claw

The size of edge set of a general bipartite graph is at most $O(n^2)$, where n is the number of vertices in the graph. However, if the input graph can be drawn in a plane without crossing edges (i.e., it is a planar graph), the size of the edge set is less than $3n$-6 [22]. Since biplanar graphs are intrinsically planar, the size of the edge set of biplanar graphs is less than $3n$-6. In fact, a biplanar graph is a tree, the size of whose edge set is n-1.

Theorem 2: The line graph of a biplanar graph is an interval graph.

Proof: Given a biplanar graph $G=(S, T, E)$, we will construct an interval model to represent the line graph of G. Since G is biplanar, we have a biplanar drawing of G on two horizontal lines. By preserving the orderings of the biplanar drawing, this drawing can be further rearranged to satisfy the following two properties (See Fig. 2 for example):

(1) Every vertex v of S and T has distinct $\chi(v)$ value and $1 \le \chi(v) \le |S|+|T|$. Hereafter $\chi(v)$ denotes the x-coordinate of a vertex v in the biplanar drawing of G.
(2) The integer set $\{\chi(u) | u \in N(v)$ and $d(u)=1\}$ consists of consecutive integers for every vertex $v \in S \cup T$.

The first property can be achieved by scaling the x-coordinates of vertices properly, and the second by packing the degree-one vertices which incident to the other same vertex.

According to the drawing, each vertex of G is labeled by using its x-coordinate, and an interval $[x\text{-o.5}, y\text{+0.5}]$ is created for each edge (x, y) in E (See Fig. 2 for example). The remainder is to show that the set of intervals represents the line graph of G. Suppose that two vertices are adjacent in $L(G)$ and the corresponding edges in G are $e_1=(u, v)$ and e_2. Without loss of generality, let $u \le v$ and the interval $[u\text{-o.5}, v\text{+0.5}]$ represents e_1. That implies that e_2 must be incident with u or v; and the interval created for e_2 is $[z, u\text{+0.5}]$ or $[v\text{-o.5}, z]$, both of which overlap with $[u\text{-o.5}, v\text{+0.5}]$.

On the other hand, given any pair of overlapped intervals $[x_1, y_1]$ and $[x_2, y_2]$ in the set, we have $x_1 \le x_2 < y_1 \le y_2$. We claim that the corresponding two edges in G are incident

with a common vertex; that also indicates the corresponding two vertices are adjacent in $L(G)$. Otherwise, the edges are not incident with a common vertex, and the created intervals for the edges must be $[x_1, y_1]$ and $[x_2, y_2]$ and $x_1 < y_1 < x_2 < y_2$ according to the two properties of the drawing. A contradiction occurs.

Thus, the set of intervals represents the line graph of G, and $L(G)$ is an interval graph. ■

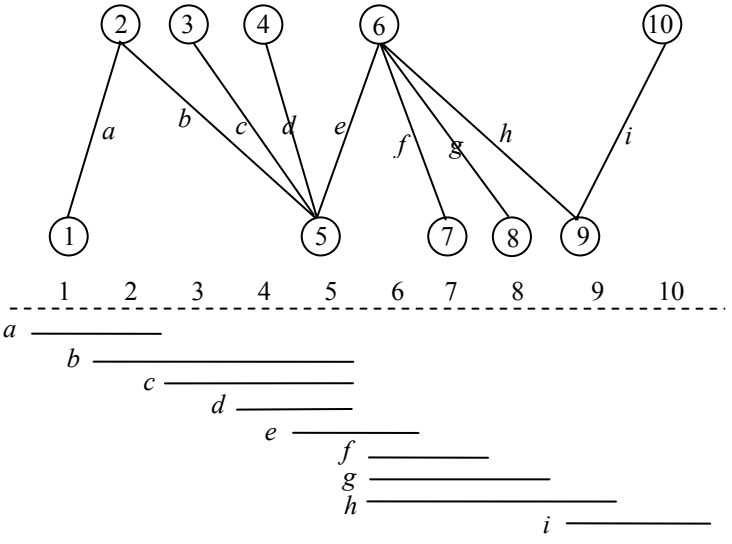


Fig. 2. A biplanar graph G with the interval model of its line graph

Based on above theorem, an algorithm for the max-edge-coloring problem is described as follows.

Algorithm AMEC:

Input: a biplanar graph $G = (S, T, E)$.

Output: an edge colorings $\{E_1, E_2, E_3, \ldots, E_z\}$ of G and the value $\sum_{i=1}^{z} \max \{w(e_k) | e_k \in E_i\}$, where $w(e_k)$ is the weight of e_k.

Step 1: Constructing the line graph $L(G)$ from G.

Step 2: Finding an interval model for $L(G)$ by applying an interval graph recognition algorithm.

Step 3: Finding a vertex coloring $\{C_1, C_2, C_3, \ldots, C_z\}$ of $L(G)$ by applying a 2-approximation algorithm for solving the max-coloring problem on interval graphs.

Step 4: Constructing an edge colorings $\{E_1, E_2, E_3, \ldots, E_z\}$ of G from $\{C_1, C_2, C_3, \ldots, C_z\}$ and compute the value

$$\sum_{i=1}^{z} \max \{w(e_k) | e_k \in E_i\}.$$

The performance guarantee and complexity of algorithm AMEC are described in the following theorems.

Theorem 3: When the input graph is biplanar, AMEC is a 2-approximation algorithm for the max-edge-coloring problem.

Proof: Since an edge coloring of an graph G corresponds to a vertex coloring of its line graph $L(G)$, the *max-edge-coloring* problem of G can be transformed to the *max-coloring* problem of $L(G)$. Since $L(G)$ is an interval graph (by Theorem 2), we obtain a 2-approximation algorithm for max-edge-coloring problem of G by applying Pemmaraju *et al* 's 2-approximation algorithm for the max-coloring problem on interval graphs [28]. Finally, we conclude that Algorithm AMEC is a 2-approximation algorithm for the max-edge-coloring problem. ■

Theorem 4: When the input graph is biplanar, AMEC requires $O(n^2)$ time, where n is the size of the vertex set of the input graph.

Proof: The time complexity of AMEC is discussed as follows. Step 1 requires $O(n^2)$ time because the step compares at most every pair in the edge set (whose size of edge set is $|E|=O(n)$) of G for constructing the edge set of $L(G)$. Step 2 can be implemented in $O(n^2)$ time if we apply Booth and Lueker's linear-time recognition algorithm for interval graphs [30]. Step 3 takes $O(n\log n)$ time if we apply Pemmaraju *et al* 's 2-approximation algorithm for the max-coloring problem on interval graphs [28]. Finally, Step 4 can be implemented in $O(n)$ time. The next theorem makes a summary. ■

4 The Splitting Chromatic Max-Edge-Coloring Problem Is NP-Complete When the Input Graph Is Restricted to Biplanar Graphs

When the input graph is restricted to biplanar graphs, we will prove that the SCMECP is NP-complete by transforming from the partition problem: Given a finite set A and a weight $s(a) \in Z^+$ for each $a \in A$, *the partition problem* is to ask whether there is a subset $A' \subseteq A$ such that

$$\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a) \cdot$$

Theorem 5: The SCMECP is NP-complete even the input graph is restricted to biplanar graphs (The proof is omitted due to page limit).

5 Applications in Scheduling Data Redistribution

We also find applications in scheduling problems for data redistribution in parallel systems for these problems in this section.

5.1 Applications of MECP and CMECP

Parallel computing systems have been extensively adopted to resolve complex scientific problems efficiently. When processing various phases of applications, parallel systems normally exploit data distribution schemes to balance the system load and yield a better performance. Array redistribution is crucial for system performance because a specific array distribution may be appropriate for the current phase, but incompatible for the subsequent one. Many parallel programming languages thus support run-time primitives for rearranging a program's array distribution. Therefore developing efficient algorithms for array redistribution is essential for designing distributed memory compilers for those languages. While array redistribution is performed at run time, a trade-off occurs between the efficiency of the new data rearrangement for the coming phase and the cost of array redistributing among processors.

In irregular redistribution, messages of varying sizes are scheduled in the same communication step. Therefore, the largest size of message in the same communication step dominates the data transfer time required for this communication step.

A bipartite graph model will be introduced to represent data redistributions. Any data redistribution can be represented by a bipartite graph $G=(S, T, E)$, called a *redistribution graph*. Where S denotes source processor set, T denotes destination processor set, and each edge denotes a message required to be sent. For example, a Block-Cyclic(x) to Block-Cyclic(y) data redistribution from P processors to Q processors (denoted by $BC(x, y, P, Q)$) can be modeled by a bipartite graph $G_{BC(x, y, P, Q)}=(S, T, E)$ where $S=\{s_0, s_1, \ldots, s_{|s|-1}\}$ ($T=\{t_0, t_1, \ldots, t_{|t|-1}\}$) denotes the source processor set $\{p_0, p_1, \ldots, p_{|s|-1}\}$ (destination processor set$\{p_0, p_1, \ldots, p_{|t|-1}\}$) and we have $(s_i, t_j)\in E$ with weight w if source processor p_i has to send the amount of w data elements to destination processor p_j. For simplicity, we use $BC(x, y, P)$ to denote $BC(x, y, P, P)$. Fig. 3 depicts the a data redistribution pattern $BC(1, 4, 4)$, and its corresponding redistribution graph $G_{BC(1, 4, 4)}$ is shown in Fig. 4.

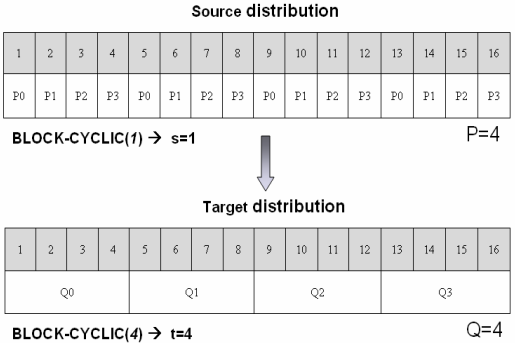


Fig. 3. A data redistribution pattern $BC(1, 4, 4)$

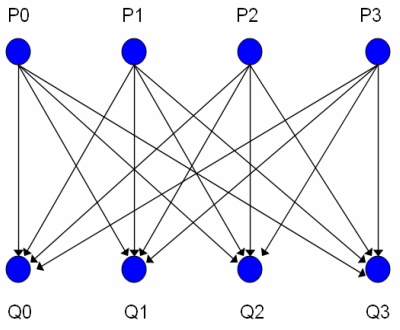


Fig. 4. The redistribution graph $G_{BC(1, 4, 4)}$ is a complete bipartite graph

Similarly, GEN_BLOCK data redistribution from P processors to Q processors (denoted by GB (P, Q)) can also be modeled by a bipartite graph $G_{GB(P, Q)}$=(S, T, E). For example, a GB(4, 4) with its redistribution graph $G_{GB(4, 4)}$ is depicted in Fig. 5 and 6.

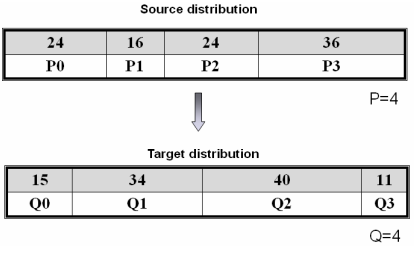


Fig. 5. GEN_BLOCK data redistribution GB(4, 4)

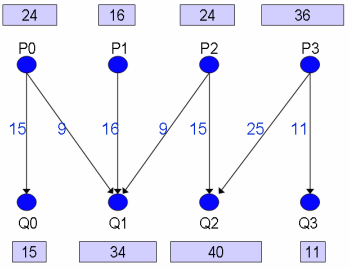


Fig. 6. A redistribution graph $G_{GB(4, 4)}$

Theorem 6: The redistribution graph of GEN-BLOCK is a biplanar graph (The proof is omitted due to page limit).

A set of conflict-free data communication can be represented by a matching of the given redistribution graph G. Thus, the data redistribution problem can be modeled as the MECP and CMECP.

5.2 Applications of SCMECP

Designing data redistribution scheduling algorithms for CMECP encounters a difficulty: shortening the overall communication time without increasing the number of communication steps at the same time. Unlike existing algorithms, Yu *et al* [31] presented an algorithm to partition large data segments into multiple small data segments and properly schedule them in different communication steps without increasing the number of total communication steps. For example, Fig. 7 depicts a redistribution graph G with a possible scheduling.

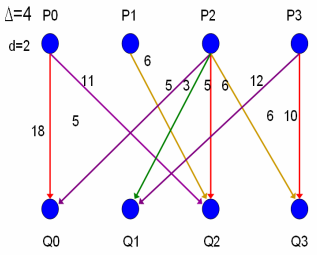


Fig. 7. A redistribution graph G with maximum degree $\Delta=4$

Since G is bipartite, it is well known that $\chi'(G)=\Delta(G)$ [22]. That indicates that the minimum number of required communication steps (colors) equals the maximum degree Δ of the given distribution graph G. Therefore, we at least need four communication steps for the data redistribution since $\chi'(G)=\Delta(G)=4$. In addition, the overall cost of the scheduling is 38 (See Table 1).

Table 1. Costs of the scheduling correspond to the edge coloring in Fig. 7

Step	1(red)	2(yellow)	3(green)	4(purple)	Total
Cost	18	6	3	11	38

Note that the cost of Step 1 (colored in red) is dominated by the data segment (with 18 data elements) from P_0 to Q_0. Suppose that we partition the data of the segment into two data segments (with 9 and 9 data elements respectively) and transmit them in different steps; then the cost required for Step 1 is reduced to 10 (Currently the step is dominated by the data segment from P_3 to Q_3). We can represent the kind of message partition by adding a new edge (P_0, Q_0) in the original redistribution graph and sharing weight with the old edge (P_0, Q_0). Similarly, we can partition other large data segments into multiple small data segments if the maximum degree of the resulting redistribution graph remains unchanging. After several data partitions, the overall communication cost can be reduced to 29 (or equivalently 76%) and the number of required communication step is still minimized (see Fig. 8 and Table 2).

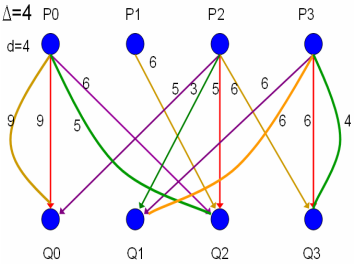


Fig. 8. The resulting redistribution graph after partitioning long data segments

Table 2. Costs of the scheduling after partitioning long data segments

Step	1(red)	2(yellow)	3(green)	4(purple)	Total
Cost	9	9	5	6	29

Evidently, the above technique can be modeled by the SCMECP. We have shown the SCMECP is NP-complete even when the input graph is biplanar by Theorem 5.

6 Conclusions

In this work, we have designed the first 2-approximmation algorithm for the MECP on biplanar graphs. We also proved that the SCMECP is NP-complete even when the input graph is restricted to biplanar graphs. These two problems find applications in scheduling data redistribution on parallel computer systems. The authors believe that these newly defined graph problems deserve serious attention and can be applied to tackle more practical problems in diverse fields.

References

[1] Bandera, G., Zapata, E.L.: Sparse Matrix Block-Cyclic Redistribution. In: Proceeding of IEEE Int'l. Parallel Processing Symposium (IPPS'99), San Juan, Puerto Rico (April 1999)
[2] Desprez, F., Dongarra, J., Petitet, A.: Scheduling Block-Cyclic Data redistribution. IEEE Trans. on PDS 9(2), 192–205 (1998)
[3] Hsu, C.-H, Bai, S.-W, Chung, Y.-C, Yang, C.-S: A Generalized Basic-Cycle Calculation Method for Efficient Array Redistribution. IEEE Transactions on Parallel and Distributed Systems 11(12), 1201–1216 (2000)
[4] Hsu, C.-H, Yang, D.-L., Chung, Y.-C., Dow, C.-R.: A Generalized Processor Mapping Technique for Array Redistribution. IEEE Transactions on Parallel and Distributed Systems 12(7), 743–757 (2001)
[5] Guo, M.: Communication Generation for Irregular Codes. The Journal of Supercomputing 25(3), 199–214 (2003)
[6] Guo, M., Nakata, I.: A Framework for Efficient Array Redistribution on Distributed Memory Multicomputers. The Journal of Supercomputing 20(3), 243–265 (2001)

[7] Guo, M., Nakata, I., Yamashita, Y.: Contention-Free Communication Scheduling for Array Redistribution. Parallel Computing 26(8), 1325–1343 (2000)
[8] Guo, M., Nakata, I., Yamashita, Y.: An Efficient Data Distribution Technique for Distributed Memory Parallel Computers. JSPP'97, pp. 189–196 (1997)
[9] Guo, M., Pan, Y., Liu, Z.: Symbolic Communication Set Generation for Irregular Parallel Applications. The Journal of Supercomputing 25, 199–214 (2003)
[10] Kalns, E.T., Ni, L.M.: Processor Mapping Technique Toward Efficient Data Redistribution. IEEE Trans. on Parallel and Distributed Systems 6(12) (1995)
[11] Kaushik, S.D., Huang, C.H., Ramanujam, J., Sadayappan, P.: Multiphase data redistribution: Modeling and evaluation. In: Proceeding of IPPS'95, pp. 441–445 (1995)
[12] Lee, S., Yook, H., Koo, M., Park, M.: Processor reordering algorithms toward efficient GEN_BLOCK redistribution. In: Proceedings of the ACM symposium on Applied computing (2001)
[13] Lim, Y.W., Bhat, P.B, Prasanna, V.K: Efficient Algorithms for Block-Cyclic Redistribution of Arrays. Algorithmica 24(3-4), 298–330 (1999)
[14] Park, N., Prasanna, V.K., Raghavendra, C.S.: Efficient Algorithms for Block-Cyclic Data redistribution Between Processor Sets. IEEE Transactions on Parallel and Distributed Systems 10(12), 1217–1240 (1999)
[15] Petitet, A.P., Dongarra, J.J.: Algorithmic Redistribution Methods for Block-Cyclic Decompositions. IEEE Trans. on PDS 10(12), 1201–1216 (1999)
[16] Prylli, L., Touranchean, B.: Fast runtime block cyclic data redistribution on multiprocessors. Journal of Parallel and Distributed Computing 45, 63–72 (1997)
[17] Ramaswamy, S., Simons, B., Banerjee, P.: Optimization for Efficient Data redistribution on Distributed Memory Multicomputers. Journal of Parallel and Distributed Computing 38, 217–228 (1996)
[18] Wakatani, A., Wolfe, M.: Optimization of Data redistribution for Distributed Memory Multicomputers. short communication, Parallel Computing 21(9), 1485–1490 (1995)
[19] Wang, H., Guo, M., Wei, D.: Divide-and-conquer Algorithm for Irregular Redistributions in Parallelizing Compilers. The Journal of Supercomputing 29(2) (2004)
[20] Wang, H., Guo, M., Chen, W.: An Efficient Algorithm for Irregular Redistribution in Parallelizing Compilers. In: Proceedings of International Symposium on Parallel and Distributed Processing with Applications (2003)
[21] Yook, H.-G., Park, M.-S.: Scheduling GEN_BLOCK Array Redistribution. In: Proceedings of the IASTED International Conference Parallel and Distributed Computing and Systems (November 1999)
[22] Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. Macmillan, London (1976)
[23] Cole, R., Hopcroft, J.: On edge-coloring bipartite graphs. SIAM J. Comput. 11, 540–546 (1982)
[24] Yu, C.W., Chen, G.H.: Efficient parallel algorithms for doubly convex-bipartite graphs. Theoretical Computer Science 147, 249–265 (1995)
[25] Eades, P., McKay, B.D., Wormald, N.C.: On an edge crossing problem. In: Proc. 9th Australian Computer Science Conference, pp. 327–334. Australian National University, Australian (1986)
[26] Tomii, N., Kambayashi, Y., Shuzo, Y.: On planarization algorithms of 2-level graphs, Papers of tech. group on electronic computers. IECEJ EC77-38, 1–12 (1977)
[27] Yu, C.W.: On the complexity of the maximum biplanar subgraph problem. Information Science 129, 239–250 (2000)

[28] Pemmaraju, S.V., Raman, R., Varadarajan, K.R.: Buffer minimization using max-coloring. In: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 562–571 (2004)
[29] Pemmaraju, S.V., Raman, R.: Approximation algorithms for the max-coloring problem. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1064–1075. Springer, Heidelberg (2005)
[30] Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. Comput. System Sci. 13, 335–379 (1976)
[31] Yu, C.W., Hsu, C.-H., Yu, K.-M., Lian, C.K., Chen, C.-I: Irregular Redistribution Scheduling by partitioning Messages. In: Srikanthan, T., Xue, J., Chang, C.-H. (eds.) ACSAC 2005. LNCS, vol. 3740, pp. 295–309. Springer, Heidelberg (2005)

Quantitative Analysis of Multi-hop Wireless Networks Using a Novel Paradigm

Chang Wu Yu

Department of Computer Science and Information Engineering
Chung Hua University, Taiwan, R.O.C
cwyu@chu.edu.tw

Abstract. Random geometric graphs (RGG) contain vertices whose points are uniformly distributed in a given plane and an edge between two distinct nodes exists when their distance is less than a given positive value. RGGs are appropriate for modeling multi-hop wireless networks consisting of n mobile devices with transmission radius r unit length that are independently and uniformly distributed randomly in an area. This work presents a novel paradigm to compute the subgraph probability in RGGs. In contrast to previous asymptotic bounds or approximation, the closed-form formulas we derived herein are fairly accurate and of practical value. The proposed paradigm can be used to make quantitative analyzes on the fundamental properties of multi-hop wireless networks.

Keywords: Random geometric graphs, subgraph counting, subgraph probability.

1 Introduction

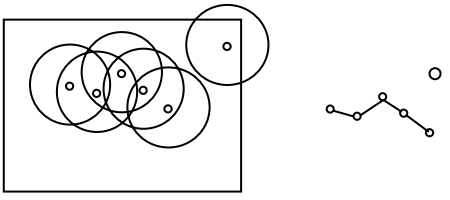
A *geometric graph* $G=(V, r)$ consists of nodes placed in a two-dimensional space R^2 and edge set $E=\{(i,j) \mid d(i,j)\leq r$, where $i, j\in V$ and $d(i,j)$ denotes the Euclidian distance between node i and node $j\}$. Let $X_n=\{x_1, x_2, \ldots, x_n\}$ be a set of independently and uniformly distributed random points. Here, $\Psi(X_n, r, A)$ is used to denote the *random geometric graph* (RGG) [14] of n nodes on X_n with radius r and placed in an area A. RGGs consider geometric graphs on random point configurations. Applications of RGGs include communications networks, classification, spatial statistics, epidemiology, astrophysics and neural networks [14].

A RGG $\Psi(X_n, r, A)$ is appropriate for modeling an ad hoc network $N=(n, r, A)$ consisting of n mobile devices with transmission radius r unit length that are independently and uniformly distributed randomly in an area A. When each vertex in $\Psi(X_n, r, A)$ represents a mobile device, each edge connecting two vertices represents a possible communication link as they are within the transmission range of each other. Fig. 1 displays a random geometric graph and its representing network. In the example, area A is a rectangle used to model the deployed area such as a meeting room. Area A, however, can be a circle, or any other shape, and even infinite space.

RGGs are different from conventional *random graphs* [2, 8, 13]. One random graph has two parameters n and p, where n denotes the number of nodes and p represents the

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 362–374, 2007.

probability of the existence of each possible edge. Edge occurrences in the random graph are independent of each other, unlike the case in RGGs. Therefore the fruitful results of random graphs cannot be directly applied to RGGs.



(a) (b)

Fig. 1. (a) An ad hoc network $N=(6, r, A)$, where A is a rectangle. (b) Its associated random geometric graph $\Psi(X_6, r, A)$.

Counting the number of subgraphs in RGGs is of priority concern in quantitatively analyzing wireless ad hoc networks [11, 12, 19, 22-25]. For example, the IEEE 802.11 CSMA/CA protocol suffers from the hidden and the exposed terminal problem [19]. The hidden terminal problem is caused by concurrent transmissions of two nodes that cannot sense each other, but transmit to the same destination. Such two terminals are referred to here a *hidden-terminal pair*. Hidden-terminal pairs in an environment seriously results in garbled messages and increases communication delay, thus degrading system performance [11, 19]. A hidden-terminal pair can be represented by a pair of edges $\{(x, y), (x, z)\}$ of $G=(V, E)$ such that $(x, y)\in E$ and $(x, z)\in E$, but $(y, z)\notin E$. In graph terms, such a pair of edges is an induced subgraph p_2 that is a path of length two (See Fig. 2). Counting the occurrences of p_2 in a given RGG helps counting the number of hidden-terminal pairs in a network. The exposed terminal problem is due to prohibiting concurrent transmissions of two nodes that sense each other, but can transmit to different receivers without conflicts, resulting in unnecessary reduction in channel utilization and throughput. These nodes are referred to here as an *exposed-terminal set*. Similarly, the problem can be modeled as a subgraph M of $G=(V, E)$ with four vertices $\{x, y, z, w\}\subseteq V$ such that $\{(x, y), (y, z), (z, w)\}\subseteq E$, but $(x, z)\notin E$ and $(y, w)\notin E$ (See Fig. 2).

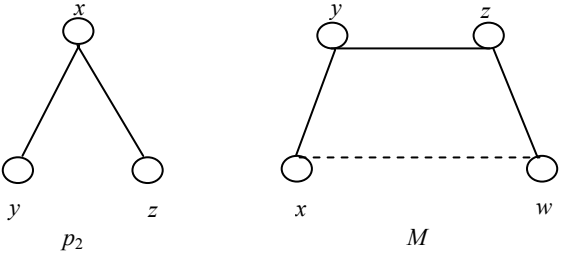


Fig. 2. Subgraphs of hidden-terminal pair p_2 and exposed-terminal set M

The *subgraph probability* of a labeled subgraph G_x in RGGs, denoted $\Pr(G_x)$, is the probability of the occurrence of G_x as an induced labeled subgraph in RGGs. A subgraph probability of a RGG G_x is *computable* if a closed-form function can be obtained for $\Pr(G_x)$. Counting the number of subgraphs of RGGs has received considerable attention [14]. Penrose demonstrated that, for arbitrary feasible connected Γ with k vertices, the number of induced subgraphs isomorphic to Γ satisfies a Poisson limit theorem and a normal limit theorem [14]. To our knowledge, results of previous studies are all asymptotic or approximate without accurate closed-form functions.

To our knowledge, this work presents a novel paradigm for computing the subgraph probability in RGGs for the first time. With the derived results, counting the numbers of some specific induced subgraphs in RGGs and their applications are discussed. In contrast to previous asymptotic bounds or approximation, the closed-form formul derived here are fairly accurate. The proposed paradigm can be used to determine the fundamental properties of multi-hop wireless networks.

Our definition of random geometric graphs $\Psi(X_n, r, A)$ is different from those of Poisson point process [1, 7], which assume that the distribution of n points (vertices) on a possibly infinite plane follows a Poisson distribution with parameter λ (the given density). In Poisson point process, the number of vertices can only be a random number rather than a tunable parameter. In practice, however, some wireless network modeling requires a fixed input n or a finite deployed area. Torus convention models the network topology so that nodes near the border are considered as close to nodes at the opposite border; they are allowed to establish links as well. Here, *torus convention* is adopted to cope with border effects [1, 10].

The rest of the paper is organized as follows. Section 2 introduces definitions and notations are introduced. Section 3 then briefly surveys pertinent literature on RGGs. Nest, Section 4 present a novel paradigm for computing the subgraph probability of RGGs. Conclusions are finally drawn in Section 5 along with areas for future research.

2 Definitions and Notations

A *graph* $G=(V, E)$ consists of a finite nonempty vertex set V and edge set E of unordered pairs of distinct vertices of V. A graph is *simple* if it has no loops and no two of its links join the same pair of vertices. A graph $G=(V, E)$ is labeled when the $|V|$ vertices are distinguished from one another by names such as $v_1, v_2, \ldots, v_{|V|}$. Two labeled graphs $G=(V_G, E_G)$ and $H=(V_H, E_H)$ are identical, denoted by $G=H$ if $V_G= V_H$ and $E_G=E_H$. A graph $H=(V_H, E_H)$ is a *subgraph* of $G=(V_G, E_G)$ if $V_H \subseteq V_G$ and $E_H \subseteq E_G$. Suppose that V' is a nonempty subset of V. The subgraph of $G_W=(V, E)$ whose vertex set is $V' \subseteq V$ and whose edge set is the set of those edges of G that have both ends in V' is called the subgraph of G_W *induced* by V', denoted by $G_{W,V'}$. The *subgraph probability* of

RGGs is defined formally as follows. Let $\Omega=\{G_1, G_2, \ldots, G_k\}$ represent every possible labeled simple graphs of $\Psi(X_n, r, A)$, where $k=2^{\binom{n}{2}}$. When G_x is a labeled subgraph in Ω, we use $\Pr(G_x)$ to denote the probability of the occurrence of G_x in $\Psi(X_n, r, A)$. Given a graph $G=(S, E)$, we define

$$\Pr(G)= \sum_{\forall G_w \in \Omega \text{ and } G = G_{w,s}} \Pr(G_w), \text{ when } 1 \le w \le k.$$

The *size* of any set S is denoted by $|S|$. The *degree* of a vertex v in graph G is the number of edge incident with v. A *leaf* is a vertex of degree 1. The notation $\binom{n}{m}$ denotes the number of ways to select m from n distinct objects. An edge e of G is said to be *contracted* if it is deleted and its ends are identified. A subgraph of G is said to be *contracted* if all its edges are contracted successively in any order. Two subgraphs are *disjoint* if their edge set are disjoint. An induced subgraph that is a path of length i is denoted by p_i. Similarly, an induced subgraph that is a cycle of length i is denoted by c_i; c_3 is often called a *triangle*. A set of vertices is *independent* if no two of them are adjacent. An induced subgraph which is an independent set of size i is denoted by I_i. A *complete graph* is a simple graph whose vertices are pairwise adjacent; the unlabeled complete graph with n vertices is denoted K_n. A *tree* is a connected acyclic graph. In a graph $G=(V, E)$, a set $S \subseteq V$ is a *dominating set* if every vertex not in S has a neighbor in S. The domination number $\gamma(G)$ is the minimum size of a dominating set in G. The notational conventions used in the paper can be found in [3].

3 Related Work in RGG

To the best of our knowledge, previous results on RGGs are all asymptotic and approximate. We summary related results as follows. A book written by Penrose [14] provides and explains the theory of random geometric graphs. Graph problems considered in the book include subgraph and component counts, vertex degrees, cliques and colorings, minimum degree, the largest component, partitioning problems, and connectivity and the number of components.

For n points uniformly randomly distributed on a unit cube in $d \ge 2$ dimensions, Penrose [17] showed that the resulting geometric random graph G is k-connected and G has minimum degree k at the same time when $n \to \infty$. In [4, 5], Díaz *et al.* discussed many layout problems including minimum linear arrangement, cutwidth, sum cut, vertex separation, edge bisection, and vertex bisection in random geometric graphs. In [6], Díaz *et al.* considered the clique or chromatic number of random geometric graphs and their connectivity.

Some results of RGGs can be applied to the connectivity problem of ad hoc networks. In [18], Santi and Blough discussed the connectivity problem of random geometric graphs $\Psi(X_n, r, A)$, where A is a d-dimensional region with the same length

size. In [1], Bettstetter investigated two fundamental characteristics of wireless networks: its minimum node degree and its k-connectivity. In [7], Dousse *et al.* obtained analytical expressions of the probability of connectivity in the one dimension case. In [9], Gupta and Kumar have shown that if

$$r=\sqrt{\frac{\log n + c(n)}{\pi n}},$$

then the resulting network is connected with high probability if and only if $c(n)\to\infty$. In [20], Xue and Kumar have shown that each node should be connected to $\Theta(\log n)$ nearest neighbors in order that the overall network is connected.

Recently, Yen and Yu have analyzed link probability, expected node degree, and expected coverage of MANETs [22]. In [21], Yang has obtained the limits of the number of subgraphs of a specified type which appear in a random graph.

4 A Paradigm for Computing Subgraph Probability

In the section, we develop a paradigm for exactly computing subgraph probability of RGGs. First, we deal with subgraphs with three vertices. Then a great deal of subgraph probabilities can be computed by the paradigm.

For simplicity, we always assume that A is sufficiently large to properly contain a circle with radius r in a $\Psi(X_n, r, A)$ throughout the paper; that implies $\pi r^2 \le |A|$. In the paper, notation E_i (E_i') denotes the event of the occurrence (absence) of edge e_i.

4.1 Base Subgraphs with Their Probabilities

Since we adopt torus convention to avoid border effects, single-edge probability in RGG is obtained trivially and listed below.

Theorem 1: We have $\Pr(e_j)=\pi r^2/|A|$, for an arbitrary edge $e_j=(u, v)$ and $u\neq v$, in a $\Psi(X_n, r, A)$.

The following theorem shows that the occurrences of arbitrary pairwise edges in RGGs are independent even if they share one end vertex. By Theorem 1 and 2, we obtain the probability of two-edge subgraphs immediately.

Theorem 2 [22]: For arbitrary two distinct edges $e_i=(u, v)$ and $e_j=(w, x)$ in a $\Psi(X_n, r, A)$, we have $\Pr(e_ie_j)=\Pr(e_i)\Pr(e_j)$.

Corollary 3: For arbitrary two distinct edges $e_i=(u, v)$ and $e_j=(w, x)$ in a $\Psi(X_n, r, A)$, we have $\Pr(e_ie_j)= (\pi r^2/|A|)^2$.

Theorem 2 does not imply that the occurrences of more edges in RGGs are also mutually independent. In fact, the next theorem shows their dependence.

Theorem 4 [24]: The occurrences of arbitrary three distinct edges in a $\Psi(X_n, r, A)$ are dependent.

In the subsection, we consider the eight labeled subgraphs with three vertices as *base subgraphs*, the probabilities of which will be used to form a basis for computing the probability of larger subgraphs later. Based on the number of edges included, subgraphs of three vertices can be classified into four groups: a triangle (c_3), an induced path of length two (p_2), an edge with an isolated vertex (p_1+I_1), and three isolated vertices (I_3) (See Fig. 3).

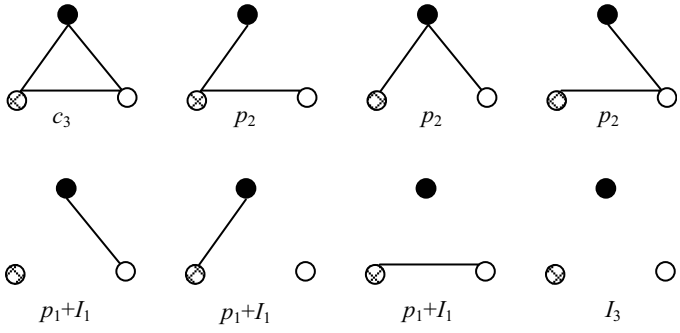


Fig. 3. Eight base subgraphs

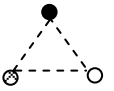
All these eight subgraph probabilities in a $\Psi(X_n, r, A)$ have been computed by using basic geometric techniques [24, 25]. Table 1 summaries their results.

Table 1. Probabilities of subgraphs with three vertices or less in a RGG

Notation	p_1	E^2	c_3	p_2	E^1+I_1	I_3
G						
Pr(G)	$\pi r^2/\|A\|$	$(\pi r^2/\|A\|)^2$	$\left(\pi-\frac{3\sqrt{3}}{4}\right)\pi r^4/\|A\|$	$\left(\frac{3\sqrt{3}}{4}\right)\pi r^4/\|A\|^2$	$\frac{\pi r^2}{\|A\|}\left(1-\frac{\pi r^2}{\|A\|}-\frac{3\sqrt{3}}{4}\frac{r^2}{\|A\|}\right)$	$1-\frac{\pi r^4}{\|A\|}-\frac{\frac{3\sqrt{3}}{4}}{\|A\|^2}$

4.2 A Paradigm for Computing Subgraph Probability of RGGs

In the paper, we adopt the following graph drawing convention. A solid line denotes an edge of G; a broken line denotes a possible edge between them; two vertices without a line denote a non-edge of G. A *class graph* G=(V, E_S, E_B) consists of a vertex set V and two disjoint edge sets E_S and E_B, where E_S (E_B) denotes a set of solid-line edge (broken-line edge) joining two vertices of V. For example, the following class graph denotes the set of eight base graphs depicted in Fig. 3.



The union of two class graphs G_1 and G_2, denoted G_1+G_2, is the set whose elements are exactly the graphs in either G_1 or G_2. The difference of two class graphs G_1 and G_2, denoted G_1-G_2, is the set containing exactly those elements in G_1 that are not in G_2. When G is a class graph, Pr(G) denotes the probability of the occurrence of $G_x \in G$ in $\Psi(X_n, r, A)$. If every element in G_1 is also in G_2, we have $G_1 \subseteq G_2$. Evidently, if $G_1 \subseteq G_2$ then $\Pr(G_1) \leq \Pr(G_2)$, and if G_1 is isomorphic to G_2 then $\Pr(G_1)=\Pr(G_2)$.

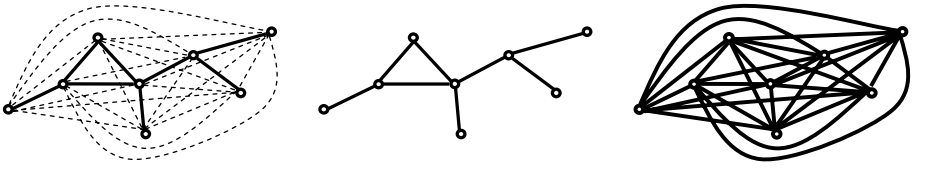
Note that we have $\Pr(E^2)=\Pr(c_3)+\Pr(p_2)$ in Table 1. This equation can be represented in two different forms: *the first-type* and *the second-type graph derivation* (depicted as follows).

Type I Type II

In fact, this first-type (second-type) graph derivation can be applied on any broken-line edge (non-edge) of any graph. Specifically, for any $e \in E_B$, we have $G(V, E_S, E_B)=G_1(V, E_S \cup \{e\}, E_B-\{e\})+G_2(V, E_S, E_B-\{e\})$. Or for any $e \notin E_S \cup E_B$, we have $G(V, E_S, E_B)=G_1(V, E_S, E_B \cup \{e\})-G_2(V, E_S \cup \{e\}, E_B)$ equivalently.

The main result of the subsection tries to partially answer the question: what kinds of subgraphs whose probabilities can be exactly computed in RGGs? The paradigm will show that great deals of subgraphs in RGGs are computable.

We can construct two graphs $\alpha(G)$ and $\beta(G)$ from a class graph $G=(V, E_S, E_B)$ such that $\alpha(G)=(V, E_S)$ and $\beta(G)=(V, E_S \cup E_B)$. Fig. 4 shows an example.



G $\alpha(G)$ $\beta(G)$

Fig. 4. A G (left) with its $\alpha(G)$ (middle) and $\beta(G)$ (right)

A class graph $G=(V, E_S, E_B)$ is a *Y-graph* if it can be constructed according to the following three rules:

R1: All base subgraphs are Y-graphs.
R2: G_1+G_2 and G_1-G_2 are also Y-graphs if G_1 and G_2 are two distinct Y-graphs.
R3: G is a Y-graph if $\alpha(\phi(G))$ is a tree and $\beta(\phi(G))$ is a complete graph, where $\phi(G)$ is obtained from G by the contractions of all the edges of each disjointed

Y-subgraphs G_c into a single vertex w adjacent to exactly those vertices that were previously not in G_c and adjacent to at least one vertex in G_c.

The following figure shows a Y-graph with its contraction of a base graph.

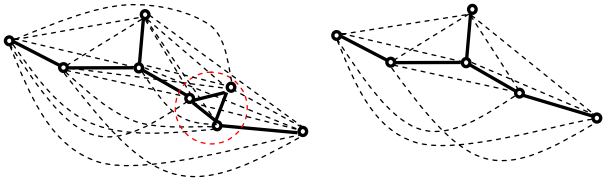


Fig. 5. A Y-graph G (left) and $\phi(G)$ (right), where $\alpha(\phi(G))$ is a tree and $\beta(\phi(G))$ is a complete graph

Theorem 5 [3]: A tree with two or more vertices has at least two leaves.

Given any Y-subgraph (if recognizable), its probability formula can be obtained (shown in the next theorem).

Theorem 6: The probability of a Y-subgraph in a $\Psi(X_n, r, A)$ is computable.

Proof: The probabilities of all base graphs are all computable as shown in Section 4.1. If $\Pr(G_1)$ and $\Pr(G_2)$ are given, we have $\Pr(G_1+G_2)=\Pr(G_1)+\Pr(G_2)$ and $\Pr(G_1-G_2)=\Pr(G_1)-\Pr(G_2)$. The rest is the case for those Y-subgraphs constructed by applying R3.

Suppose that G is constructed by applying R3. Let S be the size of vertex set of $\phi(G)$ We will prove that $\Pr(G)$ is computable by induction on S. When S=1, then G is computable since it is either a single vertex ($\Pr(G)$=1) or a base graph.

Since $\alpha(\phi(G))$ is a tree, $\alpha(\phi(G))$ must contain a leave w by Theorem 5. The removal of w together with the edges incident with it from G results in G^*, which is with S-1 vertices and then computable according to the induction hypothesis. If w is a vertex of G, we have only one solid line and S-1 broken lines incident to w due to the facts that $\alpha(\phi(G))$ is a tree and $\beta(\phi(G))$ is a complete graph; the existence of the unique solid line $e_j=(w, v)$, where v is a vertex in G^*, only depends on whether the distance between w and v is less than r; therefore we have $\Pr(G)=\Pr(G^*)\times\Pr(E_j)$. We conclude that G is computable.

Otherwise, w represents a Y-graph G_Y with size less than S. Since every vertex in G^* connects to every vertex in G_Y with broken lines except one solid line $e_j=(x, y)$, where $x(y)$ is a vertex in $G^*(G_Y)$. Similarly, we have $\Pr(G)=\Pr(G^*)\times\Pr(E_j)\times\Pr(G_Y)$; this also implies that G is computable. ■

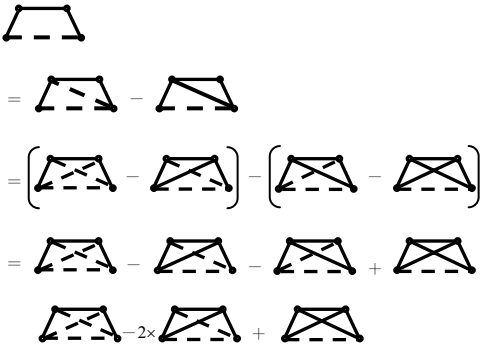
Theorem 7: Given a Y-graph $G=(V, E_S, E_B)$, if $\alpha(G)$ is a tree then $\Pr(G)=(\pi r^2/|A|)^{|V|-1}$.

Proof: (omitted). ■

Given a subgraph G, the paradigm try to compute its probability $\Pr(G)$ of a RGG by exploiting the following two steps:

(1) *Decomposing step*: Decompose G into a linear combination of class graphs: $c_1G_1+c_2G_1+\ldots c_kG_k$ such that $\beta(G_i)$ is a complete graph and c_i is a constant, for $1\le i\le k$, by recursively applying the first-type and the second-type graph derivations.
(2) *Manipulating step*: If G_i (for $1\le i\le k$) are Y-graphs, then we compute $\Pr(G_i)$ (for $1\le i\le k$) separately by using the probabilities of base graphs. Finally, Compute $\Pr(G)$ by manipulating $\Pr(G_i)$ (for $1\le i\le k$).

The following example shows how the proposed paradigm computes $\Pr(M)$ successfully (See M in Fig. 2). First, we present the decomposing step.



Next, we perform the manipulating steps on the three graphs shown in the above line:

(1) The first graph G (denoted by N) is a Y-graph because $\alpha(N)$ is a path (Note that a path is a kind of trees); therefore we have $\Pr(N)=(\pi r^2/|A|)^3$ by Theorem 7.
(2) The second graph (denoted by H) is a Y-graph because $\alpha(\phi(H))$ is a tree and $\beta(\phi(H))$ is a complete graph, where $\phi(H)$ is obtained from H by the contractions of a triangle c_3; then its probability can be obtained by following the procedure described in Theorem 6); that is, we have

$$\Pr(H)=\Pr(c_3)\times\Pr(E)=\left(\pi-\frac{3\sqrt{3}}{4}\right)\frac{\pi r^4}{|A|^2}\times\frac{\pi r^2}{|A|}.$$

(3) The last graph (denoted by B) is a Y-graph with

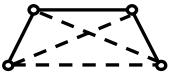
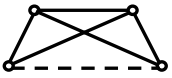
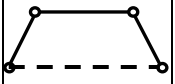
$$\Pr(B)=\left(\left(\pi-\frac{3\sqrt{3}}{4}\right)\frac{r^2}{|A|}\right)^2\times\frac{\pi r^2}{|A|}\ [24, 25].$$

In short, we have

$$\Pr(M)=\Pr(N)-2\times\Pr(H)+\Pr(B)=(\pi r^2/A)^3-2\times\left(\pi-\frac{3\sqrt{3}}{4}\right)\frac{\pi^2 r^6}{|A|^3}+\left(\pi-\frac{3\sqrt{3}}{4}\right)^2\frac{\pi r^6}{|A|^3}=\frac{27}{16}\frac{\pi r^6}{|A|^3}.$$

Table 2 lists the subgraphs and associated probabilities discussed above.

Table 2. Probabilities of some subgraphs with four vertices in a RGG

Notations	*N*	*H*	*B*	*M*
G				
Pr(*G*)	$(\pi r^2/\lvert A\rvert)^3$	$\left(\pi-\frac{3\sqrt{3}}{4}\right)\frac{\pi^2 r^6}{\lvert A\rvert^3}$	$\left(\pi-\frac{3\sqrt{3}}{4}\right)^2\frac{\pi^6 r^6}{\lvert A\rvert^3}$	$\frac{27\pi^6 r^6}{16\lvert A\rvert^3}$

4.3 Intractable Subgraphs and Their Linear Combinations

Some class graphs seem intractable to obtain their accurate probability formulas, even by using our paradigm. In the subsection, the relations and approximations of some of them are discussed.

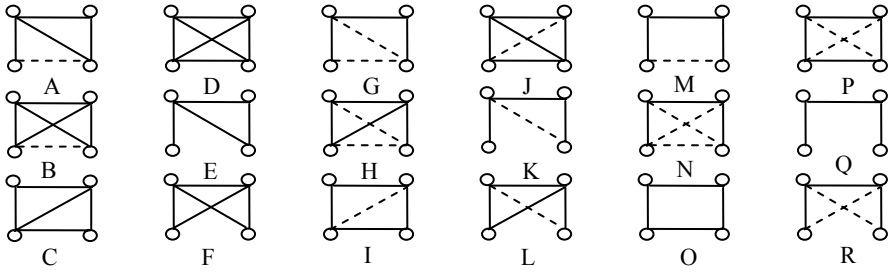


Fig. 6. Class graphs $G=(V, E_S, E_B)$ with $|V|=4$ consisting a three–solid-line path

Class graphs $G=(V, E_S, E_B)$ with $|V|=4$ consisting a three–solid-line path was first considered and depicted in Fig. 6. Note that each class graph in Fig. 6 can be viewed as a linear combination of other class graphs. Some of such linear combinations are shown in Table 3.

Note that {A, B, G, H, J, L, M, N} are Y-graphs and computable, but {C, D, E, F, I, K, O, P, Q, R} are not (so far). Moreover, if D is computable then F is computable (denoted by D→F), because D=J-F and J is computable (i.e., Pr(F)=Pr(J)-Pr(D)). This kind of dependence relations on {C, D, E, F, I, K, O, P, Q, R} is illustrated by using a digraph shown in Fig. 7.

Table 3. Some linear combinations of class graphs in Fig. 6

A=H-B	G=A+M	M
B=J=D+F	H	N
C=F	I=H-K=F+O	O=M-Q
D=J-F=B-F	J=B=D+F	P=B+I=N-R
E=L-F	K=E+Q=R-L	Q=M-O=K-E
F=C	L=H-B	R=L+K

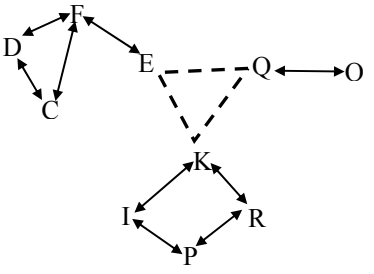


Fig. 7. The dependence digraph of {C, D, E, F, I, K, O, P, Q, R}, where A↔B denotes A→B and B→A

Note that the relation is a binary and equivalence (reflexive, symmetric, and transitive) relation. Therefore, we can partition {C, D, E, F, I, K, O, P, Q, R} into three disjoint subsets {{C, D, E, F}, {I, K, P, R}, {O, Q}} such that if any element in a subset is computable, then all elements in the subset are all computable. Moreover, if any two subsets are both computable, then the remainder is computable. This indicates that if D and O are computable, then all the ten class graphs are computable. Our paradigm, however, failed to show their computability so far. We devise their approximate formulas instead in the following two theorems; thereafter, we can obtain approximate formulas for these ten intractable graphs with the help of the linear combinations in Table 3.

Theorem 8[25]: For arbitrary four distinct nodes u, v, w, and x in a $\Psi(X_n, r, A)$, we have

$$\Pr(G_S=\mathrm{O}=c_4)\le\left(\frac{3\sqrt{3}}{4}\right)\frac{\pi r^6}{|A|^3},$$

where $S=\{u, v, w, x\}$.

Theorem 8: For arbitrary four distinct nodes u, v, w, and x in a $\Psi(X_n, r, A)$, we have $\Pr(G_S=\mathrm{D}=k_4)\le\left(\frac{\pi^2}{2}-\frac{7\sqrt{3}}{8}\pi+\frac{9}{8}\right)\frac{\pi r^6}{|A|^3}$, where $S=\{u, v, w, x\}$.

Proof: The four nodes $\{u, v, w, x\}$ need to be placed sufficiently near to each other in order to form a k_4 (See Fig. 8(a)). First, the three nodes $\{u, v, w\}$ must be a triangle c_3. The circle model for c_3 can be presented by intersections of three equal circles (See Fig. 8(b)).

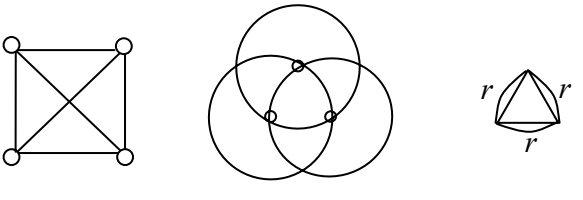


Fig. 8. (a) K_4. (b) Its circle model. (c) Reuleaux triangle.

Since the subgraph k_4 consists of a triangle c_3 and another nearby vertex x, we have $\Pr(G_S=k_4)\le\Pr(G_{S-\{x\}}=c_3)\times\Pr(x$ is near c_3 sufficiently). Note that $\Pr(x$ is near c_3 sufficiently) is the probability of putting the center of x in the common intersection of three equal circles; and the largest area of the common intersection, called Reuleaux triangle [26], is $(\frac{\pi-\sqrt{3}}{2})r^2$ (it is easily obtained by summing up the area of a equilateral triangle and three areas of a circular segment with opening angle $\pi/3$). Therefore, we have $\Pr(x$ is near c_3 sufficiently$)\le(\frac{\pi-\sqrt{3}}{2})r^2/|A|$. In a sequel, we have

$$\Pr(G_S=k_4)\le\left(\pi-\frac{3\sqrt{3}}{4}\right)\pi^4/|A|^2\times(\frac{\pi-\sqrt{3}}{2})r^2/|A|=\left(\frac{\pi^2}{2}-\frac{7\sqrt{3}}{8}\pi+\frac{9}{8}\right)\frac{\pi^6}{|A|^3}$$

by Table 1. ■

5 Conclusions

This work has proposed a novel paradigm for exactly computing numerous subgraph (i.e. Y-graphs) probabilities in RGGs. For some intractable graphs, an attempt is made to approximate their probability formulas. Future studies should devise an algorithm to recognize Y-graphs. Identifying a basis for generating a specific set of Y-graphs and non-Y-graphs is also of priority concern. We believe that the proposed paradigm can provide insight into additional fundamental properties of wireless networks.

References

1. Bettstetter, C.: On the minimum node degree and connectivity of a wireless multi-hop network. MobiHoc, pp. 80–91 (2002)
2. Bollobas, B.: Random Graphs. Academic Press, London (1985)
3. Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications, Macmillan Press (1976)
4. Díaz, J., Penrose, M.D., Petit, J., Serna, M.: Convergence theorems for some layout measures on random lattice and random geometric graphs. Combinatorics, Probability, and Computing 6, 489–511 (2000)
5. Díaz, J., Penrose, M.D., Petit, J., Serna, M.: Approximating layout problems on random geometric graphs. Journal of Algorithms 39, 78–116 (2001)
6. Díaz, J., Petit, J., Serna, M.: Random geometric problems on $[0, 1]^2$. In: Rolim, J.D.P., Serna, M.J., Luby, M. (eds.) RANDOM 1998. LNCS, vol. 1518, Springer, Heidelberg (1998)
7. Dousse, O., Thiran, P., Hasler, M.: Connectivity in ad-hoc and hybrid networks. Infocom (2002)
8. Erdös, P., Rényе, A.: On Random Graphs I. Publ. Math. Debrecen 6, 290–297 (1959)
9. Gupta, P., Kumar, P.R.: Critical power for asymptotic connectivity in wireless networks. Stochastic Analysis, Control, Optimization and Applications, pp. 547–566 (1998)
10. Hall, P.: Introduction to the Theory of Coverage Process. John Wiley and Sons, New York (1988)

11. Khurana, S., Kahol, A., Jayasumana, A.: Effect of hidden terminals on the performance of the IEEE 802.11 MAC protocol. In: Proceedings of Local Computer Networks Conference (1998)
12. Lee, S.-J., Gerla, M.: AODV-BR: Backup routing in Ad hoc Networks. IEEE Wireless Communications and Networking Conference 3, 1311–1316 (2000)
13. Palmer, E.M.: Graphical Evolution: An Introduction to the Theory of Random Graphs. John Wiley and Sons, New York (1985)
14. Penrose, M.D.: Random Geometric Graphs. Oxford University Press, Oxford (2003)
15. Penrose, M.D.: A strong low for the longest edge of the minimal spanning tree. The Annals of Probability 27(1), 246–260 (1999)
16. Penrose, M.D.: The longest edge of the random minimal spanning tree. The Annals of Applied Probability 7(2), 340–361 (1997)
17. Penrose, M.D.: On k-connectivity for a geometric random graph. Random structures and Algorithms 15(2), 145–164 (1999)
18. Santi, P., Blough, D.M.: The critical transmitting range for connectivity in sparse wireless ad hoc networks. IEEE Transactions on Mobile Computing 2(1), 25–39 (2003)
19. Tobagi, F., Kleinrock, L.: Packet switching in radio channels, Part II-The hidden terminal problem in carrier sense multiple access and the busy tone solution. IEEE Trans. Commun. COM-23(12), 1417–1433 (1975)
20. Xue, F., Kumar, P.R.: The number of neighbors needed for connectivity of wireless networks. Wireless Networks 10, 169–181 (2004)
21. Yang, K.J.: On the Number of Subgraphs of a Random Graph in $[0, 1]^d$, Unpublished D.Phil. thesis, Department of Statistics and Actuarial Science, University of Iowa (1995)
22. Yen, L.-H., Yu, C.W.: Link probability, network coverage, and related properties of wireless ad hoc networks. In: The 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pp. 525–527 (2004)
23. Yu, C.W., Wu, T.-K., Cheng, R.H.: A low overhead dynamic route repairing mechanism for mobile ad hoc networks. Computer Communications 30, 1152–1163 (2007)
24. Yu, C.W., Yen, L.-H., Cheng, Y.-M.: Computing subgraph probability of random geometric graphs with applications in wireless ad hoc networks. Tech. Rep., CHU-CSIE-TR-2004-005, Chung Hua University, R.O.C.
25. Yu, C.W., Yen, L.-H.: Computing subgraph probability of random geometric graphs: Quantitative analyses of wireless ad hoc networks. In: Wang, F. (ed.) FORTE 2005. LNCS, vol. 3731, pp. 458–472. Springer, Heidelberg (2005)
26. http://mathworld.wolfram.com/

Inverse Min-Max Spanning Tree Problem Under the Weighted Sum-Type Hamming Distance*

Longcheng Liu and Enyu Yao

Department of Mathematics, Zhejiang University, Hangzhou, China
llcly@126.com, eyyao@zju.edu.cn

Abstract. The inverse optimization problem is to modify the weight (or cost, length, capacity and so on) such that a given feasible solution becomes an optimal solution. In this paper, we consider the inverse min-max spanning tree problem under the weighted sum-type Hamming distance. For the model considered, we present its combinatorial algorithm that run in strongly polynomial times.

Keyword: Min-max spanning tree, Inverse problem, Hamming distance, Strongly polynomial algorithms.

1 Introduction

Let $G = (V, E, c)$ be a connected graph, where $V = \{1, 2, \ldots, n\}$ is the node set, $E = \{e_1, e_2, \ldots, e_m\}$ is the edge set and c is the edge cost vector defined on E. Let Γ denote the collection of all spanning tree of G. For a spanning tree $T \in \Gamma$, write $c^b(T) = \max\{c(e)|e \in T\}$ and call it the cost of T. The min-max spanning tree is to find a $T^* \in \Gamma$ such that $c^b(T^*) = \min\{c^b(T)|T \in \Gamma\}$. It is known that min-max spanning tree problem can be solved in strongly polynomial time[1].

Conversely, an inverse min-max spanning tree problem is to modify the edge cost vector as little as possible such that a given spanning tree becomes a min-max spanning tree. Yang et al. [2] showed that the inverse min-max spanning tree problem and the inverse maximum capacity path problem under l_1 and l_∞ norms are strongly polynomial time solvable, where the modification cost is measured by l_1 and l_∞ norms. In this paper, we consider the inverse min-max spanning tree problem under the weighted sum-type Hamming distance, in which we measure the modification cost by the weighted sum-type Hamming distance.

Let each edge e_i have an associated weight $w_i \geq 0$, and let w denote the edge weight vector. Let T^0 be a given spanning tree of graph G. Then for the inverse min-max spanning problem under the weighted sum-type Hamming distance, we look for a new cost vector $d = (d_1, d_2, \ldots, d_m)$ such that

(a) T^0 is a min-max spanning tree with respect to d;

(b) for each $e_i \in E$, $-l_i \leq d_i - c_i \leq u_i$, where $l_i, u_i \geq 0$ are given lower and upper modification bounds;

* Research supported by the National Natural Science Foundation of China (10371028).

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 375–383, 2007.

(c) $\sum_{i=1}^{m} w_i H(c_i, d_i)$ is minimized, where $H(c_i, d_i)$ is the Hamming distance between c_i and d_i, i.e., $H(c_i, d_i) = 0$ if $c_i = d_i$ and 1 otherwise.

In general, for an inverse combinatorial optimization problem, a feasible solution is given which is not optimal under the current parameter values, and it is required to modify some parameters with minimum modification cost such that the given feasible solution becomes an optimal solution. A lot of such problems have been well studied when the modification cost is measured by (weighted) l_1, l_2, and l_∞ norms. Readers may refer to the survey paper [3] and papers cited therein. Recently, inverse problems under the weighted Hamming distance also received attention. In fact the weighted sum-type Hamming distance represents the weighted number of modifications. It corresponds to the situation in which we might care about only whether the parameter of an arc is changed, but without considering the magnitude of its change as long as the adjustment is restricted to a certain interval. Noting that not like the l_1, l_2 and l_∞ norms which are all convex and continuous about the modification, the Hamming distance $H(\cdot, \cdot)$ is discontinuous and nonconvex, which make the known methods for l_1, l_2 and l_∞ norms unable to be applied directly to the problems under such distance measure.

He et al. [4] discussed the inverse minimum spanning tree problems under the weighted sum-type Hamming distance. For both unbounded and bounded cases, they presented strongly polynomial algorithms with a time complexity $O(n^3 m)$. Here n and m are the numbers of nodes and edges, respectively, in a given undirected network. He et al. [5] further discussed the inverse minimum spanning tree problems under the weighted bottleneck-type Hamming distance. For the unbounded case, they presented an algorithm with a time complexity $O(nm)$, and for the constrained case, they presented an algorithm with a time complexity $O(n^3 m \log m)$. Zhang et al. [6] considered the center location improvement problems under the weighted Hamming distance. For the bounded case, they showed that even under the unweighted sum-type Hamming distance, achieving an algorithm with a worst-case ratio $O(\log n)$ is strongly NP-hard, but under the weighted bottleneck-type Hamming distance, a strongly polynomial algorithm with a time complexity $O(n^2 \log n)$ is available. Yang et al. [7] discussed inverse sorting problems under the weighted sum-type Hamming distance. For both unbounded and bounded cases, they presented strongly polynomial algorithms. Liu et al. [8] discussed inverse maximum flow problems under the weighted Hamming distance, for both sum-type and bottleneck-type, they presented strongly polynomial algorithms. Liu et al. [9] discussed inverse minimum cut problem under the weighted bottleneck-type Hamming distance, they presented a strongly polynomial algorithm.

The paper is organized as follows. Section 2 contains some preliminary results. Sections 3 consider the problem under the weighted sum-type Hamming distance where we show that this problem can be solved by strongly polynomial algorithm. Some final remarks are made in Section 4.

In the following, for each edge set Ω we define $w^s(\Omega) = \sum_{e_i \in \Omega} w_i$ and use similar notation $c^s(\Omega)$ for vector c (here letters s stand for 'sum').

2 Preliminary Results

For the original min-max spanning tree problem, the following result is straightforward.

Lemma 1. *A spanning tree T of G is a min-max spanning tree under a cost vector c if and only if G becomes disconnected after deleting the edges whose costs are not less than $c^b(T)$.*

Now we consider the inverse min-max spanning tree problem under the weighted sum-type Hamming distance. The general inverse min-max spanning tree problem under the weighted sum-type Hamming distance can be formulated as follows.

$$\begin{aligned} &\min \sum_{i=1}^{m} w_i H(c_i, d_i) \\ &\text{s.t. } T^0 \text{ is a min} - \max \text{ spanning tree of } G(V, E, d); \\ &-l_i \le d_i - c_i \le u_i, \qquad 1 \le i \le m \ . \end{aligned} \tag{1}$$

Let T^* be a min-max spanning tree under the cost vector c, and assume $c^b(T^0) > c^b(T^*)$ for otherwise we need to do nothing.

Lemma 2. *There exists an optimal solution d^* of problem (1) such that $c^b(T^0) \ge d^{*b}(T^0)$.*

Proof. In fact, if $c^b(T^0) < d^{*b}(T^0)$, then we can construct a new cost vector $\overline{d}$ by the following way:

$$\overline{d}_i = \begin{cases} c^b(T^0), & \text{if } e_i \in T^0 \text{ and } d_i^* > c^b(T^0), \\ d_i^*, & \text{otherwise.} \end{cases}$$

It is clear that $\overline{d}^b(T^0) = c^b(T^0) < d^{*b}(T^0)$. By Lemma 1, the graph $G = (V, E, d^*)$ becomes disconnected after deleting the edges whose cost satisfy $d_i^* \ge d^{*b}(T^0)$. Hence the graph $G = (V, E, \overline{d})$ becomes disconnected after deleting the edges whose cost satisfy $\overline{d}_i \ge \overline{d}^b(T^0)$, which means T^0 is a min-max spanning tree of graph $G = (V, E, \overline{d})$, i.e., $\overline{d}$ is a feasible solution of problem (1).

However, by the definition of $\overline{d}$ and the definition of Hamming distance we have

$$\sum_{i=1}^{m} w_i H(c_i, d_i^*) \ge \sum_{i=1}^{m} w_i H(c_i, \overline{d}_i).$$

If $\sum_{i=1}^{m} w_i H(c_i, d_i^*) > \sum_{i=1}^{m} w_i H(c_i, \overline{d}_i)$, then d^* cannot be an optimal solution of problem (1), a contradiction. Hence, $\overline{d}$ is another optimal solution of problem (1), but it satisfies $\overline{d}^b(T^0) = c^b(T^0)$. The lemma holds.

Based on Lemma 2, the following result is straightforward:

Lemma 3. *There exists an optimal solution d^* of problem (1) satisfies:*
(a) $d_i^* = c_i$ *if* $c_i \geq d^{*b}(T^0)$ *and* $e_i \in E \setminus T^0$;
(b) $d_i^* \geq c_i$ *if* $c_i < d^{*b}(T^0)$ *and* $e_i \in E \setminus T^0$;
(c) $d_i^* = d^{*b}(T^0)$ *if* $c_i \neq d_i^*$ *and* $e_i \in T^0$.

Moreover, we have the following lemma:

Lemma 4. *There exists an optimal solution d^* of problem (1) satisfies:*
(a) $d_i^* \leq d^{*b}(T^0)$ *if* $d_i^* < c_i$;
(b) $d_i^* \geq d^{*b}(T^0)$ *if* $d_i^* > c_i$.

Proof. Suppose d^* is an optimal solution satisfies Lemma 3.

If $d_i^* < c_i$, then by the Lemma 3, we have $e_i \in T^0$, hence $d_i^* \leq d^{*b}(T^0)$, i.e., (a) holds.

Now let us consider (b). First, if $e_i \in T^0$, then by Lemma 3, we have $d_i^* = d^{*b}(T^0)$. Second, let us consider $e_i \in E \setminus T^0$. If (b) is not true, i.e., there exists an edge $e_k \in E \setminus T^0$ and $c_k < d_k^*$ such that $d_k^* < d^{*b}(T^0)$.

Define $\overline{d}$ as

$$\overline{d}_i = \begin{cases} c_i, & \text{if } i = k, \\ d_i^*, & \text{otherwise.} \end{cases}$$

Note that the difference between d^* and $\overline{d}$ is only on the edge e_k, so $d^{*b}(T^0) = \overline{d}^b(T^0)$. By Lemma 1, the graph $G = (V, E, d^*)$ becomes disconnected after deleting the edges whose cost satisfy $d_i^* \geq d^{*b}(T^0)$, combining with the fact that $\overline{d}_k = c_k < d_k^* < d^{*b}(T^0) = \overline{d}^b(T^0)$, we know the graph $G = (V, E, \overline{d})$ becomes disconnected after deleting the edges whose cost satisfy $\overline{d}_i \geq \overline{d}^b(T^0)$, which means that $\overline{d}$ is a feasible solution of problem (1). However, by the definition of $\overline{d}$ and the definition of Hamming distance we have

$$\sum_{i=1}^{m} w_i H(c_i, d_i^*) \geq \sum_{i=1}^{m} w_i H(c_i, \overline{d}_i).$$

If $\sum_{i=1}^{m} w_i H(c_i, d_i^*) > \sum_{i=1}^{m} w_i H(c_i, \overline{d}_i)$, then d^* cannot be an optimal solution of problem (1), a contradiction. Hence, $\overline{d}$ is another optimal solution of problem (1), but it satisfies $\overline{d}_k = c_k$. And by repeating the above procedure, we can conclude that there exists an optimal solution d^* of problem (1) such that $d_i^* \geq d^{*b}(T^0)$ if $d_i^* > c_i$ and $e_i \in E \setminus T^0$. From the above analysis, we know (b) holds.

Here we first give a range for the value $d^{*b}(T^0)$. First, by Lemma 2, we have $c^b(T^0) \geq d^{*b}(T^0)$. On the other hand, since there are lower bounds on the reduction of costs, the smallest possible value of $d^*(T^0)$ is $\underline{d} = \max\{c_i - l_i | e_i \in T^0\}$. So we have $\underline{d} \leq d^{*b}(T^0) \leq c^b(T^0)$. And by the definition of Hamming distance, we know the value $d^{*b}(T^0)$ must be one of the value in $P = \{\{c_i | e_i \in T^0\} \cup \{\underline{d}\} \cup \{c_i + u_i | e_i \in T^0\}\} \cap [\underline{d}, c^b(T^0)]$. Then we express the different values in P as: $p_1 > p_2 > \ldots > p_\eta$.

3 Problem Under the Weighted Sum-Type Hamming Distance

The problem considered in this section is the inverse min-max spanning tree problem under the weighted sum-type Hamming distance which can be formulated as problem (1).

Before we consider how to solve the problem (1) directly, let us consider a *restricted version* of the problem (1). That is, for a given value $p \in P$, we first consider how to make T^0 a min-max spanning tree under a cost vector d^p such that $d^{pb}(T^0) = p$, and d^p satisfies the bound restrictions and makes the modification cost minimum. We may call this restricted version of the inverse min-max spanning tree problem the *the inverse min-max spanning tree problem under the sum-type Hamming distance with value p*.

First, let $T^0(p) = \{e_i \in T^0 | c_i > p\}$, $\overline{T^0}(p) = \{e_i \in T^0 | c_i = p\}$. Clearly, for each edge $e_i \in T^0(p)$, we need to reduce their costs to let the maximum cost on T^0 be equal to p. Due to Lemma 3 for each edge $e_i \in T^0(p)$ we have $d_i^p = p$ and the associate objective value is $w^s(T^0(p)) = \sum_{e_i \in T^0(p)} w_i$. And by Lemma 3, for each edge $e_i \in \overline{T^0}(p)$ we do not need to change its cost.

Second, by Lemma 3, for each edge $e_i \in E \setminus T^0$ such that $c_i \geq p$, we do not need to change its cost.

Third, let $E(p) = \{e_i \in E | c_i < p\}$. Consider the graph $G(p) = (V, E(p))$. If $G(p)$ is not connected, we know that T^0 is already a min-max spanning tree with respect to the modified weight d^p and $d^{pb}(T^0) = p$, where $d_i^p = p$ for $e_i \in T^0(p)$ and $d_i^p = c_i$ for $e_i \in E \setminus T^0(p)$. And the objective value of problem (1) with respect to p is $w^s(T^0(p)) = \sum_{e_i \in T^0(p)} w_i$.

Thus we only need to consider the case that $G(p)$ is a connected graph. In this case, by Lemma 1, we need to increase the costs of some edges in graph $G(p)$ such that $G(p)$ becomes disconnected after deleting those edges. Now we introduce the following restricted minimum weight edge cut problem:

Restricted minimum weight edge cut problem(RMWECP)
Find an edge set $\Pi \subseteq E(p)$ in graph $G(p)$ such that

(1) for each edge $e_i \in \Pi$, $c_i + u_i \geq p$;
(2) $G(p)$ becomes disconnected after deleting all edges in Π;
(3) $\sum_{e_i \in \Pi} w_i$ is minimized.

Theorem 1. *If the RMWECP is feasible, then d^* defined as follows is one of the optimal solutions of the restricted version of the problem (1)*

$$d_i^* = \begin{cases} p, & \text{if } e_i \in T^0(p), \\ p, & \text{if } e_i \in \Pi^* \cap T^0, \\ c_i + u_i, & \text{if } e_i \in \Pi^* \cap (E \setminus T^0), \\ c_i, & \text{otherwise,} \end{cases} \tag{2}$$

and the associate objective value is $w^s(T^0(p)) + \sum_{e_i \in \Pi^*} w_i$*, where* Π^* *is a optimal solution of RMWECP. Otherwise, the restricted version of the problem* (1) *is infeasible.*

Proof. In the case that RMWECP is feasible, we first prove that d^* defined by (2) is a feasible solution of the restricted version of the problem (1). From the definition of $E(p)$ and the first and second constraints of RMWECP, we know the graph G becomes disconnected after deleting the edges whose costs are not less than p, which indicates T^0 is a min-max spanning tree of graph $G(V, E, d^*)$, and it is clear $-l_i \leq d_i^* - c_i \leq u_i$. Thus d^* defined by (2) is a feasible solution of the restricted version of the problem (1) and the associate objective value is $w^s(T^0(p)) + \sum_{e_i \in \Pi^*} w_i$.

Next we prove $w^s(T^0(p)) + \sum_{e_i \in \Pi^*} w_i$ is the minimum objective value, thus d^* is an optimal solution of the restricted version of the problem (1). If not, there exists an optimal solution $\overline{d}$ of the problem (1) such that

(a) $\overline{d}^b(T^0) = p$;
(b) $\sum_{i=1}^{m} w_i H(c_i, \overline{d}_i) < w^s(T^0(p)) + \sum_{e_i \in \Pi^*} w_i$.

Let

$$\Omega = \{e_i \in E \mid \overline{d}_i \neq c_i\} \ . \tag{3}$$

Then by the definition of Hamming distance, (b) is equivalent to

$$\sum_{i=1}^{m} w_i H(c_i, \overline{d}_i) = \sum_{e_i \in \Omega} w_i < w^s(T^0(p)) + \sum_{e_i \in \Pi^*} w_i \ . \tag{4}$$

Based on the above analysis, we can see that $T^0(p) \subseteq \Omega$, thus (4) is equivalent to

$$\sum_{e_i \in \Omega \setminus T^0(p)} w_i < \sum_{e_i \in \Pi^*} w_i \ . \tag{5}$$

Moreover, we say $\Omega \setminus T^0(p)$ is a feasible solution of RMWECP. In fact, it is clear $\Omega \setminus T^0(p) \subseteq E(p)$. Based on the analysis before, we know $\overline{d}_i > c_i$ for all $e_i \in \Omega \setminus T^0(p)$. And by Lemma 4 we know $\overline{d}_i \geq \overline{d}^b(T^0) = p$ for all $e_i \in \Omega \setminus T^0(p)$, which indicate $c_i + u_i \geq p$ for all $e_i \in \Omega \setminus T^0(p)$. At last we claim $G(p)$ becomes disconnected after deleting all edges in $\Omega \setminus T^0(p)$. If not, i.e., $G(V, E(p) \setminus \{\Omega \setminus T^0(p)\})$ is connected. Since $T^0(p) \cap E(p) = \emptyset$, we know $E(p) \setminus \{\Omega \setminus T^0(p)\} = \{E(p) \cup T^0(p)\} \setminus \Omega$, so $G(V, \{E(p) \cup T^0(p)\} \setminus \Omega)$ is connected. And thus $G(V, \{E(p) \cup T^0(p) \cup \{e_i \in E \mid c_i \geq p\}\} \setminus \{\Omega \cup \{e_i \in E \mid c_i \geq p\}\}) = G(V, E \setminus \{\Omega \cup \{e_i \in E \mid c_i \geq p\}\})$ is connected. Let $L = \{e_i \in E \mid \overline{d}_i \geq p\}$. By the above analysis we know $L \subseteq \{\Omega \cup \{e_i \in E \mid c_i \geq p\}\}$ and thus $G(V, E \setminus L)$ is connected. But since T^0 is a min-max spanning tree of $G(V, E, \overline{d})$, $G(V, E \setminus L)$ is disconnected by Lemma 1, a contradiction. So $G(p)$ becomes disconnected

after deleting all edges in $\Omega \setminus T^0(p)$. Hence $\Omega \setminus T^0(p)$ is a feasible solution of RMWECP. For Π^* is a optimal solution of RMWECP, we know

$$\sum_{e_i \in \Omega \setminus T^0(p)} w_i \geq \sum_{e_i \in \Pi^*} w_i, \tag{6}$$

which contradicts (5). So $w^s(T^0(p)) + \sum_{e_i \in \Pi^*} w_i$ is exactly the minimum objective value.

At the same time, the above analysis told us if $\overline{d}$ is a feasible solution of the restricted version of problem (1), then $\Omega \setminus T^0(p)$ is a feasible solution of RMWECP, where Ω is defined in (3). Thus if the RMWECP is infeasible, then the restricted version of problem (1) is infeasible too. □

Therefore, finding an optimal solution of the restricted version of problem (1) is equivalent to finding an optimal solution of RMWECP. To solve RMWECP in strongly polynomial time, we modify the graph $G(p) = (V, E(p))$ in the following way: the node set and the edge set are unchanged; and the value of each edge is set as

$$v_i = \begin{cases} w_i, & \text{if } e_i \in E(p) \text{ and } c_i + u_i \geq p, \\ W + 1, & \text{otherwise,} \end{cases} \tag{7}$$

where $W = \sum_{e_i \in E(p)} w_i$.

Definition 1. *Minimum value cut problem in graph is to find a set of edges whose deletion make the graph disconnected and the sum of values of edges in the set is minimized.*

The minimum value cut problem can be solved in $O(n^3)$ time [10], where n is the number of the nodes in the graph.

Theorem 2. *Let Π^* be a minimum value cut of $G(V, E(p), v)$ with a value $v^s(\Pi^*)$.*

(1) If $v^s(\Pi^*) \leq W$, then Π^* must be the optimal solution of RMWECP.
(2) If $v^s(\Pi^*) > W$, then RMWECP has no feasible solution.

Proof. (1) First, if $v^s(\Pi^*) \leq W$, then $v_i = w_i$ for all $e_i \in \Pi^*$, i.e., for each $e_i \in \Pi^*$, $e_i \in E(p)$ and $c_i + u_i \geq p$. And it is clear $G(p)$ becomes disconnected after deleting all edges in Π^*. So, Π^* is a feasible solution of RMWECP.

Moreover, it is easy to see that Π^* is an optimal solution of RMWECP. If not, suppose there exists an edge set Π' which is feasible to RMWECP, and $w^s(\Pi') < w^s(\Pi^*)$. Then from (7), we have

$$v^s(\Pi') = \sum_{e_i \in \Pi'} w_i = w^s(\Pi') < w^s(\Pi^*) = \sum_{e_i \in \Pi^*} w_i = v^s(\Pi^*),$$

which contradicts the fact that Π^* is a minimum value cut of $G(V, E(p), v)$.

(2) Suppose that $v^s(\Pi^*) > W$ but RMWECP has a feasible solution Π'. From (7), we know that $v_i = w_i$ for all $e_i \in \Pi'$. It implies that the value of Π' satisfies $v^s(\Pi') < W$ (as $W = \sum_{e_i \in E(p)} w_i$), which contradicts the fact that Π^* is a minimum value cut of $G(V, E(p), v)$ with a value $v^s(\Pi^*) > W$. □

Now we are ready to give a full description of an algorithm to solve the restricted version of the problem (1).

Algorithm 1

Step 1. For the graph $G = (V, E, c)$, the given spanning tree T^0 and the given value p, determine the graph $G(p) = (V, E(p))$. If $G(p)$ is not connected, stop and output an optimal solution d^* of the restricted version of the problem (1) as

$$d_i^* = \begin{cases} p, & \text{if } e_i \in T^0(p), \\ c_i, & \text{otherwise,} \end{cases}$$

and the associated optimal value $w^s(T^0(p))$. Otherwise go to Step 2.

Step 2. Construct graph $G(V, E(p), v)$ according to formula (7). Find a minimum value cut Π^* of the graph $G(V, E(p), v)$. If the value of the minimum cut satisfies $v^s(\Pi^*) > W$, then the restricted version of the problem (1) has no feasible solution, stop. Otherwise, go to Step 3.

Step 3. Output an optimal solution d^* of the restricted version of the problem (1) as

$$d_i^* = \begin{cases} p, & \text{if } e_i \in T^0(p), \\ p, & \text{if } e_i \in \Pi^* \cap T^0, \\ c_i + u_i, & \text{if } e_i \in \Pi^* \cap (E \setminus T^0), \\ c_i, & \text{otherwise,} \end{cases} \tag{8}$$

and the associate objective value is $w^s(T^0(p)) + \sum_{e_i \in \Pi^*} w_i$.

It is clear that Step 1 to check whether $G(p)$ is connected or not takes $O(n)$ time. Step 2 to find a minimum value cut Π^* of the graph $G(V, E(p), v)$ takes $O(n^3)$ time (see [10]). Hence, Algorithm 1 runs in $O(n + n^3) = O(n^3)$ time in the worst-case, and it is a strongly polynomial algorithm.

Now we consider the problem (1). In section 2, we range the possible value of $d^{*b}(T^0)$ as $P = \{p_1, p_1, \ldots, p_\eta\}$. And above we discussed for a given value $p \in P$, we can get an associated modification weight in $O(n^3)$ time. So we can give an algorithm to solve the problem (1) in strongly polynomial time as follows:

Algorithm 2

Step 0. Let $i = 1$ and $I = \emptyset$.

Step 1. For value p_i, run Algorithm 1 to solve the restricted version of the problem (1). If the restricted version problem is infeasible, then go to Step 2; otherwise, we denote the objective value get from Algorithm 1 as V_i and $I = I \cup \{i\}$, then go to Step 2.

Step 2. $i = i + 1$, if $i \leq \eta$, go back to Step 1; otherwise go to Step 3.

Step 3. Output an optimal objective value $\min_{i \in I} V_i$.

If we call solve the restricted version problem as an iteration, then Algorithm 2 needs to run η iteration, and from section 2, we know $\eta \leq (2n-1)$, combining with the analysis of Algorithm 1, Algorithm 2 runs in $O((2n-1) * n^3) = O(n^4)$ time in the worst-case, and it is a strongly polynomial algorithm.

4 Concluding Remarks

In this paper we studied the inverse min-max spanning tree problem under the weighted sum-type Hamming distance. For the model considered, we presented strongly polynomial algorithm to solve it.

As a future research topic, it will be meaningful to consider other inverse combinational optimization problems under Hamming distance. Studying computational complexity results and proposing optimal/approximation algorithms are promising.

References

1. Camerini, P.M.: The min-max spanning tree problem and some extansions. Information Processing Letters 7, 10–14 (1978)
2. Yang, C., Zhang, J., Ma, Z.: Some inverse min-max network problems under weighted l_1 and l_∞ norms with bound constrints on changes. Journal of Combinatorial Optimization 13, 123–135 (2007)
3. Heuberger, C.: Inverse Optimization: A survey on problems, methods, and results. Journal of Combinatorial Optimization 8, 329–361 (2004)
4. He, Y., Zhang, B.W., Yao, E.Y.: Wighted inverse minimum spanning tree problems under Hamming distance. Journal of Combinatorial Optimization 9, 91–100 (2005)
5. He, Y., Zhang, B.W., Zhang, J.Z.: Constrained inverse minimum spanning tree problems under the bottleneck-type Hamming distance. Journal of Global Optimization 34(3), 47–474 (2006)
6. Zhang, B.W., Zhang, J.Z., He, Y.: The center location improvement problem under the Hamming distance. Journal of Combinatorial Optimization 9, 187–198 (2005)
7. Yang, X.G., Zhang, J.Z.: Sone new results on inverse sorting problems. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 985–992. Springer, Heidelberg (2005)
8. Liu, L.C., Zhang, J.Z.: Inverse maximum flow problems under the weighted Hamming distance. Journal of Combinatorial Optimization 12, 395–408 (2006)
9. Liu, L.C., Yao, E.Y.: Weighted inverse minimum cut problem under the bottleneck type Hamming distance. Asia-Pacific Journal of Operational Research (to appear)
10. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice-Hall, Englewood Cliffs (1993)

Robust Optimization Model for a Class of Uncertain Linear Programs

Weimin Miao[1], Hongxia Yin[1,⋆], Donglei Du[2,⋆⋆], and Jiye Han[3,⋆⋆⋆]

[1] School of Mathematical Sciences, Graduate University of Chinese Academy of Sciences, P.O. Box 4588, Beijing 100049, China
wmmiao@hotmail.com, hxyin@gucas.ac.cn

[2] Faculty of Business Administration, University of New Brunswick, P.O.Box 4400, Fredericton, NB, E3B 5A3, Canada
ddu@unb.ca

[3] Institute of Applied Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100080, China
jiyehan@vip.sina.com

Abstract. In the paper, we propose a tractable robust counterpart for solving the uncertain linear optimization problem with correlated uncertainties related to a causal ARMA(p, q) process. This explicit treatment of correlated uncertainties under a time series setting in robust optimization is in contrast to the independent or simple correlated uncertainties assumption in existing literature. under some reasonable assumptions, we establish probabilistic guarantees for the feasibility of the robust solution. Finally, we provide a numerical method for the selection of the parameters which controls the tradeoff among the tractability, the robustness and the optimality of the robust model.

1 Introduction

Robust optimization is an important technique for investigating optimization problems with uncertainties. We consider the uncertain linear optimization problem

$$\max \left\{ \boldsymbol{c}'\boldsymbol{x} | \tilde{\boldsymbol{A}}\boldsymbol{x} \leq \tilde{\boldsymbol{b}} \right\}, \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{c}$ are real vectors in $\mathcal{R}^n$, $\tilde{\boldsymbol{A}} \in \mathcal{R}^{m\times n}$ and $\tilde{\boldsymbol{b}} \in \mathcal{R}^m$ are uncertain matrix and vector with entries belonging to a known set $\mathcal{U}$. The robust counterpart of problem (1) is given by

$$\max \left\{ \boldsymbol{c}'\boldsymbol{x} | \tilde{\boldsymbol{A}}\boldsymbol{x} \leq \tilde{\boldsymbol{b}}, \forall (\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}) \in \mathcal{U} \right\}. \tag{2}$$

⋆ The author's research is supported by the National Natural Science Foundation of China 10671203,70531040, and 70621001.

⋆⋆ The author's research is supported in part by NSERC grant 283103, and URF, FDF at UNB.

⋆⋆⋆ The author's research is supported by the National Natural Science Foundation of China NSF 10401038.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 384–395, 2007.

As a first attempt to attack the uncertain optimization problem with a sense of 'robustness', Soyster [15] proposed a worst-case model for (1) where each column of matrix $\tilde{\boldsymbol{A}}$ in the constraints $\tilde{\boldsymbol{A}}\boldsymbol{x} \leq \boldsymbol{b}, x \geq 0$ belongs to a convex set. However, the optimal solution of the robust counterpart is too conservative as it protects against the worst-case scenario. In order to overcome this over-conservatism, in the past decade, many different robust models have been proposed, and they differ mainly in the choice of the uncertainty sets and the tractability of the resultant robust counterparts [1,2,3,4,5,6,7,11,12,13]. The reader is referred to the aforementioned papers and references therein for more comprehensive information on the area of robust optimization, which has emerged as a competitive methodology for solving uncertain optimization problems.

For a uncertain linear constraint in problem (1), i.e.,

$$\tilde{\boldsymbol{a}}'\boldsymbol{x} \leq \tilde{b}, \tag{3}$$

where the input parameters $\tilde{\boldsymbol{a}} \in \mathcal{R}^n, \tilde{b} \in \mathcal{R}$ are uncertain data. Let $\mathcal{N} = \{1, 2, \cdots, N\}$ be an index set. Under the usual assumption that the uncertain data $(\tilde{\boldsymbol{a}}, \tilde{b})$ follows the affine data perturbation, we can represent uncertainties over a set of random variables $\{\tilde{z}_j, j \in \mathcal{N}\}$ as follows,

$$(\tilde{\boldsymbol{a}}, \tilde{b}) = (\boldsymbol{a}^0, b^0) + \sum_{j \in \mathcal{N}} (\Delta\boldsymbol{a}^j, \Delta b^j)\tilde{z}_j, \tag{4}$$

where $(\boldsymbol{a}^0, b^0)$ is the nominal value of the data, $(\Delta\boldsymbol{a}^j, \Delta b^j)$ is a direction of the data perturbation, and $\tilde{z}_j$ is a random variable, for each $j \in \mathcal{N}$. With a suitable selection of $(\Delta\boldsymbol{a}^j, \Delta b^j)$, the uncertainty model (4) will reduces to the simple case where all the uncertain coefficients in (3) are independent as assumed in [1,3,5,6].

In the existing literature [1,2,3,6,7,8,12,13], it is generally assumed that $\{\tilde{z}_j, j \in \mathcal{N}\}$ are independent and identically distributed (i.i.d.). Very recently, in order to capture distributional asymmetry of $\{\tilde{z}_j\}$, Chen et al. [11] propose the forward and backward deviation measures for bounded random variables to lead to better approximations of the stochastic constraints, and produce a scalable method for solving multistage stochastic programs. However, as limited to our knowledge, more complicated correlated uncertainties on the random variables $\tilde{z}_j$'s have not yet been considered up to now.

In this paper, we consider the case where the random variables $\{\tilde{z}_j, j \in \mathcal{N}\}$ are truncated from a causal ARMA process $\{\tilde{z}_j, j \in \mathcal{Z}(\text{an integer set })\}$, which is defined in terms of homogeneous linear difference equations with constant coefficients. A process $\{\tilde{z}_j, j \in \mathcal{Z}\}$ is said to be an *ARMA(p, q) process* if it is stationary and for each j,

$$\tilde{z}_j - \phi_1\tilde{z}_{j-1} - \cdots - \phi_p\tilde{z}_{j-p} = \tilde{\omega}_j + \theta_1\tilde{\omega}_{j-1} + \cdots + \theta_q\tilde{\omega}_{j-q}, \tag{5}$$

where ϕ_p and θ_q are nonzero constants, and each $\tilde{\omega}_k (k \in \mathcal{Z})$ is a white noise with mean zero. We call $\{\tilde{\omega}_k, k \in \mathcal{Z}\}$ the *primitive uncertainties* and $\{\tilde{z}_j, j \in \mathcal{N}\}$ the *sub-primitive uncertainties* in our research. Moreover, with a suitable selection

of $(\Delta \boldsymbol{a}^j, \Delta b^j)$, the uncertainty model (4) will reduce to the special case that the uncertain coefficients $\{\tilde{a}_1, \cdots, \tilde{a}_n, \tilde{b}\}$ in (3) follow an ARMA process, which is often encountered in practice.

ARMA processes are the most important stochastic processes in time series analysis. They have wide applications in economic forecasting, signal processing and communications. For example, it is used in modeling the S&P 500 stock index, the UK interest rate spread, and the returns on the FTA All Share index (see [14], 30-37). In particular, in many finance applications, it is reasonable to assume that the input random variables $\tilde{\boldsymbol{a}}$ and/or $\tilde{b}$ in the constraints (3) are uncertainties related to (or truncated from) the causal ARMA process. For example, the stochastic interest rate has been introduced in many financial models recently, and the interest rate series $\{i_t, t \in \mathcal{Z}\}$ is often modeled to belong to the class of ARIMA(p, d, q) process, mostly ARIMA$(p, 1, q)$ process, which means that the interest spread series $\{\tilde{z}_{t+1} \mid \tilde{z}_{t+1} := i_{t+1} - i_t\}$ is an ARMA(p, q) process. Thus, we can rewrite the future interest rate series with respect to interest spread $\tilde{z}_j$ as $\{i_t \mid i_t = i_0 + \sum_{j=1}^{t} \tilde{z}_j\}$ and find that $i_1, \cdots, i_n$ satisfy our uncertainty model (4).

The main contribution of this work is to introduce a new robust formulation for the uncertain linear constraint (3) with affine data perturbation related to a causal ARMA process. We believe that the embedding of the natural correlated uncertainties related to time series into existing models is an integrated part of almost all practical applications of robust optimization. This explicit treatment of correlated uncertainties under time series setting in robust optimization is a major difference from existing literature, such as [3,7], where the i.i.d. of the sub-primitive uncertainties $\tilde{z}_j$'s are assumed. Such consideration leads to much more involved uncertain constraints. As a consequence, the robust formulation obtained from the existing robust technology becomes intractable because of the infinite items of primitive uncertainties. In our paper, we give a numerical robust method to address such problems.

The rest of the paper is organized as follows. In Section 2 we recall some preliminary results on ARMA process which are necessary in the sequel. In Section 3, we consider a special ARMA process, where $\tilde{\omega}_k$'s, $k \in \mathcal{Z}$, are white noises with means zero, and $\tilde{\omega}_k$'s themselves are i.i.d. for all k. In Section 4, we derive level of conservatism of the robust solutions in terms of probabilistic bounds of constraint violations and propose a numerical method to select the parameters to control the tractability, the robustness, and the optimality of the robust model. In Section 5, we propose an approximate numerical method for calculating the error bounds. In Section 6, we conclude the paper.

2 The Causal ARMA Process

In this section, we recall the definition and some crucial properties of ARMA process [10], which is one important family of process in time series analysis.

Definition 1. *The process* $\{\tilde{z}_j, j \in \mathcal{Z}\}$ *is said to be an ARMA*(p,q) *process if the process* $\{\tilde{z}_j\}$ *is stationary and for each* j,

$$\tilde{z}_j - \phi_1 \tilde{z}_{j-1} - \cdots - \phi_p \tilde{z}_{j-p} = \tilde{\omega}_j + \theta_1 \tilde{\omega}_{j-1} + \cdots + \theta_q \tilde{\omega}_{j-q}, \tag{6}$$

with $\phi_p \neq 0$ *and* $\theta_q \neq 0$, *where* $\{\tilde{\omega}_j\}$ *is a white noise with mean zero.*

We call (6) the ARMA(p,q) model, and call $\phi(\cdot)$ and $\theta(\cdot)$ the autoregressive and moving average polynomials respectively with the following representations:

$$\phi(t) = 1 - \phi_1 t - \cdots - \phi_p t^p, \quad \theta(t) = 1 + \theta_1 t + \cdots + \theta_q t^q, \quad \phi_p, \theta_q \neq 0. \tag{7}$$

Definition 2. *An ARMA*(p,q) *process defined by (6) is said to be causal if there exists a sequence of constants* $\{\psi_k\}$ *such that* $\sum_{k=0}^{\infty} |\psi_k| < \infty$ *and*

$$\tilde{z}_j = \sum_{k=0}^{\infty} \psi_k \tilde{\omega}_{j-k}, \quad j \in \mathcal{Z}. \tag{8}$$

The following theorem gives a necessary and sufficient condition for an ARMA model to be causal, and provides an explicit representation of $\tilde{z}_j$ in terms of $\{\tilde{\omega}_k, k \leq j\}$.

Theorem 1. (Theorem 3.1.1 in [10]) *Let* $\{\tilde{z}_j\}$ *be an ARMA*(p,q) *process where* $\phi(\cdot)$ *and* $\theta(\cdot)$ *have no common zeros. Then* $\{\tilde{z}_j\}$ *is causal if and only if* $\phi(t) \neq 0, \forall |t| \leq 1$. *The coefficients* ψ_k *in (8) are determined by*

$$\psi(t) = \sum_{k=0}^{\infty} \psi_k t^k = \theta(t)/\phi(t), \qquad |t| \leq 1. \tag{9}$$

For a linear process (6), the condition $\phi(t) \neq 0, \forall |t| \leq 1$ guarantees the process $\{\tilde{z}_j\}$ to be stationary. So in some literatures, such condition is also called the stationary condition of ARMA process ([9], 77-78).

Without loss of generality, we assume that the polynomials $\phi(\cdot)$ and $\theta(\cdot)$ have no common zeros—otherwise, we can delete the common factors of $\phi(\cdot)$ and $\theta(\cdot)$ in (6). Then we can determine the coefficients ψ_k by

$$\psi_k - \sum_{0<l\leq k} \phi_l \psi_{k-l} = \theta_k, \quad 0 \leq k < \max(p, q+1), \tag{10}$$

$$\psi_k - \sum_{0<l\leq p} \phi_l \psi_{k-l} = 0, \quad k \geq \max(p, q+1). \tag{11}$$

With the aid of the analysis for homogeneous linear difference equations with constant coefficients [9,10], (11) can be written as

$$\psi_k = \sum_{i=1}^{l} \sum_{j=0}^{r_i - 1} \alpha_{ij} k^j \xi_i^{-k}, \quad k \geq \max(p, q+1) - p, \tag{12}$$

where ξ_i, $i = 1, \cdots, l$, are the distinct zeros of $\phi(t)$, and r_i is the multiplicity of ξ_i. The p constants α_{ij} and the coefficients $\psi_k (0 \leq k < \max(p, q+1) - p)$, are then determined uniquely by the $\max(p, q+1)$ boundary conditions in (10).

Note that $\psi(t) = \frac{\theta(t)}{\phi(t)}$ is an analytic function in $\{t : |t| < \min_{i=1:l} |\xi_i|\}$. Let $\rho = \min_{i=1:l} |\xi_i| > 1$. Subsequently, we can generalize (9) as $\psi(t) = \sum_{k=0}^{\infty} \psi_k t^k = \theta(t)/\phi(t), |t| < \rho$. Using Abel's theorem on power series, we obtain that $\sum_{k=1}^{\infty} \psi_k \rho_0^k$, $1 < \rho_0 < \rho$, is absolutely convergent, which means that $\{\psi_k \rho_0^k\}$ is absolutely convergent to zero. Therefore, the coefficients $\{\psi_k\}$ is negative exponentially convergent to zero and can be calculated numerically. Moreover, $\{\psi_k\}$ converges faster when ρ becomes larger.

3 The Robust Model

Now we focus on the uncertain linear constraint (3) under affine data perturbation, where the uncertainties can be expressed as (4). Thus, we can rewrite the constraint in terms of sub-primitive uncertainties $\{\tilde{z}_j, j \in \mathcal{N}\}$ as follows:

$$\boldsymbol{a}^{0\prime}\boldsymbol{x} + \sum_{j \in \mathcal{N}} (\Delta \boldsymbol{a}^{j\prime}\boldsymbol{x} - \Delta b^j)\tilde{z}_j \leq b^0. \tag{13}$$

In this paper, we consider the case that the sub-primitive uncertainties $\tilde{z}_j$'s, $j \in \mathcal{N}$, are correlated uncertainties truncated from a causal ARMA(p, q) process defined in Definition 1. We also need the following assumptions.

Assumption 1. In ARMA model (6), the forcing terms $\tilde{\omega}_k, k \in \mathcal{Z}$ are i.i.d..

Assumption 2. The autoregressive and moving average polynomials $\phi(\cdot)$ and $\theta(\cdot)$ have no common zeros.

For simplicity, we denote $y_j(\boldsymbol{x}) =: \Delta \boldsymbol{a}^{j\prime}\boldsymbol{x} - \Delta b^j$ for each $j \in \mathcal{N}$. Now we have the following result.

Theorem 2. *If the uncertain data $(\tilde{\boldsymbol{a}}, \tilde{b})$ follows the affine data perturbation defined by (4) with the sub-primitive uncertainties $\{\tilde{z}_j, j \in \mathcal{N}\}$ truncated from a casual ARMA process, then, under Assumptions 1 and 2, the uncertain linear constraint (3) is equivalent to*

$$\boldsymbol{a}^{0\prime}\boldsymbol{x} + \sum_{k=1}^{\infty} \zeta_k(\boldsymbol{x})\tilde{\eta}_k \leq b^0, \tag{14}$$

where

$$\zeta_k(\boldsymbol{x}) = \sum_{j \in \mathcal{N}} y_j(\boldsymbol{x})\psi_{j+k-N-1}, \quad \tilde{\eta}_k = \tilde{\omega}_{N+1-k}. \tag{15}$$

Proof. We rewrite the left-hand side of (13) in terms of the primitive uncertainties $\{\tilde{\omega}_k\}$:

$$\begin{aligned}\boldsymbol{a}^{0\prime}\boldsymbol{x}+\sum_{j\in\mathcal{N}}\left(\Delta\boldsymbol{a}^{j\prime}\boldsymbol{x}-\Delta b^j\right)\tilde{z}_j &= \boldsymbol{a}^{0\prime}\boldsymbol{x}+\sum_{k=N}^{-\infty}\left(\sum_{j\in\mathcal{N}}y_j(\boldsymbol{x})\psi_{j-k}\right)\tilde{\omega}_k\\ &= \boldsymbol{a}^{0\prime}\boldsymbol{x}+\sum_{k=1}^{\infty}\left(\sum_{j\in\mathcal{N}}y_j(\boldsymbol{x})\psi_{j+k-N-1}\right)\tilde{\omega}_{N+1-k}\\ &= \boldsymbol{a}^{0\prime}\boldsymbol{x}+\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x})\tilde{\eta}_k,\end{aligned} \tag{16}$$

where $\psi_k = 0$ for $-(N-1) \le k \le -1$. From the variable translations (15), it is easy to see that $\tilde{\eta}_k$'s are also i.i.d. as same as $\tilde{\omega}_k$'s. Moreover, for each $\boldsymbol{x} \in \mathcal{R}$, we have $\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x})\tilde{\eta}_k < \infty$.

From Theorem 2, the constraint (3) is dependent on infinite uncertainties $\zeta_k(\boldsymbol{x})$ and $\tilde{\eta}_k$, $1 \le k < \infty$. Thus we cannot obtain a robust formulation of constraint (3) by existing robust techniques directly, because an optimization problem with a constraint as (14) is intractable in general.

However, we know that $\{\zeta_k(\boldsymbol{x})\}$ converges to zero negative exponentially for each feasible solution $\boldsymbol{x}$, followed by Assumptions 1, 2, equality (15), and the same property of $\{\psi_k\}$. Therefore the first M random variables $\tilde{\eta}_k$, $1 \le k \le M$ dominate in (14). Let $\mathcal{M} = \{1, 2, \cdots, M\}$ be an index set. So it is natural to use the following constraint with finite primitive uncertainties to approximate the constraint (14):

$$\boldsymbol{a}^{0\prime}\boldsymbol{x}+\sum_{k\in\mathcal{M}}\zeta_k(\boldsymbol{x})\tilde{\eta}_k \le b^0. \tag{17}$$

We define the uncertainty set of (17), named $\mathcal{U}$, as follows:

$$\mathcal{U}=\left\{(\boldsymbol{a},b)\;\middle|\;\exists\,\tilde{\eta}\in\mathcal{R}^M,(\boldsymbol{a},b)=(\boldsymbol{a}^0,b^0)+\sum_{k\in\mathcal{M}}\left(\sum_{j\in\mathcal{N}}\Delta\boldsymbol{a}^j,\sum_{j\in\mathcal{N}}\Delta b^j\right)\tilde{\eta}_k,\;\|\tilde{\eta}\|\le\Omega\right\}, \tag{18}$$

where $\tilde{\eta} = (\tilde{\eta}_1, \tilde{\eta}_2, \cdots, \tilde{\eta}_M)'$ and Ω is the parameter controlling the tradeoff between the robustness and the optimality. Subsequently, we obtain a robust formulation to the uncertain linear constraint (3) as follows:

$$\boldsymbol{a}^{0\prime}\boldsymbol{x}+\sum_{k\in\mathcal{M}}\zeta_k(\boldsymbol{x})\tilde{\eta}_k \le b^0, \quad \forall\ \|\tilde{\eta}\| \le \Omega\ (\text{or } \forall\ (\tilde{\boldsymbol{a}},\tilde{b}) \in \mathcal{U}). \tag{19}$$

Theorem 3. *The robust formulation (19) is equivalent to*

$$\boldsymbol{a}^{0\prime}\boldsymbol{x}+\Omega\|\boldsymbol{t}\|^* \le b^0, \tag{20}$$

where

$$t_k = \zeta_k(\boldsymbol{x}) = \sum_{j\in\mathcal{N}}(\Delta\boldsymbol{a}^{j\prime}\boldsymbol{x}-\Delta b^j)\psi_{j+k-N-1},\ \forall k\in\mathcal{M}, \tag{21}$$

and the dual norm $\|\cdot\|^*$ is defined by $\|\boldsymbol{t}\|^* = \max_{\|\boldsymbol{s}\|\le 1}\boldsymbol{t}'\boldsymbol{s}$.

Proof. It is easy to see that (19) is equivalent to

$$\boldsymbol{a}^{0\prime}\boldsymbol{x}+\max_{\|\tilde{\eta}\|\le\Omega}\sum_{k\in\mathcal{M}}\zeta_k(\boldsymbol{x})\tilde{\eta}_k \le b^0. \tag{22}$$

From the definition of the dual norm, we know that

$$\begin{aligned}\max_{\|\tilde{\eta}\|\leq\Omega}\sum_{k\in\mathcal{M}}\zeta_k(\boldsymbol{x})\tilde{\eta}_k &= \Omega\max_{\|\tilde{\eta}\|\leq 1}\sum_{k\in\mathcal{M}}\zeta_k(\boldsymbol{x})\tilde{\eta}_k \\ &= \Omega\|(\zeta_1(\boldsymbol{x}),\cdots,\zeta_M(\boldsymbol{x}))'\|^*.\end{aligned} \tag{23}$$

By letting $\boldsymbol{t}=(\zeta_1(\boldsymbol{x}),\cdots,\zeta_M(\boldsymbol{x}))'$, we obtain what we desired.

4 Probabilistic Guarantees and Parameters Selection

Let $\boldsymbol{y}(\boldsymbol{x})=(y_1(\boldsymbol{x}),y_2(\boldsymbol{x}),\cdots,y_N(\boldsymbol{x}))'$, $\Psi_{k-N}=(\psi_{k-N},\psi_{k-N+1},\cdots,\psi_{k-1})'$ and $\Lambda_{k-N}=\Psi_{k-N}\Psi'_{k-N}$. Then we have $\zeta_k(\boldsymbol{x})=\Psi'_{k-N}\boldsymbol{y}(\boldsymbol{x})$ and $\zeta_k^2(x)=\boldsymbol{y}(\boldsymbol{x})'\Lambda_{k-N}\boldsymbol{y}(\boldsymbol{x})$. We denote $\lambda_{\min}(\Lambda)$ and $\lambda_{\max}(\Lambda)$ as the minimum and the maximum eigenvalues of matrix Λ, respectively.

Proposition 1

(i) Every element in matrix Λ_{k-N} converges to zero negative exponentially as $k\to\infty$. Thus $\sum_{k=1}^{\infty}\Lambda_{k-N}$ is meaningful, and $\sum_{k=1}^{\infty}\zeta_k^2(\boldsymbol{x})<\infty$, for each $\boldsymbol{x}\in\mathcal{R}^n$.

(ii) For any integers L and K satisfying $1\leq L\leq K$ (including $K=\infty$), $\sum_{k=L}^{K}\Lambda_{k-N}$ is positive semidefinite, and

$$\lambda_{\min}\left(\sum_{k=L}^{K}\Lambda_{k-N}\right)\|\boldsymbol{y}(\boldsymbol{x})\|_2^2\leq\sum_{k=L}^{K}\zeta_k^2(\boldsymbol{x})\leq\lambda_{\max}\left(\sum_{k=L}^{K}\Lambda_{k-N}\right)\|\boldsymbol{y}(\boldsymbol{x})\|_2^2. \tag{24}$$

(iii) On one hand, $\lambda_{\max}\left(\sum_{k=K}^{\infty}\Lambda_{k-N}\right)$ and $\lambda_{\min}\left(\sum_{k=K}^{\infty}\Lambda_{k-N}\right)$ are monotonically decreasing and converge to 0 respectively as $K\to\infty$. On the other hand, $\lambda_{\max}\left(\sum_{k=1}^{K}\Lambda_{k-N}\right)$ and $\lambda_{\min}\left(\sum_{k=1}^{K}\Lambda_{k-N}\right)$ are monotonically increasing and converge to $\lambda_{\max}\left(\sum_{k=1}^{\infty}\Lambda_{k-N}\right)$ and $\lambda_{\min}\left(\sum_{k=1}^{\infty}\Lambda_{k-N}\right)$ respectively as $K\to\infty$.

(iv) Moreover

$$\lambda_{\min}\left(\sum_{k=1}^{K}\Lambda_{k-N}\right)>0,\ \forall K\geq N\quad(\textit{including } K=\infty). \tag{25}$$

Proof. (i) For each $\boldsymbol{x}$, $\zeta_k(\boldsymbol{x})$ is a linear combination of $\psi_{k-N},\cdots,\psi_{k-1}$. The element of Λ_{k-N} in m-th row and n-th column is $\lambda_{m,n}^{k-N}=:\psi_{k-N+m-1}\psi_{k-N+n-1}$, $1\leq m,n\leq N$. Note that $\lim_{k\to\infty}\lambda_{m,n}^{k-N}\to 0$ and $\sum_{k=1}^{\infty}\lambda_{m,n}^{k-N}$ is convergent because $\{\psi_k\}$ converges to zero negative exponentially as $k\to\infty$. Thus, $\sum_{k=1}^{\infty}\zeta_k^2(\boldsymbol{x})<\infty$, and hence $\sum_{k=1}^{\infty}\Lambda_{k-N}$ is meaningful.

(ii) Because of $\sum_{k=L}^{K}\zeta_k^2(\boldsymbol{x})=\sum_{k=L}^{K}\boldsymbol{y}(\boldsymbol{x})'\Lambda_{k-N}\boldsymbol{y}(\boldsymbol{x})=\boldsymbol{y}(\boldsymbol{x})'\left(\sum_{k=L}^{K}\Lambda_{k-N}\right)\boldsymbol{y}(\boldsymbol{x})$, the inequalities (24) can be obtained from Courant-Fischer Theorem ([16],107).

(iii) Let K be any given positive integer. Because all the matrices in consideration are symmetric, so their eigenvalues must satisfy

$$\begin{aligned}\lambda_{\max}\left(\sum_{j=1}^{K+1}\Lambda_{j-N}\right) &\geq \lambda_{\max}(\sum_{k=1}^{K}\Lambda_{k-N}) + \lambda_{\min}(\Lambda_{K+1-N}),\\ \lambda_{\min}\left(\sum_{k=1}^{K+1}\Lambda_{k-N}\right) &\geq \lambda_{\min}(\sum_{k=1}^{K}\Lambda_{k-N}) + \lambda_{\min}(\Lambda_{K+1-N}).\end{aligned} \tag{26}$$

From (i), Λ_{K+1-N} is positive semidefinite and $\{\Lambda_{k-N}\}$ converges to the zero matrix as $k \to \infty$. The monotonicity and the convergence of the maximum and minimum eigenvalues follows easily.

(iv) Assume that there exists a $K \geq N$ (including $K = \infty$) such that $\lambda_{\min}(\sum_{k=1}^{K}\Lambda_{k-N}) = 0$. Consequently, there exists a vector $\boldsymbol{y}^* \neq 0$ such that $\boldsymbol{y}^*\left(\sum_{k=1}^{K}\Lambda_{k-N}\right)\boldsymbol{y}^* = \sum_{k=1}^{K}(\Psi'_{k-N}\boldsymbol{y}^*)^2 = 0$. Thus,

$$\Psi'_{k-N}\boldsymbol{y}^* = 0, \forall k \in \mathcal{N}, \tag{27}$$

As discussed in Section 2, we have $\psi_0 = 1$. Therefore, Equations (27) have the unique solution $\boldsymbol{y} = 0$, in contradiction with our assumption. Subsequently, we conclude that $\lambda_{\min}\left(\sum_{k=1}^{K}\Lambda_{k-N}\right) > 0, \forall K \geq N$ (including $K = \infty$).

Now we are ready to give the probabilistic guarantees on the feasibility.

Theorem 4

(i) For any given feasible solution $\boldsymbol{x}^*$ in the robust formulation (20) with respect to the uncertainty model (3), we have

$$P\left(\tilde{\boldsymbol{a}}\boldsymbol{x}^* > \tilde{b}\right) \leq P\left(\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k > \Omega\|\boldsymbol{t}\|^*\right), \tag{28}$$

where $\boldsymbol{t} = (\zeta_1(\boldsymbol{x}^*), \cdots, \zeta_M(\boldsymbol{x}^*))'$.

(ii) Assume that, in the ARMA(p, q) process, the primitive uncertainties $\tilde{\omega}_k$'s are independent and normally distributed with mean 0 and variance δ^2. That is, $\tilde{\omega}_k \sim N(0, \delta^2)$, and i.i.d. Then, for any $M \geq N$, assuming the l_2-norm (and hence $\|\cdot\|^* = \|\cdot\|_2$), we have

$$P\left(\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k > \Omega\|\boldsymbol{t}\|^*\right) \leq \Phi\left(-\frac{\Omega}{\delta\sqrt{1+\Gamma(M)}}\right), \tag{29}$$

where

$$\Gamma(M) = \frac{\lambda_{\max}\left(\sum_{k=M+1}^{\infty}\Lambda_{k-N}\right)}{\lambda_{\min}\left(\sum_{k\in\mathcal{M}}\Lambda_{k-N}\right)}, \tag{30}$$

and

$$\Phi(\theta) = \frac{1}{\sqrt{2\pi}}\int_{\theta}^{-\infty}\exp\left(-\frac{t^2}{2}\right)\mathrm{d}t \tag{31}$$

is the cumulative density function (cdf) of a standard normal distribution.

(iii) Assume that, in the ARMA(p,q) process, the primitive uncertainties $\tilde{\omega}_k$'s are independent, symmetrical and identically distributed in $[-\omega,\omega]$ with mean 0. Then, for any $M \geq N$, assuming the l_2-norm, we have

$$P\left(\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k > \Omega\|\boldsymbol{t}\|^*\right) \leq \exp\left(\frac{\Omega^2}{2\omega^2}(\Gamma(M)-1)\right), \tag{32}$$

where $\Gamma(M)$ is defined as (30).

Proof. (i) From the robust counterpart (20), we have

$$\begin{aligned} P\left(\tilde{\boldsymbol{a}}'\boldsymbol{x}^* > \tilde{b}\right) &= P\left(\boldsymbol{a}^{0\prime}\boldsymbol{x}^* + \textstyle\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k > b^0\right) \\ &\leq P\left(\boldsymbol{a}^{0\prime}\boldsymbol{x}^* + \textstyle\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k > \boldsymbol{a}^{0\prime}\boldsymbol{x}^* + \Omega\|\boldsymbol{t}\|^*\right) \\ &= P\left(\textstyle\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k > \Omega\|\boldsymbol{t}\|^*\right), \end{aligned}$$

where $\boldsymbol{t} = (\zeta_1(\boldsymbol{x}^*),\cdots,\zeta_M(\boldsymbol{x}^*))'$.

(ii) For the l_2-norm, we have $\|\boldsymbol{t}\|^* = \|\boldsymbol{t}\|_2 = \sqrt{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}$. Note that $\tilde{\omega}_k \sim N(0,\delta^2)$, and i.i.d. Therefore, $\tilde{\eta}_k \sim N(0,\delta^2)$, and i.i.d. Consequently, $\frac{\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k}{\delta\sqrt{\sum_{k=1}^{\infty}\zeta_k^2(\boldsymbol{x}^*)}} \sim N(0,1)$. Now, we have

$$\begin{aligned} P\left(\textstyle\sum_{k=1}^{\infty}\zeta_j(\boldsymbol{x}^*)\tilde{\eta}_k > \Omega\|\boldsymbol{t}\|^*\right) &= P\left(\frac{\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k}{\delta\sqrt{\sum_{k=1}^{\infty}\zeta_k^2(\boldsymbol{x}^*)}} > \frac{\Omega\sqrt{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}}{\delta\sqrt{\sum_{k=1}^{\infty}\zeta_k^2(\boldsymbol{x}^*)}}\right) \\ &= \Phi\left(-\frac{\Omega\sqrt{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}}{\delta\sqrt{\sum_{k=1}^{\infty}\zeta_k^2(\boldsymbol{x}^*)}}\right). \end{aligned}$$

From (24) in Proposition 1, we obtain that

$$\begin{aligned} P\left(\textstyle\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k > \Omega\|\boldsymbol{t}\|^*\right) &= \Phi\left(-\frac{\Omega}{\delta}\left(1+\frac{\sum_{k=M+1}^{\infty}\zeta_k^2(\boldsymbol{x}^*)}{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}\right)^{-1/2}\right) \\ &\leq \Phi\left(-\frac{\Omega}{\delta}\left(1+\frac{\lambda_{\max}\left(\sum_{k=M+1}^{\infty}\Lambda_{k-N}\right)\|\boldsymbol{y}(\boldsymbol{x}^*)\|_2^2}{\lambda_{\min}\left(\sum_{k\in\mathcal{M}}\Lambda_{k-N}\right)\|\boldsymbol{y}(\boldsymbol{x}^*)\|_2^2}\right)^{-1/2}\right) \\ &= \Phi\left(-\frac{\Omega}{\delta\sqrt{1+\Gamma(M)}}\right), \end{aligned}$$

where $\Gamma(M)$ is defined as (30).

(iii) From the Markov Inequality, we have

$$\begin{aligned} P\left(\textstyle\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k > \Omega\|\boldsymbol{t}\|^*\right) &= P\left(\exp\left(\frac{\Omega\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k}{\omega^2\sqrt{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}}\right) > \exp\left(\frac{\Omega^2}{\omega^2}\right)\right) \\ &\leq \exp\left(-\frac{\Omega^2}{\omega^2}\right)E\left(\exp\left(\frac{\Omega\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k}{\omega^2\sqrt{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}}\right)\right). \end{aligned}$$

Because $\tilde{\eta}_k$'s have the same distribution as $\tilde{\omega}_k$'s, which are independent, symmetric and identical distribution in $[-\omega,\omega]$ with mean 0, thus we have

$$
\begin{aligned}
P\left(\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k > \Omega\|\boldsymbol{t}\|^*\right) &\le \exp\left(-\frac{\Omega^2}{\omega^2}\right)\Pi_{k=1}^{\infty}E\left(\exp\left(\frac{\Omega\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k}{\omega^2\sqrt{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}}\right)\right)\\
&= \exp\left(-\frac{\Omega^2}{\omega^2}\right)\Pi_{k=1}^{\infty}\sum_{l=0}^{\infty}\left(\frac{1}{(2l)!}E\left(\frac{\Omega\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k}{\omega^2\sqrt{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}}\right)^{2l}\right)\\
&\le \exp\left(-\frac{\Omega^2}{\omega^2}\right)\Pi_{k=1}^{\infty}\exp\left(\frac{\Omega^2}{2\omega^2}\frac{\zeta_k^2(\boldsymbol{x}^*)}{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}\right)\\
&= \exp\left(-\frac{\Omega^2}{\omega^2}\right)\exp\left(\frac{\Omega^2}{2\omega^2}\frac{\sum_{k=1}^{\infty}\zeta_k^2(\boldsymbol{x}^*)}{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}\right)\\
&= \exp\left(-\frac{\Omega^2}{2\omega^2}\left(1-\frac{\sum_{k=M+1}^{\infty}\zeta_k^2(\boldsymbol{x}^*)}{\sum_{k\in\mathcal{M}}\zeta_k^2(\boldsymbol{x}^*)}\right)\right).
\end{aligned}
$$

From (24) in Proposition 1, we obtain that

$$
\begin{aligned}
P\left(\sum_{k=1}^{\infty}\zeta_k(\boldsymbol{x}^*)\tilde{\eta}_k > \Omega\|\boldsymbol{t}\|^*\right) &\le \exp\left(-\frac{\Omega^2}{2\omega^2}\left(1-\frac{\lambda_{\max}\left(\sum_{k=M+1}^{\infty}\Lambda_{k-N}\right)\|\boldsymbol{y}(\boldsymbol{x}^*)\|_2^2}{\lambda_{\min}\left(\sum_{k\in\mathcal{M}}\Lambda_{k-N}\right)\|\boldsymbol{y}(\boldsymbol{x}^*)\|_2^2}\right)\right)\\
&= \exp\left(-\frac{\Omega^2}{2\omega^2}(1-\Gamma(M))\right),
\end{aligned}
$$

where $\Gamma(M)$ is defined as (30).

5 An Approximate Numerical Method for Calculating $\Gamma(M)$

For several simple cases such as where the sub-primitive uncertainties $\{\tilde{z}_j, j \in \mathcal{N}\}$ are truncated from an ARMA$(1,0)$ process, we are able to calculate $\Gamma(M)$ analytically. But for most practical applications, $\Gamma(M)$ is usually difficult to be calculated directly since we can not derive an explicit expression of matrix $\sum_{k=M+1}^{\infty}\Lambda_{k-N}$. Fortunately, in many cases, the autoregressive and moving average orders p and q are not very large or even small. In such cases we are able to obtain the explicit expression of each ψ_k $(k \ge \max(p, q+1)-p)$ in (12). Based on these, we propose a numerical method to approximate $\Gamma(M)$, which provides a controllable error bound to the exact value of $\Gamma(M)$.

Algorithm 1. (Method to calculate $\Gamma(M)$ approximately)

Step 0. Let $M \ge N$ be an integer. Choose an $s \in (0,2)$ and a tolerance parameter $\tau > 0$. Let $\alpha = \sum_{i=1}^{l}\sum_{j}^{r_i-1}|\alpha_{ij}|$, $r = \max_{i=1:\ l}\{r_i\} - 1$, and $\rho = \min_{i=1:\ l}|\xi_i| > 1$, where the notations are the same as those in (12).

Step 1. Calculate $\lambda_1 = \lambda_{\min}\left(\sum_{k\in\mathcal{M}}\Lambda_{k-N}\right)$.

Step 2. Choose K_1 to be the smallest nonnegative integer k satisfying $k^{2r} \le \rho^{sk}$. Choose K_2 to be the smallest nonnegative integer k satisfying $\frac{\alpha^2\rho^{-(2-s)k}}{1-\rho^{-(2-s)}} \le \frac{\tau\lambda_1}{N}$. Let $K = \max\{K_1, K_2, \max(p, q+1)-p\}$.

Step 3. If $K + N - 1 \le M$, then let $\Gamma^*(M) = 0$ and stop. Otherwise, calculate $\lambda_2 = \lambda_{\max}\left(\sum_{k=M+1}^{K+N-1}\Lambda_{k-N}\right)$. Let $\Gamma^*(M) = \frac{\lambda_2}{\lambda_1}$ and stop.

Theorem 5. *Assume that ψ_k has the explicit expression as (12). For any given integer M satisfying $M \ge N$, let $\Gamma^*(M)$ be generated from the Algorithm 1. Then the exact value $\Gamma(M) \in [\max\{0, \Gamma^*(M)-\tau\}, \Gamma^*(M)+\tau]$.*

Proof. Note that $K = \max\{K_1, K_2, \max(p, q+1) - p\}$ in Step 2 of Algorithm 1.

First, $K > \max(p, q+1) - p$ implies that $|\psi_k| = \left|\sum_{i=1}^{l} \sum_{j=0}^{r_i - 1} \alpha_{ij} k^j \xi_i^{-k}\right| \leq \alpha k^r \rho^{-k}$, $\forall k \geq K$. Consequently, the (m, n)-th element in $\sum_{k=K+N}^{\infty} \Lambda_{k-N}$ is

$$\begin{aligned}\left(\sum_{k=K+N}^{\infty} \Lambda_{k-N}\right)_{m,n} &= \sum_{k=K}^{\infty} |\psi_{k+m-1}\psi_{k+n-1}| \\ &\leq \sum_{k=K}^{\infty} \alpha^2 (k+m-1)^r \rho^{-(k+m-1)} (k+n-1)^r \rho^{-(k+n-1)} \\ &\leq \sum_{k=K}^{\infty} \alpha^2 k^{2r} \rho^{-2k}.\end{aligned} \tag{33}$$

Second, $K \geq K_1$ implies that $k^{2r} \leq \rho^{sk}, \forall k \geq K$. Thus, $\sum_{k=K}^{\infty} \alpha^2 k^{2r} \rho^{-2k} \leq \sum_{k=K}^{\infty} \alpha^2 \rho^{-(2-s)k}$.

Third, $K \geq K_2$ implies that $\frac{\alpha^2 \rho^{-(2-s)K}}{1-\rho^{-(2-s)}} \leq \frac{\tau\lambda_1}{N}$. Combining the fact that $s \in (0, 2)$, we obtain that $\sum_{k=K}^{\infty} \alpha^2 \rho^{-(2-s)k} = \frac{\alpha^2 \rho^{-(2-s)K}}{1-\rho^{-(2-s)}} \leq \frac{\tau\lambda_1}{N}$.

Thus, each element of $\sum_{k=K+N}^{\infty} \Lambda_{k-N}$ satisfies $\left(\sum_{k=K+N}^{\infty} \Lambda_{k-N}\right)_{m,n} \leq \frac{\tau\lambda_1}{N}$.

1. $K + N - 1 \geq M + 1$. The Wielandt-Hoffman Theorem (see [16], pp.104) implies that

$$\begin{aligned}&\left(\lambda_{\max}\left(\sum_{k=M+1}^{\infty} \Lambda_{k-N}\right) - \lambda_{\max}\left(\sum_{k=M+1}^{K+N-1} \Lambda_{k-N}\right)\right)^2 \\ &\leq \sum_{i=1}^{N} \left(\lambda_i\left(\sum_{k=M+1}^{\infty} \Lambda_{k-N}\right) - \lambda_i\left(\sum_{k=M+1}^{K+N-1} \Lambda_{k-N}\right)\right)^2 \\ &\leq \left\|\sum_{k=K+N}^{\infty} \Lambda_{k-N}\right\|_F^2 \\ &\leq \tau^2 \lambda_1^2,\end{aligned} \tag{34}$$

   Then, we have $\lambda_{\max}\left(\sum_{k=M+1}^{\infty} \Lambda_{k-N}\right) \in [\lambda_2 - \tau\lambda_1, \lambda_2 + \tau\lambda_1]$. Let $\Gamma^*(M) = \frac{\lambda_2}{\lambda_1}$. Comining the fact that $\lambda_{\max}\left(\sum_{k=M+1}^{\infty} \Lambda_{k-N}\right) \geq 0$, we have $\Gamma(M) \in [\max\{0, \Gamma^*(M) - \tau\}, \Gamma^*(M) + \tau]$.
2. $K + N - 1 \leq M$. We have

$$\lambda_{\max}\left(\sum_{k=M+1}^{\infty} \Lambda_{k-N}\right) \leq \lambda_{\max}\left(\sum_{k=K+N}^{\infty} \Lambda_{k-N}\right) \leq \tau\lambda_1. \tag{35}$$

   Thus, $\lambda_{\max}\left(\sum_{k=M+1}^{\infty} \Lambda_{k-N}\right) \in [0, \tau\lambda_1]$. Let $\Gamma^*(M) = 0$. Then we obtain that $\Gamma(M) \in [0, \Gamma^*(M) + \tau] = [\max\{0, \Gamma^*(M) - \tau\}, \Gamma^*(M) + \tau]$.

From the discussion above, for any integer M satisfying $M \geq N$, we are able to calculate an approximate value $\Gamma^*(M)$ by Algorithm 1. Moreover, for any $\epsilon > 0$, if we choose Ω in case (ii) and case (iii) of Theorem 4 satisfying

$$\Omega \geq -\delta \Phi^{-1}(\varepsilon)\sqrt{1 + \Gamma^*(M) + \tau}, \tag{36}$$

and

$$\Omega \geq \omega\sqrt{\frac{-2\ln(\varepsilon)}{1 - \Gamma^*(M) - \tau}}, \quad \Gamma^*(M) + \tau < 1. \tag{37}$$

respectively, then the probabilistic bounds, (29) and (32), are controlled to be no more than ε.

6 Concluding Remarks

In this work, we propose a robust model for the uncertain linear program with correlated uncertainties related to a causal ARMA process. (Without difficulties, the results can also be generalized to the case where correlated uncertainties related to an ARMA process with non-zero mean or an ARIMA process.) Under the l_2-norm and some general symmetrically distributed assumptions on the primitive uncertainties $\{\tilde{\omega}_k\}$, we establish the probabilistic guarantees for feasibility and provide a numerical method for calculating $\Gamma(M)$ approximately, yielding an explicit ways to select M, Ω and ϵ to control the tractability, the optimality and the robustness of the proposed robust model.

References

1. Ben-Tal, A., Nemirovski, A.: Robust convex optimization. Math. Oper. Res. 23, 769–805 (1998)
2. Ben-Tal, A., Nemirovski, A.: Robust solutions to uncertain linear programs. Oper. Res. Let. 25, 1–13 (1999)
3. Ben-Tal, A., Nemirovski, A.: Robust solutions of Linear Programming problems contaminated with uncertain data. Math. Prog. 88, 411–424 (2000)
4. Ben-Tal, A., Goryashko, A., Guslitzer, E., Nemirovski, A.: Adjustable robust solutions of uncertain linear programs. Math. Prog. 99, 351–376 (2004)
5. Bertsimas, D., Pachamanova, D., Sim, M.: Robust Linear Optimization under General Norms. Oper. Res. Let. 32, 510–516 (2004)
6. Bertsimas, D., Sim, M.: Price of Robustness. Oper. Res. 52, 35–53 (2004)
7. Bertsimas, D., Sim, M.: Tractable Approximation to Robust Conic Optimization Problems. Math. Prog. 107, 5–36 (2006)
8. Bertsimas, D., Thieley, A.: Robust and Data-Driven Optimization: Modern Decision-Making Under Uncertainty. Working paper. MIT Press, Cambridge (2006)
9. Box, G., Jenkins, G.M., Reinsel, G.: Time series analysis: Forecasting and Control. Prentice-Hall, Englewood Cliffs, NJ (1994)
10. Brochwell, P.J., Davis, R.A.: Time series: theory and methods. Springer, New York (1991)
11. Chen, X., Sim, M., Sun, P.: A Robust Optimization Perspective of Stochastic Programming. Working Paper (2005)
12. El-Ghaoui, L., Lebret, H.: Robust solutions to least-square problems to uncertain data matrices. SIAM J. Matrix Anal. Appl. 18, 1035–1064 (1997)
13. El-Ghaoui, L., Oustry, F., Lebret, H.: Robust solutions to uncertain semidefinite programs. SIAM J. Optim. 9, 33–52 (1998)
14. Mills, T.C.: The Econometric Modelling of Financial Time Series, 2nd edn. Cambridge University Press, Cambridge (1999)
15. Soyster, A.L.: Convex programming with set-inclusive constraints and applications to inexact linear programming. Oper. Res. 21, 1154–1157 (1973)
16. Wilkinson, J.H.: The algebraic eigenvalue problem. Clarendon Press, Oxford (1965)

An Efficient Algorithm for Solving the Container Loading Problem

Wenqi Huang and Kun He*

Theoretical Computer Science Institute, College of Computer Science,
Huazhong University of Science and Technology, Wuhan 430074, China
wqhuang@mail.hust.edu.cn, brooklet60@gmail.com

Abstract. We present a new algorithm for the three-dimensional packing problem with a single container to be loaded. Deviates from the traditional approaches, it uses a principle —— "largest caving degree" such that items are packed as closely as possible. Then, it incorporates a backtracking strategy that gains a good trade-off between solution quality and computation time. We tested our algorithm using all the 47 related benchmarks from the OR-Library that have no orientation constraints. Experiments indicate the algorithm's good quality. The container volume utilization, achieved within comparable time, is 94.6% on average. This significantly improves current best-known results in the literature by 3.6%.

Keywords: Heuristic; Cargo Loading; Cutting and Packing.

1 Introduction

The cutting and packing (C&P) problem is a well-motivated research issue in combinatorial optimization area. Theoretically, it is NP-hard in the strong sense and is very difficult to solve [1]. Practically, it has a wide variety of applications in the real world. E.g., the efficient cargo transportation is of high economic value in logistics, and near optimal cargo loading leads to lower costs all over the distribution system. A C&P problem usually corresponds to a job scheduling problem. E.g., the two-dimensional (2D) bin packing problem is equivalent to a parallel job scheduling problem on identical machines located on a 2D mesh [2]. Therefore, methods for the C&P problem are also useful for the scheduling problem, which is another kind of important problem both in theory and practice.

In this paper, we consider a typical three-dimensional (3D) C&P problem: the container loading problem. We assume there is no orientation constraint, i.e. the item may be rotated by 90^o before it is packed. This complicates the problem since the number of possible packing is now a factor of 6^n larger for n items. Yet, it may allow better packing, and small modification on algorithms could easily eliminate this scenario.

Heuristic approaches are often the feasible options for the 3D C&P problem. Basic methods are wall building [1,3,4], layering [5], guillotine cutting [6], block arrangement [7,8,9], etc. Tree-search [1,7], tabu search [8], simulated annealing [8],

* Corresponding author.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 396–407, 2007.

or genetic search [9] is incorporated to improve the solution's quality. Sometimes, parallel methods are used to shorten the computation time [8,9]. Based on the wall building and layering methods, Lim et al. proposed a basic algorithm MFB and a strengthened algorithm MFB_L [4]. They tested on 760 benchmarks from the OR-Library [10]. The results of MFB_L were about 3% better on average than that in [3], which was the only available work they found containing packing results for the frontier 700 benchmarks.

We proposed a new approach for the 3D packing problem. The inspiration originates from an old adage "gold corner, silver side and grass belly" in Chinese I-go. The adage has concentrated human's experience and wisdom formed in the last thousand years. It indicates that corner worth most while center worth dirt on the chessboard. Moreover, we improved the idea with "diamond cave". Based on a principle of "largest caving degree" [11], a packing item should always occupy a corner, or even a cave, bounded by surfaces of the packed items or the container. In this way, items are packed as closely as possible. We tested our algorithm using all the 47 benchmarks from the OR-Library [10] that allow rotations. Results generated are compared with those in [4] which tested the same data. Experiments indicate that we are able to achieve an average packing utilization of 94.6%. To our knowledge, this significantly improves the best-known results by 3.6%.

2 Problem Specification

The container loading problem is to pack a subset of rectangular items (e.g. boxes) into a single rectangular container with fixed dimensions. The objective is to find a feasible solution that maximizes the total volume of the packed items, i.e. maximizes the container's volume utilization. A solution is feasible if: ① each item is packed completely in the container; ② there is no overlapping between any two items; ③ each item is placed parallel to the surfaces of the container.

Given a container with dimensions (L,W,H) and a set of n items $S=\{(l_1,w_1,h_1),...,(l_n,w_n,h_n)\}$. Variables l_i, w_i and h_i are the three dimensions of item i. Without loss of generality, suppose all the variables are plus integer numbers. Consider the container embedded into a three-dimensional Cartesian reference frame. The lower-left-near corner coincides with the origin and the upper-right-far corner coincides with point (L,W,H), as Fig. 1 shows. Let $\delta_i \in \{0,1\}$ denotes whether item i is packed into the container. If item i has been packed, let (x_{i1},y_{i1},z_{i1}) and (x_{i2},y_{i2},z_{i2}) denote the coordinates of its lower-left-near corner and upper-right-far corner respectively. Then, the problem can be formulized as follows:

$$\max \sum_{i=1}^{n} l_i w_i h_i \delta_i$$

Subject to

(1) $(x_{i2}-x_{i1},y_{i2}-y_{i1},z_{i2}-z_{i1}) \in \{(l_i,w_i,h_i),(w_i,l_i,h_i),(l_i,h_i,w_i),(h_i,l_i,w_i),(h_i,w_i,l_i),(w_i,h_i,l_i)\}$;

(2) $\max(\ \max(x_{i1},x_{j1})-\min(x_{i2},x_{j2}),\max(y_{i1},y_{j1})-\min(y_{i2},y_{j2}),$
$\max(z_{i1},z_{j1})-\min(z_{i2},z_{j2})\ )\delta_i\delta_j \geq 0$;

(3) $0 \le x_{i1} < x_{i2} \le L$, $0 \le y_{i1} < y_{i2} \le W$, $0 \le z_{i1} < z_{i2} \le H$;

(4) $\delta_i \in \{0,1\}$;

Where $i, j = 1, 2, ..., n$, $i \ne j$. Item i has six orientations whose dimensions on x-, y-, and z-axis are (l_i, w_i, h_i), (w_i, l_i, h_i), (l_i, h_i, w_i), (h_i, l_i, w_i), (h_i, w_i, l_i) and (w_i, h_i, l_i), with their orientation number from one to six respectively. Constraints (1) to (3) are corresponded to the three conditions of a feasible solution.

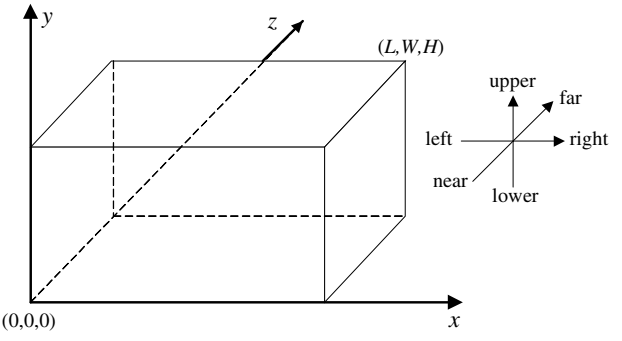


Fig. 1. Coordinate system

3 Algorithm Description

The main idea of our algorithm is to always do a Real Corner Occupying Action (RCOA) with the largest caving degree. A RCOA is an action that places an item at a corner where three orthogonal surfaces meet. Caving degree is a value defined to judge how close a packing item is to other items filled in already.

3.1 Definitions

Definition 1 (Real Corner). It is an unoccupied space where three orthogonal surfaces meet and form a corner. The surface could be an interior surface of the container, or an exterior surface of a packed item. The three surfaces intersect each other on surface or edge.

The six surfaces of the container can be regarded as six items with unit length for the third dimension, and are packed at the corresponding positions in the beginning. The three surfaces of a real corner belong to different items. A corner is denoted by its coordinates (x,y,z) and its corner direction. There are eight different corner directions. They are corresponded with the directions of the eight corners, formed by the coordinate planes in the eight quadrants of the 3D Cartesian reference frame.

Theorem 1. There are at most two real corners formed by any three items.

Proof. According to the real corner's definition, any two items of a real corner must intersect each other on surface or edge.

① If the two items intersect on surface, as in Fig.2 (a) item *i* and item *k* or item *j* and item *k* show, there are at most four intersecting edges. And at each direction, there are at most two parallel edges. The surface of the third item must be orthogonal with the intersecting edges. The third item could intersect at most two parallel edges to form corners, as in Fig.2 (a) item *j* shows. Or the third item could intersect one edge with its two parallel surfaces, as in Fig.2 (a) item *i* shows. So, there are at most two real corners formed.

② If the two items intersect on edge, as in Fig.2 (b) item *i* and item *j* show, there are at most two pairs of intersecting surfaces. Each pair of surfaces is orthogonal with each other and belongs to the two items respectively. So, there are at most two real corners formed with the third item, as Fig.2 (b) item *k* shows.

So, the theorem is true. □

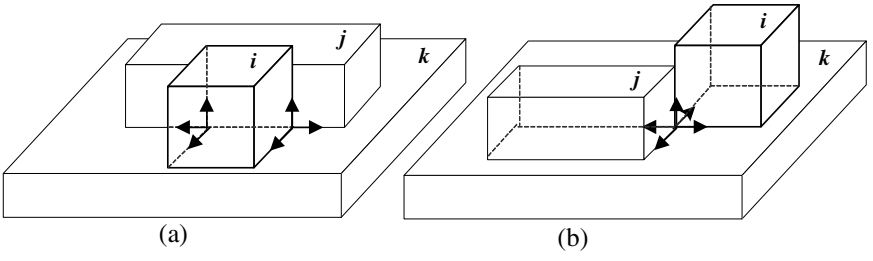


Fig. 2. Real corners formed by three items

Definition 2 (Real Corner Occupying Action, RCOA). A packing action includes three aspects: which item to be packed, which position to be placed, and which item orientation to be set. An action is called a RCOA if the packing item occupies a real corner.

A RCOA is denoted by the occupied corner's coordinate and direction, as well as the packing item's id and orientation.

Definition 3 (Distance of Two Point Sets). It's an infimum on distances of two points chosen from the two sets respectively.

$$d(A,B) = \inf\{d(a,b): a \text{ in } A, b \text{ in } B\}. \quad (1)$$

Definition 4 (Distance of Two Items, d_{ij}). Regarding each item as a point set, their distance is the Minkowski distance between the two sets.

The distance between item *i* and *j* is $d_{ij}=dx_{ij}+dy_{ij}+dz_{ij}$, as Fig. 3 shows, where

$$dx_{ij}=\max(|x_{ic}-x_{jc}|-\frac{l_i+l_j}{2},0),$$

$$dy_{ij}=\max(|y_{ic}-y_{jc}|-\frac{w_i+w_j}{2},0),$$

$$dz_{ij}=\max(|z_{ic}-z_{jc}|-\frac{h_i+h_j}{2},0),$$

and (x_{ic},y_{ic},z_{ic}), (x_{jc},y_{jc},z_{jc}) are the central coordinates of item i and item j respectively.

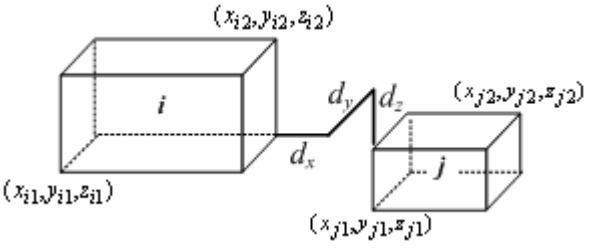


Fig. 3. Distance of two items

Definition 5 (Paste Number, k_i). It's the number of pasted surfaces for a RCOA packing item. $k_i\in\{3,4,5,6\}$. A surface is pasted if the intersect area with another surface is larger than 0.

Definition 6 (Paste Ratio, p_i). It's the total pasted area of a RCOA packing item, divided by the total surface area of the item. $p_i\in(0,1]$

Definition 7 (Distance of a RCOA, d_i). Aligned with the k pasted plane and their normal directions (pointing to the space the packing item is in), a space Ω is enclosed with 6-k dimensions. Distance of a RCOA is the minimum distance between the packing item and a set of packed items. An item, including the six items forming the container, is considered in the set if its intersection with space Ω is not null, the intersect area or volume is larger than 0, and it does not paste the packing item. Let $d_i=0$ when k=6.

$$d_i=\begin{cases}\min(d_{ij}, j\cap\Omega\neq\varnothing\wedge j\cap i=\varnothing) & k=\{3,4,5\}\\ 0 & k=6\end{cases}. \quad (2)$$

Fig. 4 shows the space Ω for packing item i with its upper, down, left and far surfaces pasted (k=4), and its occupied corner is at the upper-left-far vertex. When computing d_i , item a pastes with item i and it is not considered, while item b and item c are considered; the near surface and right surface of the container are considered too.

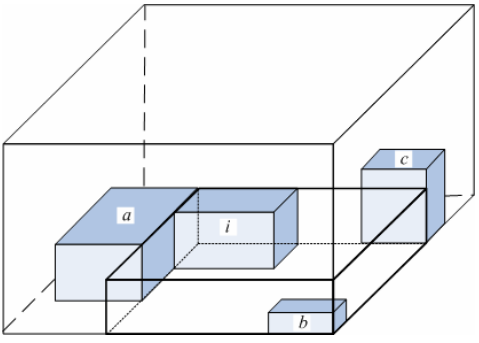


Fig. 4. Space Ω

Definition 8 (Caving Degree, C_i). It is defined on a RCOA to judge how close the packing item is with other items filled in already.

$$C_i = \mu(k_i + p_i) - \frac{d_i}{\sqrt[3]{l_i w_i h_i}}, \ \mu = \max(L, W, H) \ . \tag{3}$$

The larger the caving degree is, the better a RCOA is. A good caving degree is determined on the largest paste number, the largest paste ratio, and the smallest distance of a RCOA in dictionary order.

3.2 Packing Procedure

In the main procedure, a packing method is invoked recursively and items are packed one by one from outside to the container. Suppose at a packing step, some items have been packed and some remain outside. The packing method searches all feasible RCOAs by iterating all free real corners, all free items and six item orientations. Thereafter, based on the largest caving degree principle, an item is selected to pack in an appointed position with an appointed orientation. This action makes the packing item as closely as possible to other packed items.

So, one item is packed at each packing step. The inside items are increasing and the outside items are decreasing until the final packing step comes. At the final step, all items have been packed into the container without overlapping, or none of the remainders can be packed in.

Above procedure is the greedy B_0 algorithm. The strengthened B_1 and B_2 algorithms add backtracking strategy on B_0 to get higher solution quality by sacrificing the computation time.

3.3 Ranking Scheme

In B_0 algorithm, the ranking scheme is employed to select a RCOA at each packing step, which can be formulated as follows:

Main criteria: largest caving degree.
Tiebreaker 1: smallest length of the long dimension.
Tiebreaker 2: smallest length of the middle dimension.

Tiebreaker 3: smallest length of the short dimension.
Tiebreaker 4: smallest coordinate z of the gravity.
Tiebreaker 5: smallest coordinate y of the gravity.
Tiebreaker 6: smallest coordinate x of the gravity.
Tiebreaker 7: smallest orientation number.

Tiebreakers 1 to 3 are aimed at packing potentially awkward items early on. Tiebreakers 4 to 7 primarily serve to produce a definitive selection when the former rules lead to identical scores.

3.4 Basic Algorithm

There is no item in the container in the beginning. The six surfaces of the container are regarded as six items, and are packed at the corresponding positions. Then, the packing method is invoked recursively. At the end of each packing step, the corners the packing item consumed are deleted from the free corner map, and the corners the packing item created are added to the free corner map. An update procedure is used to find the consumed and created corners and to update the free corner map. E.g., in Fig. 5, an item is packed at the lower-left-near corner of the container. Its dimensions on x-, y- and z-axis are l, w and h respectively. The corner it consumed is $(0,0,0,1)$, and the corners it created are $(l,0,0,1)$, $(0,w,0,1)$ and $(0,0,h,1)$.

Algorithm B_0

Input: a container denoted by (L,W,H); a free item list with each item denoted by (l_i,w_i,h_i).

```
init container(L,W,H);
 packing(freeItemList){
  if freeItemList is null, return;
  for each free real corner in freeCornerMap{
    for each outside item in freeItemList{
      for each item orientation {
        check current RCOA's validity;
        if it's a feasible RCOA,
          compute its caving degree;
      }
    }
  }
  if current feasible RCOAs are not null {
    choose a best RCOA to do by the ranking scheme;
    move the selected item from freeItemList to
usedItemList;
    update freeCornerMap;
    packing(freeItemList);
  }
}
```

Output: a list of packed items in usedItemList, each with coordinates of its lower-left-near and upper-right-far corners; container volume utilization.

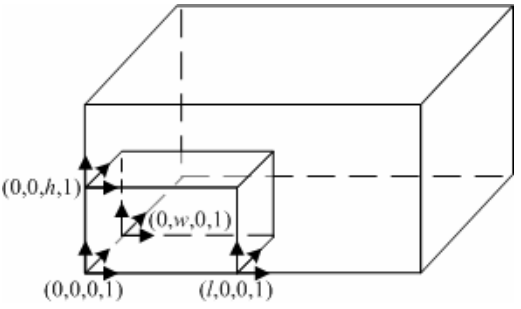


Fig. 5. Corners consumed and created

3.5 Strengthened Algorithm

Backtracking strategy is incorporated to further improve the packing utilization. At each packing step of algorithm B_1, instead of choosing a RCOA with the largest caving degree, we choose a RCOA with the largest volume utilization that pseudo computed by B_0. "Pseudo computation" means that the outside items are packed temporarily by B_0 so as to get corresponding container volume utilization for the candidate RCOA item, and are removed from the container later. In case of ties, the same tiebreakers as that of algorithm B_0 are used. Figure 5 depicts the evolvement of the volume utilization at each packing step, which is produced by algorithm B_1 on two cases of test file 9 from the OR-Library.

All feasible RCOAs can be regarded as a neighborhood of current solution, and the neighbor with the largest volume utilization is selected as the next step. Instead of terminating when there is no neighbors in B_1, algorithm B_2 will terminate when an optimal filling is found, or the volume utilization does not improve at the next step. As can be seen in Fig. 6, case 2 will terminate at step 6, and case 15 will terminate at step 4. This gives a good trade-off between solution quality and computation time.

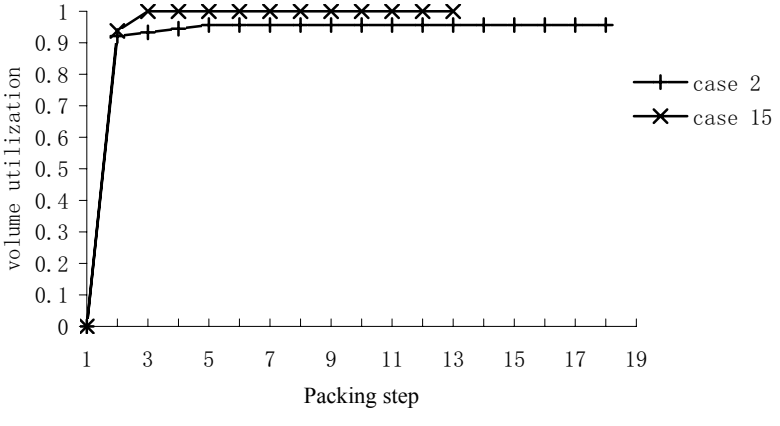


Fig. 6. Volume utilization at each packing step

4 Experimental Results

We have implemented our algorithms in java and run them on a 1.7GHz IBM notebook. Benchmarks are from OR-Library [10], which is a collection of test data sets for a variety of Operations Research (OR) problems. There are nine sets of test data with 762 test cases for the 3D packing problem. The 47 cases in file 9 allow rotations and are considered here. They are weakly heterogeneous problems, and their item types are from two to five.

Figure 7 refers to the volume utilization of algorithm B_0, where x-axis represents the test case index and y-axis represents the container volume utilization. Figure 8 contains a curve graph detailing the computation time of B_0 involved in generating the solutions. The average container volume utilization is 87.2% on average. Moreover, algorithm B_0 is very fast with average time 1.13 seconds. It takes 0.22 to 2.7 seconds to pack one container, except 4.22 seconds for the 45th case and 11 seconds for the 46th case. Note that in [4], the authors did not compute the 45th to the 47th cases.

Figure 9 refers to the volume utilization of algorithm B_2, where x-axis represents the test case index and y-axis represents the container volume utilization. Figure 10 contains a curve graph detailing the computation time of B_2 involved in generating the solutions. The average container volume utilization is 94.6% on average. The backtracking strategy enables B_2 to improve the average volume utilization by 7.4%. Moreover, algorithm B_2 is fairly fast with average time 876 seconds. It takes 0.25 seconds to 48 minutes to pack one container, except 2.66 hours for the 45th case and 3.85 hours for the 46th case. Note that in [4], the authors did not compute the 45th to the 47th cases.

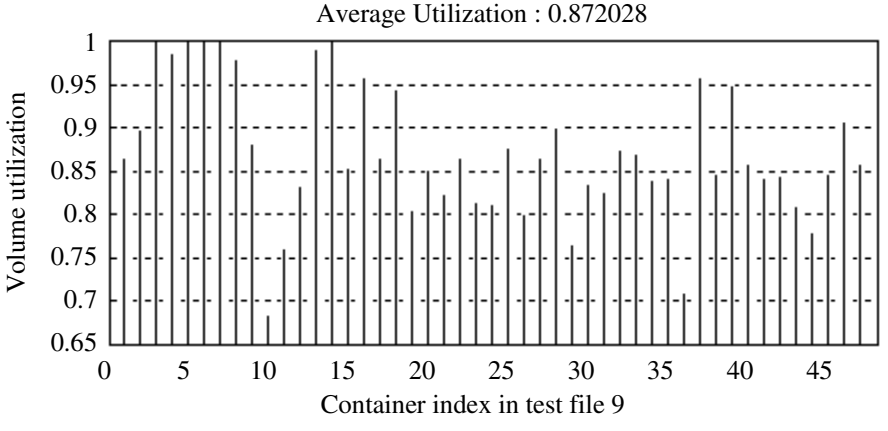


Fig. 7. Volume utilization for algorithm B_0

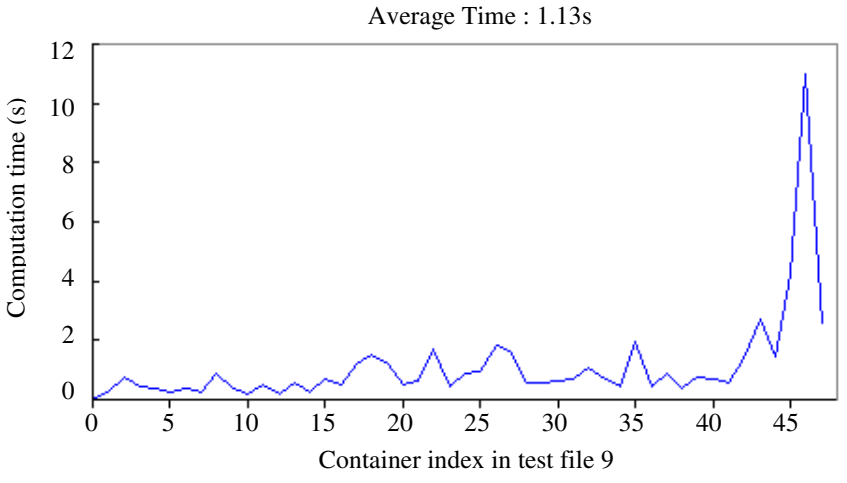


Fig. 8. Computation time for algorithm B_0

In [4], the authors computed the anterior 44 test cases of file 9. What we did is 3.6% better than theirs judged by these cases. Table 1 shows the comparison. As can be seen, the average volume utilization is 87.21% of B_0 as opposed to 83.68% of MFB_L [4] applied, and the average volume utilization is 94.59% of B_2 as opposed to 91% of MFB_L [4] applied. The B_0 and B_2 algorithms perform better than that of MFB and MFB_L respectively.

As for the time, since they used a small-scale computer while we used a personal PC, our algorithms are at least not slower than theirs. And as Fig. 8 and Fig. 9 show, all results are obtained within reasonable times.

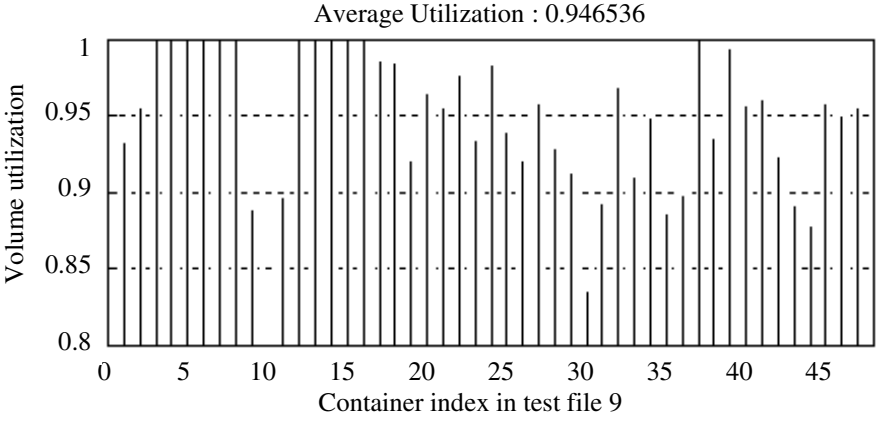


Fig. 9. Volume utilization for algorithm B_2

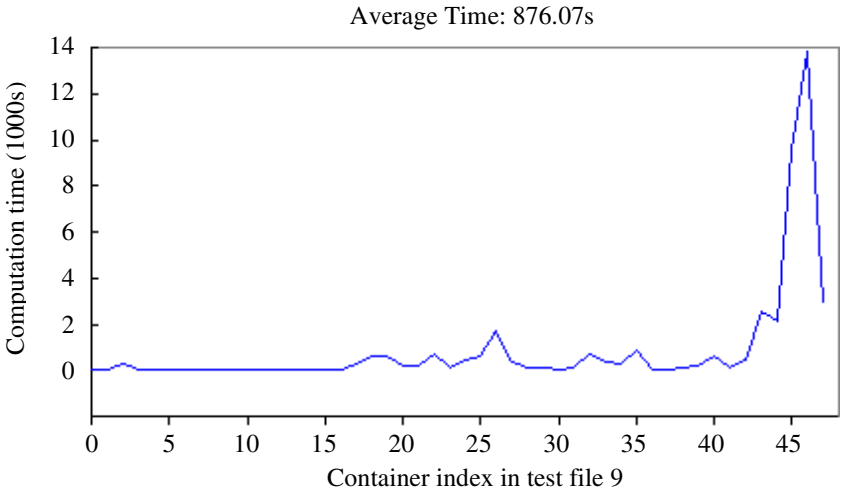


Fig. 10. Computation time for algorithm B_2

Table 1. Comparisons with MFB and MFB_L algorithms

Comparisons	MFB	B_0	MFB_L	B_2
Test computer	Unix Alpha 600 MHz	Windows 1.7GHz	Unix Alpha 600 MHz	Windows 1.7GHz
Average utilization	0.8368	0.8721	0.91	0.94595
Average time	2-3s	0.80s	205s	337.78s

5 Conclusions and Future Work

We presented a new approach, which is a heuristic in nature, for the three-dimensional container packing problem. The rotation of items is allowed. The approach is efficient yet not complicated. Different from the previous approaches, it uses a conception of caving degree to judge how good a real corner occupying action is. The approach achieved a high average packing utilization of 94.6%, computed within rather short times, using all the 47 related benchmarks from the OR-Library test suite. This is 3.6% better than the best-known results. Tests show that our approach is comparable, in terms of the container volume utilization, to other approaches reported.

There are some factors that can be considered aftertime, such as orientation restrictions, load bearing strength of items, loading stability, grouping of items, and weight distribution. Other variants of the 3D cutting and packing problem, e.g. strip packing and bin packing, may use this approach for reference. More work will be done on these in the future.

Acknowledgments. This work was supported by National Natural Science Foundation of China (Grant No. 10471051) and by the NKBRPC (Grant No. G2004CB318000).

References

1. David, P.: Heuristics for the Container Loading Problem. European Journal of Operational Research 141, 382–392 (2002)
2. Zhang, G.C.: A 3-Approximation Algorithm for Two-Dimensional Bin Packing. Operations Research Letters 33, 121–126 (2005)
3. Bischoff, E.E., Ratcliff, M.S.W.: Issues in the Development of Approaches to Container Loading. OMEGA: The International Journal of Management Science 23, 377–390 (1995)
4. Lim, A., Rodrigues, B., Wang, Y.: A Multi-faced Buildup Algorithm for Three-Dimensional Packing Problems. OMEGA: The International Journal of Management Science 31, 471–481 (2003)
5. Loh, T.H., Nee, A.Y.C.: A Packing Algorithm for Hexahedral Boxes. In: Proceedings of the Conference of Industrial Automation, Singapore, pp. 115–126 (1992)
6. Morabito, R., Arenales, M.: An AND/OR Graph Approach to the Container Loading Problem. International Transactions in Operational Research 1, 59–73 (1994)
7. Eley, M.: Solving Container Loading Problems by Block Arrangements. European Journal of Operational Research 141, 393–409 (2002)
8. Mack, D., Bortfeldt, A., Gehring, H.: A Parallel Local Algorithm for the Container Loading Problem. International Transactions in Operational Research 11, 511–533 (2004)
9. Gehring, H., Bortfeldt, A.: A Parallel Generic Algorithm for Solving the Container Loading Problem. European Journal of Operational Research 131, 143–161 (2001)
10. OR-Library: http://mscmga.ms.ic.ac.uk/jeb/orlib/thpackinfo.html
11. Huang, W.Q., Xu, R.C.: Introduction to the Modern Theory of Computation — Background, Foreground and Solving Method for the NP-hard Problem (in Chinese). Science, Beijing

A Bijective Code for k-Trees with Linear Time Encoding and Decoding

Saverio Caminiti, Emanuele G. Fusco, and Rossella Petreschi

Computer Science Department
University of Rome "La Sapienza", via Salaria, 113-00198 Rome, Italy
{caminiti, fusco, petreschi}@di.uniroma1.it

Abstract. The problem of coding labeled trees has been widely studied in the literature and several bijective codes that realize associations between labeled trees and sequences of labels have been presented. k-trees are one of the most natural and interesting generalizations of trees and there is considerable interest in developing efficient tools to manipulate this class, since many NP-Complete problems have been shown to be polynomially solvable on k-trees and partial k-trees. In 1970 Rényi and Rényi generalized the Prüfer code to a subset of labeled k-trees; subsequently, non redundant codes that realize bijection between k-trees (or Rényi k-trees) and a well defined set of strings were produced. In this paper we introduce a new bijective code for labeled k-trees which, to the best of our knowledge, produces the first encoding and decoding algorithms running in linear time with respect to the size of the k-tree.

1 Introduction

The problem of coding labeled trees, also called Cayley's trees after the celebrated Cayley's theorem [6], has been widely studied in the literature. Coding labeled trees by means of strings of vertex labels is an interesting alternative to the usual representations of tree data structures in computer memories, and it has many practical applications (e.g. Evolutionary algorithms over trees, random trees generation, data compression, and computation of forest volumes of graphs). Several different bijective codes that realize associations between labeled trees and strings of labels have been introduced, see for example [7,9,10,17,18,19,20]. From an algorithmic point of view, the problem has been investigated thoroughly and optimal encoding and decoding algorithms are known for most of these codes [4,5,7,9,19].

k-trees are one of the most natural and interesting generalizations of trees (for a formal definition see Section 2) and there is considerable interest in developing efficient tools to manipulate this class of graphs. Indeed each graph with treewidth k is a subgraph of a k-tree, and many NP-Complete Problems (e.g. Vertex Cover, Graph k-Colorability, Independent Set, Hamiltonian Circuit, etc.) have been shown to be polynomially solvable when restricted to graphs of bounded treewidth. We suggest the interested reader to see [2,3].

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 408–420, 2007.

In 1970 Rényi and Rényi [21] generalized Prüfer's bijective proof of Cayley's theorem to code a subset of labeled k-trees (Rényi k-trees). They introduced a redundant Prüfer code for Rényi k-trees and then characterized the valid codewords. Subsequently, non redundant codes that realize bijection between k-trees (or Rényi k-trees) and a well defined set of strings were produced [8,11] together with encoding and decoding algorithms. Attempts have been made to obtain an algorithm with linear running time for the redundant Prüfer code [15], however to the best of our knowledge, no one has developed linear time algorithms for non redundant codes.

In this paper we present a new bijective code for k-trees, together with encoding and decoding algorithms, whose running time is linear with respect to the size of the encoded k-tree.

2 Preliminaries

In this section we recall the concepts of k-trees and Rényi k-trees and highlight some properties related to this class of graphs.

Let us call q-clique a clique on q nodes and $[a, b]$ the interval of integer from a to b (a and b included).

Definition 1. *[14] An unrooted k-tree is defined in the following recursive way:*

1. *A complete graph on k nodes is a k-tree.*
2. *If $T'_k = (V, E)$ is a k-tree, $K \subseteq V$ is a k-clique and $v \notin V$, then $T_k = (V \cup \{v\}, E \cup \{(v, x) \mid x \in K\})$ is also a k-tree.*

By construction, a k-tree with n nodes has $\binom{k}{2} + k(n-k)$ edges, $n-k$ cliques on $k+1$ nodes, and $k(n-k)+1$ cliques on k nodes. Since every T_k with k or $k+1$ nodes is simply a clique, in the following we will assume $n \geq k+2$.

In a k-tree nodes of degree k are called k-leaves. Note that the neighborhood of each k-leaf forms a clique and then k-leaves are simplicial nodes. A rooted k-tree is a k-tree in which one of its k-cliques $R = \{r_1, r_2, \ldots, r_k\}$ is distinguished; this clique is called the root.

Remark 1. In an unrooted k-tree T_k there are at least two k-leaves; when T_k is rooted at R at least one of those k-leaves does not belong to R (see [21]). Since k-trees are perfect elimination order graphs [22], when a k-leaf is removed from a k-tree the resulting graph is still a k-tree and at most one of its adjacent nodes may became a k-leaf, unless the resulting k-tree is nothing more than a single clique.

In this paper we will deal with k-trees labeled with distinct integer values in $[1, n]$. In Figure 1(a) an example of k-tree with $k = 3$ and 11 nodes labeled with integers in $[1, 11]$ is given. The same k-tree, rooted at $R = \{2, 3, 9\}$, is given in Figure 1(b).

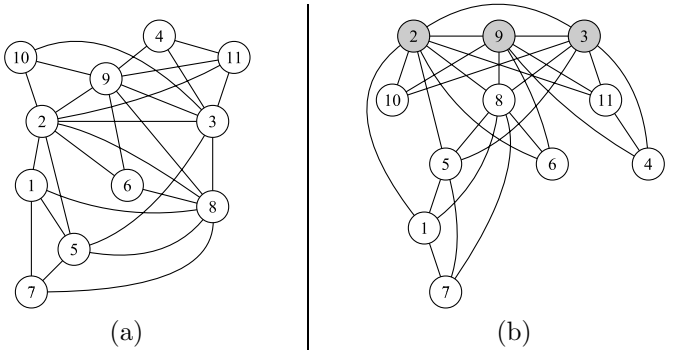


Fig. 1. (a) An unrooted 3-tree T_3 on 11 nodes. (b) T_3 rooted at the clique $\{2, 3, 9\}$.

Let us call $\mathcal{T}_k^n$ the set of labeled k-trees with n nodes. It is well known that [1,12,16,21]:

$$|\mathcal{T}_k^n| = \binom{n}{k}(k(n-k)+1)^{n-k-2}$$

When $k = 1$ the set $\mathcal{T}_1^n$ is the set of Cayley's trees and $|\mathcal{T}_1^n| = n^{n-2}$, i.e. the well-known Cayley's theorem.

Definition 2. *[21] A Rényi k-tree R_k is a k-tree with n nodes labeled in $[1, n]$ rooted at the fixed k-clique $R = \{n-k+1, n-k+2, \ldots, n\}$.*

It has been proven [16,21] that:

$$|\mathcal{R}_k^n| = (k(n-k)+1)^{n-k-1}$$

where $\mathcal{R}_k^n$ is the set of Rényi k-trees with n nodes. It is obvious that $\mathcal{R}_k^n \subseteq \mathcal{T}_k^n$; the equality holds only when $k = 1$ (i.e. the set of labeled trees rooted in n is equivalent to the set of unrooted labeled trees) and when $n = k$ or $n = k + 1$ (i.e. the k-tree is a single clique).

3 Characteristic Tree

Here we introduce the *characteristic tree* $T(R_k)$ of a Rényi k-tree that will be used to design our algorithm for coding a generic k-tree.

Let us start by introducing the *skeleton* of a Rényi k-tree. Give a a Rényi k-tree R_k its skeleton $S(R_k)$ is defined according to the definition of k-trees:

1. if R_k is a single k-clique R, $S(R_k)$ is a tree with a single node R;
2. let us consider a k-tree R'_k, its skeleton $S(R'_k)$, and a k-clique K in R'_k. If R_k is the k-tree obtained from R'_k by adding a new node v attached to K, then $S(R_k)$ is obtained by adding to $S(R'_k)$ a new node $X = \{v\} \cup K$ and a new edge (X, Y) where Y is the node of $S(R'_k)$ that contains K, at minimum distance from the root.

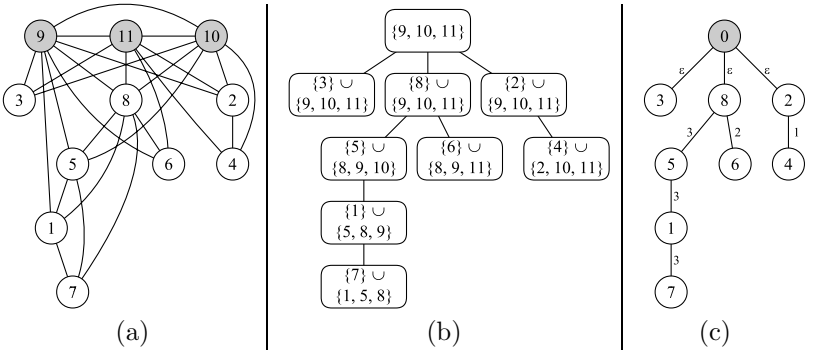


Fig. 2. (a) A Rényi 3-tree R_3 with 11 nodes and root $\{9, 10, 11\}$. (b) The skeleton $S(R_3)$, with nodes $\{v\} \cup K$. (c) The characteristic tree $T(R_3)$.

$S(R_k)$ is well defined, in particular it is always possible to find a node Y containing K in $S(R'_k)$ because K is a clique in $S(R'_k)$. Moreover Y is unique, indeed it is easy to verify that if two nodes in $S(R'_k)$ contain a value v, their lower common ancestor still contains v. Since it holds for all $v \in K$, there always exists a unique node Y containing K at minimum distance from the root.

The characteristic tree $T(R_k)$ is obtained by labeling nodes and edges of $S(R_k)$ as follows:

1. each node $\{v\} \cup K$ with parent X is labeled v. The node R is labeled 0;
2. each edge from node $\{v\} \cup K$ to its parent $\{v'\} \cup K'$ is labeled with the index of the node in K' (considered as an ordered set) that does not appear in K. When the parent is R the edge is labeled ε.

In Figure 2 a Rényi 3-tree with 11 nodes, its skeleton and its characteristic tree are shown.

It is easy to reconstruct a Rényi k-tree R_k from its characteristic tree $T(R_k)$ since the characteristic tree is well defined and conveys all information needed to rebuild the skeleton of R_k. We point out that there will always be one, and only one, node in K' that does not appear in K (see 2. in the definition of $T(R_k)$). Indeed, v' must appear in K, otherwise $K' = K$ and then the parent of $\{v'\} \cup K'$ would contain K and this would contradict each node in $S(R_k)$ being attached as closely as possible to the root (see 2. in the definition of $S(R_k)$).

Remark 2. For each node $\{v\} \cup K$ of $S(R_k)$ each $w \in K - R$ appears as label of a node in the path from v to 0 in $T(R_k)$.

A linear time algorithm to reconstruct R_k from $T(R_k)$ with a simple traversal of the tree is detailed in Section 6. This algorithm avoids the explicit construction of $S(R_k)$.

Let us consider $\mathcal{Z}_k^n$, the set of all trees with $n-k+1$ nodes labeled with distinct integers in $[0, n-k]$ in which all edges incident on 0 have label ε and all other

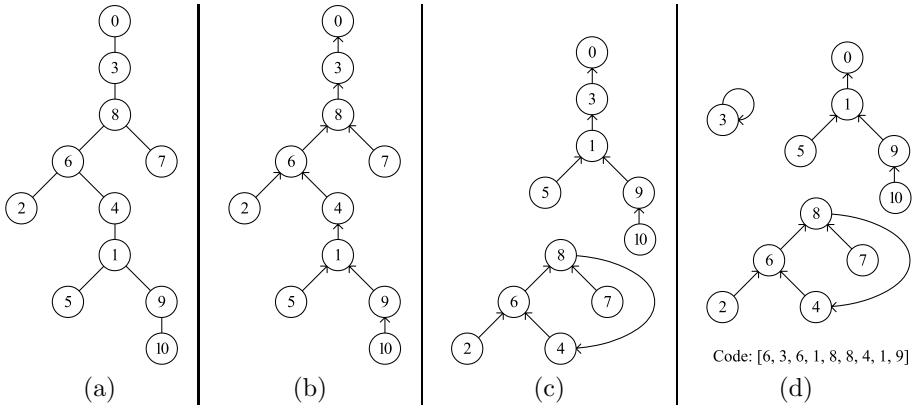


Fig. 3. (a) A simple tree T with 11 nodes labeled in $[0, 10]$. (b) The functional digraph G at the beginning of the Dandelion encoding. (c) G after the first swap $p(1) \leftrightarrow p(8)$. (d) G at the end of the encoding after the swap $p(1) \leftrightarrow p(3)$, together with the code string.

edges take a label from $[1, k]$. The association between a Rényi k-tree and its characteristic tree is a bijection between $\mathcal{R}_k^n$ and $\mathcal{Z}_k^n$. Obviously, for each Rényi k-tree its characteristic tree belongs to $\mathcal{Z}_k^n$, and this association is invertible. In Section 4 we will show that $|\mathcal{Z}_k^n| = |\mathcal{R}_k^n|$; this will imply the bijectivity of this association.

Our characteristic tree coincides with the *Doubly Labeled Tree* defined in a completely different way in [13] and used in [8]. Our new definition gives us the right perspective to build the tree in linear time, as will be shown in Section 5.

4 Generalized Dandelion Code

As stated in the introduction, many codes producing bijection between labeled trees with n nodes and strings of length $n-2$ have been presented in the literature. Here we show a generalization of one of these codes, because we need to take into account labels on edges. We have chosen Dandelion code due to special structure of the code strings it produces. This structure will be crucial to ensure the bijectivity at the end of the encoding process of a k-tree (see Section 5 Step 3).

Dandelion code was originally introduced in [10], but its poetic name is due to Picciotto [19]. Our description of this code is based on [5] where linear time coding and decoding algorithms are detailed.

The basic idea behind Dandelion code is to root the tree in 0 and to ensure the existence of edge $(1, 0)$. A tree T with n nodes labeled in $[0, n-1]$ is transformed into a digraph G, such that 0 has no outgoing edges and each node $v \neq 0$ has one outgoing edge; the outgoing edge of 1 is $(1, 0)$. For each v in T, let $p(v)$ be the parent of v in T rooted in 0. G is obtained starting with the edge set $\{(v, p(v)) | v \in V - \{0\}\}$, i.e. initially G is the whole tree with edges oriented

upwards to the root (see Figure 3(b)). Now we have to shrink the path between 1 and 0 into a single edge. This can be done by iteratively swapping $p(1)$ with $p(w)$ where $w = max\{u \in path(1,0)\}$ (see Figure 3(c) and 3(d)). The code string will be the sequence of $p(v)$ for each v from 2 to $n-1$ in the obtained G.

Since the trees we are dealing with have labels on both nodes and edges, we need to modify the Dandelion code to enable it to hold edge information. In particular, we have to specify what happens when two nodes u and v swap their parents. Our algorithm ensures that the label of the edge $(v, p(v))$ remains associated to $p(v)$. More formally, the new edge $(u, p(v))$ will have the label of the old edge $(v, p(v))$ and similarly the new edge $(v, p(u))$ will have the label of the old edge $(u, p(u))$.

A further natural generalization is to adopt two nodes r and x as parameters instead of the fixed 0 and 1.

In Program 1 the pseudo-code for the generalized Dandelion Code is given; $l(u, v)$ is the label of edge (u, v). The tree T is considered as rooted in r and it is represented by the parent vector p. The condition of Line 2 can be precomputed for each node in the path between x and r with a simple traversal of the tree, so the linear time complexity of the algorithm is guaranteed.

Program 1. Generalized Dandelion Code

```
1: for v from x to r do
2:   if v = max{w ∈ path(v, r)} then
3:     swap p(x) and p(v), together swap l(x, p(x)) and l(v, p(v))
4: for v ∈ V(T) − {x, r} in increasing order do
5:   append (p(v), l(v, p(v))) to the code
```

The decoding algorithm proceeds to break cycles and loops in the digraph obtained by a given string, and to reconstruct the original path from x to r. We refer to [5] for details.

As an example consider the coding of tree shown in Figure 2(c) with $r = 0$ and $x = 1$, the only swap that occurs is between $p(1)$ and $p(8)$. The code string obtained is: $[(0, \varepsilon), (0, \varepsilon), (2, 1), (8, 3), (8, 2), (1, 3), (5, 3)]$.

As mentioned in the previous section, we now exploit the Generalized Dandelion code to show that $|\mathcal{Z}_k^n| = |\mathcal{R}_k^n|$. Each tree in $\mathcal{Z}_k^n$ has $n-k+1$ nodes and therefore is represented by a code string of length $n-k-1$. Each element of this string is either $(0, \varepsilon)$ or a pair in $[1, n-k] \times [1, k]$. Then there are exactly $k(n-k)+1$ possible pairs. This implies that there are $(k(n-k)+1)^{n-k-1}$ possible strings, thus proving $|\mathcal{Z}_k^n| = (k(n-k)+1)^{n-k-1} = |\mathcal{R}_k^n|$.

5 A Linear Time Algorithm for Coding k-Trees

In this section we present a new bijective code for k-trees and we show that this code permits linear time encoding and decoding algorithms. To the best of our knowledge, this is the first bijective encoding of k-trees with efficient

implementation. In [11] a bijective code for k-trees was presented, but it is very complex and does not seem to permit efficient implementation.

In our algorithm, initially, we have to root the k-tree T_k in a particular clique Q and perform a relabeling to obtain a Rényi k-tree R_k. Then, exploiting the characteristic tree $T(R_k)$ and the Generalized Dandelion code, we bijectively encode R_k. The most demanding step of this process is the computation of $T(R_k)$ starting from R_k and viceversa. This will be shown to require linear time.

Notice that the coding presented in [8], which deals with Rényi k-trees, is not suitable to be extended to obtain a non redundant code for general k-trees.

As noted at the end of the previous section, using the Generalized Dandelion Code, we are able to associate elements in $\mathcal{R}_k^n$ with strings in:

$$\mathcal{B}_k^n = (\{(0,\varepsilon)\} \cup ([1,n-k] \times [1,k]))^{n-k-1}$$

Since we want to encode all k-trees, rather than just Rényi k-trees, our final code will consist of a substring of the Generalized Dandelion Code for $T(R_k)$, together with information concerning the relabeling used to transform T_k into R_k.

Codes for k-trees associate elements in $\mathcal{T}_k^n$ with elements in:

$$\mathcal{A}_k^n = \binom{[1,n]}{k} \times (\{(0,\varepsilon)\} \cup ([1,n-k] \times [1,k]))^{n-k-2}$$

The obtained code is bijective: this will be proved by a decoding process that is able to associate to each code in $\mathcal{A}_{n,k}$ its corresponding k-tree. Note that $|\mathcal{A}_k^n| = |\mathcal{T}_k^n|$.

The encoding algorithm is summarized in the following 4 steps:

Coding Algorithm
Input: a k-tree T_k
Output: a code in $\mathcal{A}_{n,k}$

1. Identify Q, the k-clique adjacent to the maximum labeled leaf l_M of T_k. By a relabeling process ϕ, transform T_k into a Rényi k-tree R_k.
2. Generate the characteristic tree T for R_k.
3. Compute the generalized Dandelion Code for T using as parameters $r = 0$ and $x = \phi(\overline{q})$, where $\overline{q} = \min\{v \notin Q\}$. Remove from the obtained code string S the pair corresponding to $\phi(l_M)$.
4. Return the code (Q, S).

Assuming that the input k-tree is represented by adjacency lists adj, we detail how to implement the first three steps of our algorithm in linear time.

Step 1. Compute the degree $d(v)$ of each node v and find l_M, i.e. the maximum v such that $d(v) = k$, then the node set Q is $adj(l_M)$. In order to obtain a Rényi k-tree, nodes in Q have to be associated with values in $\{n-k+1, n-k+2, \ldots, n\}$. This relabeling can be described by a permutation ϕ defined by the following three rules:

1. if q_i is the i-th smallest node in Q, assign $\phi(q_i) = n - k + i$;
2. for each $q \notin Q \cup \{n-k+1, \ldots, n\}$, assign $\phi(q) = q$;
3. unassigned values are used to close permutation cycles, formally: for each $q \in \{n-k+1, \ldots, n\} - Q$, $\phi(q) = i$ such that $\phi^j(i) = q$ and j is maximized.

Figure 4 provides a graphical representation of the permutation ϕ corresponding to the 3-tree in Figure 1(a), where $Q = \{2, 3, 9\}$, obtained as the neighborhood of $l_M = 10$. Forward arrows correspond to values assigned by rule 1, small loops are those derived from rule 2, while backward arrows closing cycles are due to rule 3.

Fig. 4. Graphical representation of ϕ for 3-tree in Figure 1(a)

The Rényi k-tree R_k is T_k relabeled by ϕ. The final operation of this step is to order the adjacency lists of R_k. The reason for this will be clear in the next step.

Figure 2(a) gives the Rényi 3-tree R_3 obtained by relabeling the T_3 of Figure 1(a) by ϕ represented in Figure 4. The root of R_3 is $\{9, 10, 11\}$.

Let us now prove that the overall time complexity of step 1 is $O(nk)$. The computation of $d(v)$ for each node v can be implemented by scanning all adjacency lists of T_k. Since a k-tree with n nodes has $\binom{k}{2} + k(n-k)$ edges, it requires $O(nk)$ time, which is linear with respect to the input size.

The procedure to compute ϕ in $O(n)$ time is given in Program 2:

Program 2. Compute ϕ

```
1: for q_i ∈ Q in increasing order do
2:    φ(q_i) = n − k + i
3: for i = 1 to n − k do
4:    j = i
5:    while φ(j) is assigned do
6:       j = φ(j)
7:    φ(j) = i
```

Assignments of rule 1 are made by the loop in Line 1, in which it is assumed that elements in Q appear in increasing order. The loop in Line 3 implements rules 2 and 3 in linear time. Indeed the while loop condition of Line 5 is always false for all those values not belonging to $Q \cup \{n-k+1, \ldots, n\}$. For remaining values the inner while loop scans each permutation cycle only once, according to rule 3 of the definition of ϕ.

Relabeling all nodes of T_k to obtain R_k requires $O(nk)$ operations, as well as the procedure in Program 3 used to order its adjacency lists.

Program 3. Order Adjacency Lists

```
1: for i = 1 to n do
2:    for each j ∈ adj(i) do
3:       append i to newadj(j)
4: return newadj
```

Step 2. The goal of this step is to build the characteristic tree T of R_k. In order to guarantee linear time complexity we avoid the explicit construction of the skeleton $S(R_k)$. We build the node set and the edge set of T separately.

The node set is computed identifying all maximal cliques in R_k; this can be done by pruning R_k from k-leaves. The pruning process proceeds by scanning the adjacency lists in increasing order: whenever it finds a node v with degree k, a node in T labeled by v, representing the maximal clique with node set $v \cup adj(v)$, is created. Then v is removed from R_k and consequently the degree of each of its adjacent nodes is decreased by one.

In a real implementation of the pruning process, in order to limit time complexity, the explicit removal of each node should be avoided, keeping this information by marking removed nodes and decreasing node degrees. When v becomes a k-leaf, the node set identifying its maximal clique is given by v union the nodes in the adjacent list of v that have not yet been marked as removed. We will store this subset of the adjacency list of v as K_v, it is a list of exactly k integers.

Note that, when v is removed, at most one of its adjacent nodes becomes a k-leaf (see Remark 1). If this happens, the pruning process selects the minimum between the new k-leaf and the next k-leaf in the adjacency list scan.

At the end of this process the original Rényi k-tree is reduced to its root $R = \{n-k+1, \ldots, n\}$. To represent this k-clique the node labeled 0 is added to T (the algorithm also assigns $K_0 = R$).

This procedure is detailed in Program 4; its overall time complexity is $O(nk)$. Indeed, it removes $n-k$ nodes and each removal requires $O(k)$ time.

In order to build the edge set, let us consider for each node v the set of its eligible parents, i.e. all w in K_v. Since all eligible parents must occur in the ascending path from v to the root 0 (see Remark 2), the correct parent is the one at maximum distance from the root. This is the reason why we proceed following the reversed pruning order.

The edge set is represented by a vector p identifying the parent of each node. 0 is the parent of all those nodes s.t. $K_v = R$. The level of these nodes is 1.

To keep track of the pruning order, nodes can be pushed into a stack during the pruning process. Now, following the reversed pruning order, as soon as a node v is popped from the stack, it is attached to the node in K_v at maximum level. We assume that the level of nodes in R (which do not belong to T) is 0.

The pseudo-code of this part of Step 2 is shown in Program 5.

The algorithm of Program 5 requires $O(nk)$ time. In fact, it assigns the parent of $n-k$ nodes, each assignment involves the computation of the maximum (Line 6) and requires k comparisons.

Program 4. Prune R_k

function remove(x)

```
1: let K_x be adj(x) without all marked elements
2: create a new node in T with label x        // it corresponds to node
   {x} ∪ K_x of the skeleton
3: mark x as removed
4: for each unmarked y ∈ adj(x) do
5:    d(y) = d(y) − 1
```

main

```
1: for v = 1 to n − k do
2:    w = v
3:    if d(w) = k then
4:       remove(w)
5:       while ∃ an unmarked u ∈ adj(w) s.t u < v and d(u) = k do
6:          w = u
7:          remove(w)
```

Program 5. Add edges

```
1: for each v ∈ [1, n − k] in reversed pruning order do
2:    if K_v = R then
3:       p(v) = 0
4:       level(v) = 1
5:    else
6:       choose w ∈ K_v s.t. level(w) is maximum
7:       p(v) = w
8:       level(v) = level(w) + 1
```

To complete step 2 it only remains to label each edge $(v, p(v))$. When $p(v) = 0$, the label is ε; in general, the label $l(v, p(v))$ must receive the index of the only element in $K_{p(v)}$ that does not belong to K_v. This information can be computed in $O(nk)$ by simple scans of lists K_v. The ordering of the whole adjacency list made at the end of step 1 ensures that elements in all K_v appear in increasing order.

Figure 2(c) shows the characteristic tree computed for the Rényi 3-tree of Figure 2(a).

Step 3. Applying the generalized Dandelion Code with parameters 0 and $x = \phi(\overline{q})$, where $\overline{q} = \min\{v \notin Q\}$, we obtain a code S consisting in a list of $n-k-1$ pairs. For each $v \in \{1, 2, \ldots, n-k\} - \{x\}$ there is a pair $(p(v), l(v, p(v)))$ taken from the set $\{(0, \varepsilon)\} \cup ([1, n-k] \times [1, k])$. Given all this, the obtained code is redundant because we already know, from the relabeling process performed in Step 1, that the greatest leaf l_M of T_k corresponds to a child of the root in T. Therefore the pair associated to $\phi(l_M)$ must be $(0, \varepsilon)$ and can be omitted.

The Generalized Dandelion code already omits the information $(0, \varepsilon)$ associated with the node x, so, in order to reduce the code length, we need to guarantee that $\phi(l_M) \neq x$. We already know that a k-tree on $n \geq k+2$ nodes has at least 2 k-leaves. As Q is chosen as the adjacent k-clique of the maximum leaf l_M it cannot contain a k-leaf. So there exists at least a k-leaf less than l_M that does not belong to Q; $\overline{q}$ will be less or equal to this k-leaf. Consequently $\overline{q} \neq l_M$ and, since ϕ is a permutation, $\phi(l_M) \neq \phi(\overline{q})$. The removal of the redundant pair from the code S completes Step 3.

Since the Generalized Dandelion Code can be computed in linear time, the overall time complexity of the coding algorithm is $O(nk)$.

We want to remark that we choose Dandelion Code because it allows us to easily identify an information (the pair $(0, \varepsilon)$ associated to $\phi(l_M)$) that can be removed in order to reduce the code length from $n-k-1$ to $n-k-2$: this is crucial to obtain a bijective code for all k-trees.

Many other known codes for Cayley's trees can be generalized to encode edge labeled trees, obtaining bijection between Rényi k-trees and strings in $\mathcal{B}_{n,k}$. However other known codes, such as all Prüfer-like codes, do not make it possible to identify a removable redundant pair. This means that not any code for Rényi k-trees can be exploited to obtain a code for k-trees.

The returned pair (Q, S) belongs to $\mathcal{A}_{n,k}$, since $Q \in \binom{[1,n]}{k}$, and S is a string of pairs obtained by removing a pair from a string in $\mathcal{B}_{n,k}$. Due to lack of space we cannot discuss here how this pair can be efficiently represented in $\lceil \log_2(|\mathcal{A}_{n,k}|) \rceil$ bits.

The Dandelion Code obtained from the characteristic tree in Figure 2(c) with parameters $r = 0$ and $x = 1$ is: $[(0, \varepsilon), (0, \varepsilon), (2, 1), (8, 3), (8, 2), (1, 3), (5, 3)] \in \mathcal{B}_3^{11}$; this is a code for the Rényi 3-tree in Figure 2(a). The 3-tree T_3 in Figure 1(a) is coded as: $(\{2, 3, 9\}, [(0, \varepsilon), (2, 1), (8, 3), (8, 2), (1, 3), (5, 3)]) \in \mathcal{A}_3^{11}$. We recall that in this example $Q = \{2, 3, 9\}$, $l_M = 10$, $\overline{q} = 1$, $\phi(l_M) = 3$, and $\phi(\overline{q}) = 1$.

6 A Linear Time Algorithm for Decoding k-Trees

Any pair $(Q, S) \in \mathcal{A}_{n,k}$ can be decoded to obtain a k-tree whose code is (Q, S). This process can be performed with the following algorithm:

Decoding Algorithm
Input: a code (Q, S) in $\mathcal{A}_{n,k}$
Output: a k-tree T_k

1. Compute ϕ starting from Q and find l_M and $\overline{q}$.
2. Insert the pair $(0, \varepsilon)$ corresponding to $\phi(l_M)$ in S and decode it to obtain T.
3. Rebuild the Rényi k-tree R_k by visiting T.
4. Apply ϕ^{-1} to R_k to obtain T_k.

Let us detail the decoding algorithm. Once Q is known, it is possible to compute $\overline{q} = \min\{v \in [1, n] | v \notin Q\}$ and ϕ as described in Step 1 of coding algorithm. Since all internal nodes of T explicitly appear in S, it is easy to derive set L of all leaves of T by a simple scan of S. Note that leaves in T coincide with k-leaves in

R_k. Applying ϕ^{-1} to all elements in L we can reconstruct the set of all k-leaves of the original T_k, and therefore find l_M, the maximum leaf in T_k.

In order to decode S, a pair $(0, \varepsilon)$ corresponding to $\phi(l_M)$ needs to be added, and then the decoding phase of the Generalized Dandelion Code with parameters $\phi(\overline{q})$ and 0 applied. The obtained tree T is represented by its parent vector.

Program 6. Rebuild R_k

```
1: initialize R_k as the k-clique R on {n-k+1, n-k+2, ..., n}
2: for each v in T in breadth first order do
3:   if p(v) = 0 then
4:     K_v = R in increasing order
5:   else
6:     let w be the element of index l(v, p(v)) in K_{p(v)}
7:     K_v = K_{p(v)} - {w} ∪ {p(v)} in increasing order
8:   add v to R_k
9:   add to R_k all edges (u, v) s.t. u ∈ K_v
```

The reconstruction of the Rényi k-tree R_k is detailed in Program 6. Finally, T_k is obtained by applying ϕ^{-1} to R_k.

The overall complexity of the decoding algorithm is $O(nk)$. In fact the only step of the algorithm that requires some explanation is Line 7 of Program 6. Assuming that $K_{p(v)}$ is ordered, to create K_v in increasing order, $K_p(v)$ simply needs to be scanned omitting w and inserting $p(v)$ in the correct position. As $K_0 = \{n-k+1, \ldots, n\}$ is trivially ordered, all K_v will be ordered.

7 Conclusions

In this paper we have introduced a new bijective code for labeled k-trees which, to the best of our knowledge, produces the first encoding and decoding algorithms running in linear time with respect to the size of the k-tree.

In order to develop our bijective code for k-trees we passed through a transformation of a k-tree in a Rényi k-tree and developed a new coding for Rényi k-trees based on a generalization of the Dandelion code. The choose of Dandelion code is motivated by the need of identifying and discarding a redundant information. This is crucial to ensure the resulting code for k-trees to be bijective.

All details needed to obtain linear time implementations for encoding and decoding algorithms have been presented.

References

1. Beineke, L.W., Pippert, R.E.: On the Number of k-Dimensional Trees. Journal of Combinatorial Theory 6, 200–205 (1969)
2. Bodlaender, H.L.: A Tourist Guide Through Treewidth. Acta Cybernetica 11, 1–21 (1993)

3. Bodlaender, H.L.: A Partial k-Arboretum of Graphs with Bounded Treewidth. Theoretical Computer Science 209, 1–45 (1998)
4. Caminiti, S., Finocchi, I., Petreschi, R.: A Unified Approach to Coding Labeled Trees. In: Farach-Colton, M. (ed.) LATIN 2004. LNCS, vol. 2976, pp. 339–348. Springer, Heidelberg (2004)
5. Caminiti, S., Petreschi, R.: String Coding of Trees with Locality and Heritability. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 251–262. Springer, Heidelberg (2005)
6. Cayley, A.: A Theorem on Trees. Quarterly Journal of Mathematics 23, 376–378 (1889)
7. Chen, W.Y.C.: A general bijective algorithm for trees. In: Proceedings of the National Academy of Science, vol. 87, pp. 9635–9639 (1990)
8. Chen, W.Y.C.: A Coding Algorithm for Rényi Trees. Journal of Combinatorial Theory 63A, 11–25 (1993)
9. Deo, N., Micikevičius, P.: A New Encoding for Labeled Trees Employing a Stack and a Queue. Bulletin of the Institute of Combinatorics and its Applications (ICA) 34, 77–85 (2002)
10. Eğecioğlu, Ö., Remmel, J.B.: Bijections for Cayley Trees, Spanning Trees, and Their q-Analogues. Journal of Combinatorial Theory 42A(1), 15–30 (1986)
11. Eğecioğlu, Ö., Shen, L.P.: A Bijective Proof for the Number of Labeled q-Trees. Ars Combinatoria 25B, 3–30 (1988)
12. Foata, D.: Enumerating k-Trees. Discrete Mathematics 1(2), 181–186 (1971)
13. Greene, C., Iba, G.A.: Cayley's Formula for Multidimensional Trees. Discrete Mathematics 13, 1–11 (1975)
14. Harary, F., Palmer, E.M.: On Acyclic Simplicial Complexes. Mathematika 15, 115–122 (1968)
15. Markenzon, L., Costa Pereira, P.R., Vernet, O.: The Reduced Prüfer Code for Rooted Labelled k-Trees. In: Proceedings of 7th International Colloquium on Graph Theory. Electronic Notes in Discrete Mathematics, vol. 22, pp. 135–139 (2005)
16. Moon, J.W.: The Number of Labeled $k-$Trees. Journal of Combinatorial Theory 6, 196–199 (1969)
17. Moon, J.W.: Counting Labeled Trees. William Clowes and Sons, London (1970)
18. Neville, E.H.: The Codifying of Tree-Structure. In: Proceedings of Cambridge Philosophical Society, vol. 49, pp. 381–385 (1953)
19. Picciotto, S.: How to Encode a Tree. PhD thesis, University of California, San Diego (1999)
20. Prüfer, H.: Neuer Beweis eines Satzes über Permutationen. Archiv der Mathematik und Physik 27, 142–144 (1918)
21. Rényi, A., Rényi, C.: The Prüfer Code for k-Trees. In: Erdös, P., et al. (eds.) Combinatorial Theory and its Applications, pp. 945–971. North-Holland, Amsterdam (1970)
22. Rose, D.J.: On Simple Characterizations of k-Trees. Discrete Mathematics 7, 317–322 (1974)

Market-Based Service Selection Framework in Grid Computing

Yanxiang He and Haowen Liu

School of Computer, State Key Lab of Software engineering,
Wuhan University, Wuhan, Hubei 430072
yxhe@whu.edu.cn, jimlhw@126.com

Abstract. Much research focuses on the foundational components of computational grid infrastructure, making the grid service economic possible. The service consumers could submit their jobs to the published services but don't need to concern how the jobs are mapped to the resources, how the resources are organized and managed, and how the revenue is divided among the organization. However, most of the services are contributed by the providers voluntarily; it means the services could be published optionally or canceled at any time even when they are carrying out some jobs, and the consumers take much risk in selecting the providers. There are few mechanisms to encourage the service providers to compete for profit and satisfy the consumers, especially considering the consumers' preference of deadline and price. We develop a market-based framework to attract the providers, and decrease the risk of the consumers, in which different bidding strategies of providers and different behavior models of consumers are discussed.

1 Introduction

Grid infrastructure is a large VO (Virtual Organization) which integrates a large mount of distributed heterogeneous resources and high performance computing capabilities into a super service plat, which can provide huge computing capability, storage capability and so on [1, 2]. And with the development of the Grid Technology, which is trended to incorporate web services[3], it is more easy to provide dependable, consistent, pervasive, and inexpensive access to high-end computing services, storage services, retrieving services, and so on.

GIIS component in Globus Toolkit [4] provides a service for service registration and aggregation, but it couldn't prevent some service providers which had been assigned some jobs leaving at any time; it also provides a service for service discovering, but couldn't assure the service consumers would be satisfactory with the service according to their real demand and preference [16]. The former is due to the heterogeneous feature of the grid resources and the dynamic feature of services; it is difficult to form a steady service market so that the users couldn't ease themselves to search and select on-demand. The latter is referring to the service providers and consumers themselves. Both the service providers and consumers have their own goals, objectives, strategies and supply-and-demand patterns, So even the service is free of charge, the consumers

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 421–434, 2007.

are afraid of whether it would complete their jobs on time, or even the service costs the consumers much more, but the consumers would prefer to select it, the examples can be found in our another paper [16].

In order to make good use of the resources and satisfy the real demand of the consumers, many other mechanisms have been brought forward, such as the scheduling mechanism of considering deadline and cost in Nimrod/G [5, 6] and GRACE infrastructure for the Nimrod/G resource broker [7, 9], and service level agreement based scheduling mechanism in the Grid [13, 14].

As mentioned in [5, 6], the Nimrod/G resource broker, a global resource management and scheduling system for computational grid, built with Globus services. They tried to make the resources economization, but only support static price, tried to support deadline based scheduling and dynamic resource trading using Globus and GRACE services, but also ignore that the low-price services will still be the bottleneck of scheduling. In [7, 9], they introduced dynamic price mechanism, which is just the diversity in different time slots, the example is that the access price on weekend is rush low. However, it would come with two problems. The first is that the rational and intelligent consumers would prefer to submit their short jobs on weekend which finally affected the performance of the service, and the second is that it is difficult to account the fee when the long job is crossed over different time slots. The scheduling mechanism based on service level agreement in [13, 14] aims to meet the job's requirement on performance, cost, security, or other quality of service metrics, and overcomes the difficulties during the mappings that requires negotiation among application and resources, but still couldn't promote the competing among the providers.

The best way to adjust the price is to consider the market mechanism. E-Bay Style market mechanisms such as Yahoo! Auction, Amazon, run well in the e-Commerce [10, 11], and provide great security technique to assure the success of the transaction. However, it is not fit for the grid service. The goods in the transaction is the real entity not the virtual encapsulated service, which could be published or canceled at any time. And the seller's selection in the platform is partly based on the transaction-based feedback (rating and comment) by the buyers, which is subjective and maybe not the true.

In order to resolve above problems, we advance a market-based service selection framework, which encourages the service providers (Provider) to live more time to fulfill assigned jobs and increase their resource utilization, to compete for more profit, to form a steady market, decreasing the risk of the service consumers (Consumer), and satisfying their demand more possibly.

This article firstly describes the whole framework in Section 2. Section 3 discusses the strategies of the service providers, the behaviors of the service consumers. The experiments and analysis are following in section 4. Section 5 summaries our works and gives a future plan.

2 Framework

In this section, we introduce the framework of the service market for matching consumers with relevant providers in the case of grid environments. The architecture is seen in Figure 1, and the basic roles, functions and issues are described below.

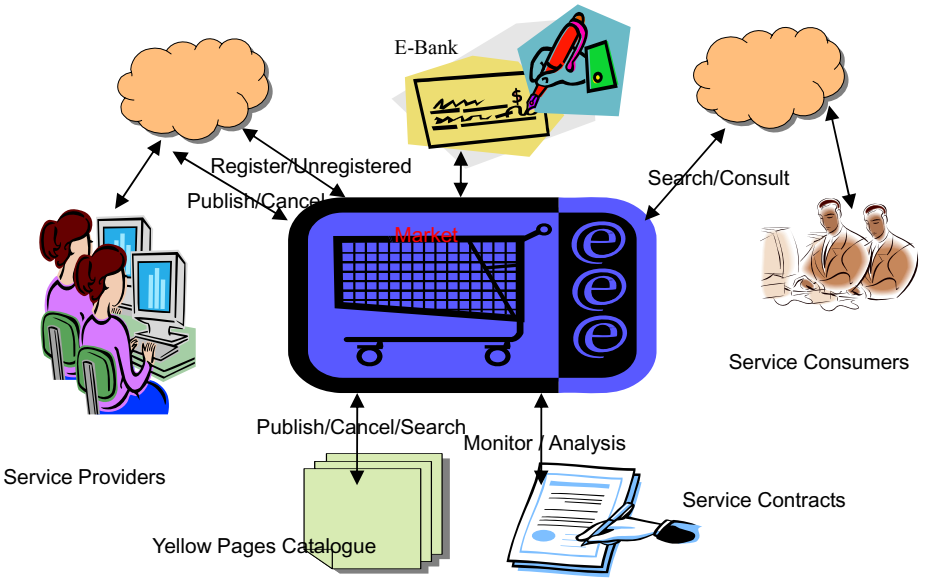


Fig. 1. Architecture of a Service Market with service providers and consumers

2.1 General Design

We describe the general process of the mechanism in the service market, and the process in an auction could be divided into four stages. In the first stage, the providers take the actions of registration, unregistration, publishing services and canceling services, to form a service market. In the second stage, the consumers submit their jobs and specify a desired service price and deadline, and even the preference to price or deadline, which are passed on to market manager, calling for bids. In the third stage, the providers subsequently compete against each other in an auction by placing bid to obtain a contract. In the fourth stage, each consumer is shown a list of providers, who could satisfy his demands, so that he could select one according to his preference right now or even after consulting about each provider further from the market manager. Then the contract is formed, the jobs are carried out according to the contracts, and the winning providers will get paid for their services and also get sanctioned if they don't deliver what they promise. The consumer or provider may take another action before the job is completed, which satisfies the real market. So during a fix period, many auctions happen to form a round of auction. And during the process, we presuppose that the market manager has no any collusion with any user, promise the market is fair.

2.2 Service Providers and Consumers

Software agents are used to facilitate the fine grain of interaction, bidding, and selection in the service market. A software agent is a program with an owner, whom it represents. The software agent can autonomously perform tasks on behalf of its owner in order to satisfy its goals. To this point, it also has the ability to interact with other

agents or external sources. For our mechanisms, we have software agents for the providers and consumers and for the enabling intermediary the market manager.

A provider may be a person or a community, who can supply different services aiming of different applications. As we know different services are incorporated from different resources representing different capability, just like that a "computing" service would require a large amount of processor resources, which is different from a "storage" service which would require a large amount of disk space. And even the same service has different application environments. Taking two "weather-forecast" services for example, they may aim different area, one serves China, and the other serves the U.S.A. So when publishing the services, he must describe the service type, service environment clearly. And before publishing any service, a provider should register in the market with enough credit.

The main task of a provider agent is to bid on arriving job according to his internal resource state and his bidding strategy, which is detailed in Section 3.2.

Also a service consumer has his own aim, strategy, behavior, and demand. We mainly consider the preference of the consumers about the price and deadline, and the optional constraint about the reputation of the provider, which is detailed in Section 3.3.

2.3 Service Market Manager

2.3.1 Providing Yellow Page Service

For the direct and convenient access from the consumers, the service market manager must get the services together as a market, which is also easy to manage and maintain. In order to form a steady and trusted environment, the market has a high doorstep for the provider's entering and a constraint for the provider's leaving.

On receiving the request for registration from a provider, the manager should check whether he has minimum credit to pay for possible violation of the contract. The credit reflects the competence of the provider in assuring to fulfill assigned jobs, it means the provider was able to burden a contract when his credit is bigger than the possible compensate of violating a contract. If the provider is allowed, the manager would apply a credit account for him in the e-Bank.

On receiving the request for unregistration from a provider, the manager should check whether he has fulfilled all his contracts, and apply to freeze his account before compelling to recoup related compensation to related consumers, which assures the interest of the consumers.

On receiving the request for publishing the service from a provider, the manager should check whether the service description is clear and unabridged, and then publish it in the Yellow Pages catalogue. The reaction after receiving request of canceling is similar to unregistration.

On receiving the request for searching for services from a consumer, the manager also check his credit, and then help search in the Yellow Pages catalogue.

2.3.2 Supervising the Service Contracts

In addition to the functions of service registration, unregistration, publishing, canceling and searching, the manager acts as a mediator. Once the provider and

consumer had agreed on a contract, the copy of the contract would reserved by the manager, which assure capital transfer between the provider and the consumer. By monitoring and checking the contracts in process, the mediator compels the provider to pay for corresponding compensation to the consumer by automating transferring the capital from the e-Bank, once the contract is unfulfilled on time. Also the payment from consumers to provider is transferred through the e-Bank. In this paper, the communication protocol between the consumer and the provider is based on the Contract Net Protocol [12], and the process of generating a new contract is simplified, but not like the SLAs [13, 14], which is beyond our consideration.

2.3.3 Providing Consultation Service for the Consumers

In order to attract the providers and the consumers to the market, the manager provides "free" service of registering, publishing and searching services, and etc. The state would be changed, especially when there exists more than one market competing for providing the "market" service. However, it is possible for the manager to obtain additional profit for providing useful information, besides the profit from the successful transaction. Owing to the management and supervision of the service contracts, the manager knows whether a contract is fulfilled, what the capital of the contract is, who is the provider and consumer of the contract. In the market, the manager should not reveal the detail information of any contract; however, he could statistic and analysis the knowledge about a provider's reputation information, such as the ability whether he could burden a new contract, the fulfillment rate of his contracts, and the transaction capital of his contracts. The manager thinks that the providers with higher reputation take better state of capital and fulfillment rate of contracts. So the manager could act as a consultee and obtain additional profit for providing the consultation service.

3 Auction

3.1 Novel Auction

In auction, usually the owners of the goods are sellers, the users of the goods are buyers, and the sellers cash in by calling for bids on their goods, seldom the case that the sellers cash in by bidding to sell goods. However, it is still an auction, a reverse auction, which usually appears in the situation that the buyers lead the market and call for bids. In grid service market, the goods is grid service, which is special un-storable merchandise, and requires to be made good use of. When great grid services compete for consume, the providers have to react to the calling for bids.

The actual choice of the auction process can depend on many factors. In this paper, we focus on the single-bid sealed auction, being a communication-efficient auction

[15]. But there are two differences from the general first-price, owing to the service feature in grid and the demands of the consumers.

To a provider, the price is not the unique competence of his service, but one of the most important parameters. We mainly concern the parameters including capability, access price and execution time. These parameters are changing and affecting each other. For example, when the resources are enough and the capability is idle, the provider could contribute more resources to a job, so that he would bid with less time and higher price, when the resources are in tense, the provider had to bid with lower price and more time to attract the consumers.

To a consumer, the less cost and the less time, the more satisfied. Generally speaking, the less cost, the better when he prefers to the price, and the less time, the better when he prefers to the deadline.

So the first difference is that the provider would bid not only on price, but also on time. The second is to bid as low price and less time as possibly to win the contract, neither the lowest price or the least time, nor the lowest price and the least time simultaneous. Whether he could win greatly depend on the consumer's preference, which is preliminarily introduced in another paper [16].

In more detail, the market operates in the following manner. Each time the consumer submits a job, an auction is activated. In each round of auction, the auctioneer agent retrieves the initial services which satisfy the conditions of service type and service environment. Then the auctioneer calls for bids to the related service owners. After a fixed time, the auctioneer agent ranks all the bids it received according to the consumer's preference. The ranked services are showed to the consumer for further selection. Finally the owner of the selected service would take a contract with the consumer, and then carry out the job before deadline under the monitor. Once the contract is fulfilled, the resources occupied are released for competition in the next auction.

3.2 Bidding Strategies of Provider

A rational bidder seeks to maximize his revenue with minimum resources, and the outcome of his bid is unselected or selected. It is because the preference of the consumers is difficult to capture that the provider couldn't decide to adjust price or time blindly. What he could do is to adjust his access price and execution time mainly according to the capability of his resources and the state of his history bids, as the latest bid could reflect the status of the market furthest. If the remainder resources are enough which are compared to last bid, he could bid less execution time to complete the jobs with increasing the investment of the available resources. If the remainders are not enough, he could only bid lower price to increase his competence in the market. And once his bid is selected, he would increase the bid price in the next auction, aiming of more revenue. Generally speaking, the provider prefers the price to time.

Therefore, the provider will adjust his bid time and price as table 1.

Table 1. Adjustment Strategy of bid time and bid price

Result of Last Bid	State of Resource	Adjustment of bid time	Adjustment of bid price
Not selected	Enough	$-\Delta t(\Delta t > 0)$	$-\Delta p\ (\Delta p \geq 0)$
	Not enough	$+\Delta t(\Delta t \geq 0)$	$-\Delta p\ (\Delta p > 0)$
Selected	Enough	$-\Delta t(\Delta t \geq 0)$	$+\Delta p\ (\Delta p > 0)$
	Not enough	$+\Delta t(\Delta t \geq 0)$	$+\Delta p\ (\Delta p \geq 0)$

3.3 Behavior Models and Selection Strategies of Consumer

When a consumer is showed a list of providers, he might choose one or not. No matter which one he would prefer, his job would be fulfilled on time or else he should get the corresponding compensation within the contract. However, the question is whether the provider has the ability to pay for compensation, so he could further choose to buy some reputation information about the providers to decrease the risk.

We model four classes of consumer behaviors according to his preference.

(a) Select the first provider in the list, without further consultation about the provider. If he prefers the deadline, he would choose the least time no matter the price (NMP), which means he would not concern how high the price is, and have a risk of unknowing whether the provider could burden the new contract. If he prefers the price, he would choose the lowest price but no matter the time (NMT).

(b) Select the first accepted provider in the list, without further consultation about the provider. If he prefers the deadline, he could bear the price beyond his expectation, but not too high, which means he could choose the less time, but accepted price (LTAP). If he prefers the time, he would choose the lower price, but accepted time (LPAT) similarly.

(c) Select the first satisfying provider in the list, without further consultation. If he prefers the deadline, he couldn't bear the price is beyond the expectation at the same time, so he would choose the less time and satisfying price (LTSP). The reverse is lower price and satisfying time (LPST).

(d) Select the first disburdening behavior (FD). In this model, the consumer would consult some aspect about the provider in sequential order from top to bottom, until he finds the provider, whose concerned aspect is disburdening. If he prefers the deadline, which reflects his expectation that his job is completed on time and should be completed, he could concern whether the provider has broken any contract. If the prefers the price, which reflects his expectation that the job cost less, or the compensation is satisfying, he could concern whether the provider has the ability to pay the fiddler of violating the new contract.

As mentioned above, the reputation information about a provider includes the ability whether he could burden a new contract, the fulfillment rate of his contracts, and the transaction capital of his contracts, which information to consult depends on the consumer's real demand.

As a rational consumer, taking the first behavior is a little risky, and taking the fourth behavior costs more but with less risk. So usually he would take the second or the third behavior, or combine the forth behavior with the first three behaviors.

4 Experiments and Analysis

4.1 Experiments Design

In order to understand the mechanism of the service market and its dynamics, we just simulated the computational service and perform simulations of the auction as above. In the experiments, the consumers submit multi-jobs at the same time, which could be simulated as a consumer submits a standard job in fixed time interval. Suppose the speed of a standard processor is 10 MIPS (Million Instructions per Second), and a standard job costs one standard processor 100 seconds to complete, then the length of a standard job is 1000 MI, which is similar with Rajkumar Buyya [8]. With more then one standard processor, the execution time will be shortened, which is simply considered to be inverse with the number of processors, but the more processors, the more time of the communication and data delay between the processors, so we process the relation between time and number of processors as the below formula (1), which might be not exact, but is used by all the bidders, so that it would have little affection on the result of auction.

$$t = 100/m + 0.01*(100 - 100/m) \tag{1}$$

Where, m represents the number of the standard processors participating the job.

For each experiment, we create 20 test services offered by different providers, with the same service type and environment, different capability (represented by the number of processors) ranged between 30 and 600, different initial execution time randomly around 10 seconds, different credit in E-Bank ranged between 1000 and 20000 units, different initial access price randomly around 10 units/MI, and the lowest price with 1 unit/MI which represents the cost of consumed resources. As a rational provider, he would consider to make full use of his resource predominance firstly, after determining how many processors is proper, he could compute the possible bid time with formula (1), when he found the priority of his resources is not obvious, he would consider bidding with lower price in the next bid then, which reflect on the table 1, and the scope of change is no more than 1. However, when he found the possible bid price is under the lowest price, he could reject to bid. And he would not invest all his resources in one job, but reserve part to bid another job at the same time, once the job is completed, the released resources could increase his competence again.

We set the compensation coefficient and consultation coefficient as 1.5 and 0.1 respectively, the detailed compensation of violating a contract and cost of consultation is based on the real contract.

4.2 Results and Analysis

4.2.1 Stability

To evaluate the stability of the market with respect to different consumer behavior models, we now consider the change of the market time (average bid time) and the market price (average bid price), which is showed in Figure 2a, 2b and Figure 3a, 3b about the change of market time and price in 1000 round of auctions with the consumer' preference of deadline and price respective.

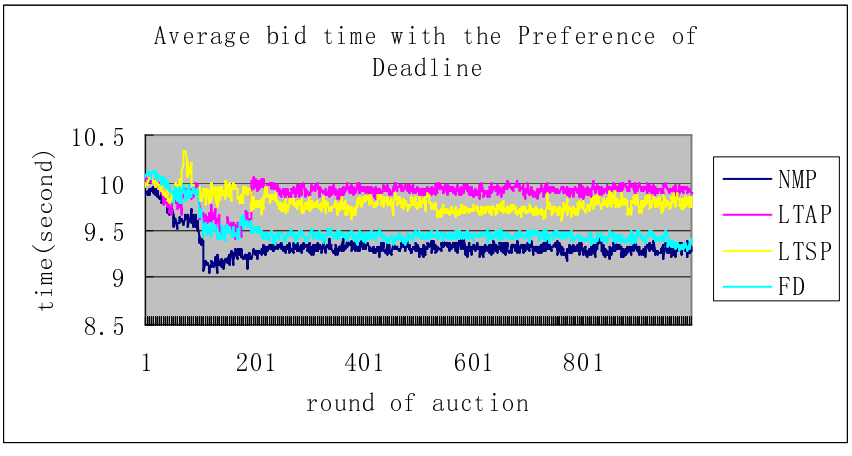


Fig. 2a. Average bid time with the Preference of Deadline

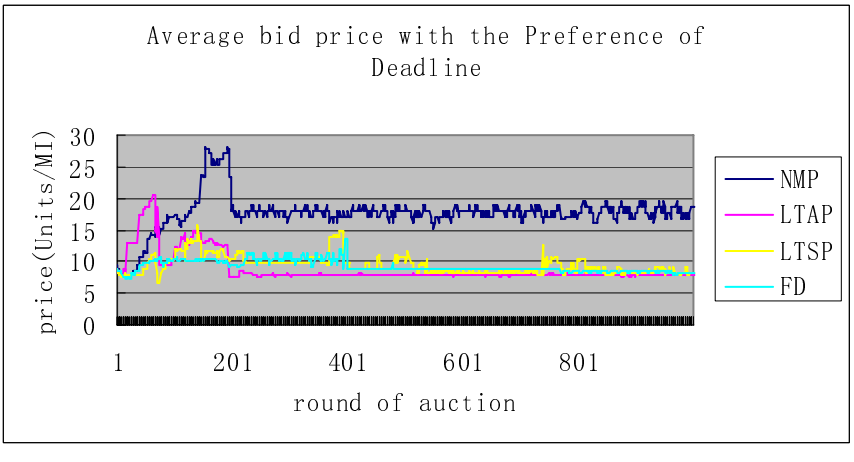


Fig. 2b. Average bid price with the Preference of Deadline

We can see from Figure 2a, 2b that the market time trends to be under 10 seconds and reaches stable under the four behavior models of consumer, however, the NMP results in least market time, the FD results a little more that NMP, and the ordinal is

LTSP and LTAP. However, the change of the market price is absolutely reverse to the change of the market time under the four behavior models. It is because that with the preference of Deadline, the list of received candidate bids is sorted according to time, and the bid in the top of the list would have more chance to win the contract. For the NMP, the consumer chooses the first of the list, which has the least time and possible high price. For the FD, the consumer chooses the one which hadn't violate before, giving the same chance to the provider of new entering and the one with high reputation. Both LTSP and LTAP have constraint with the price, giving the chance to the bit not in the front of the list, which results higher market time than NMP and FD, and lower market price than NMP.

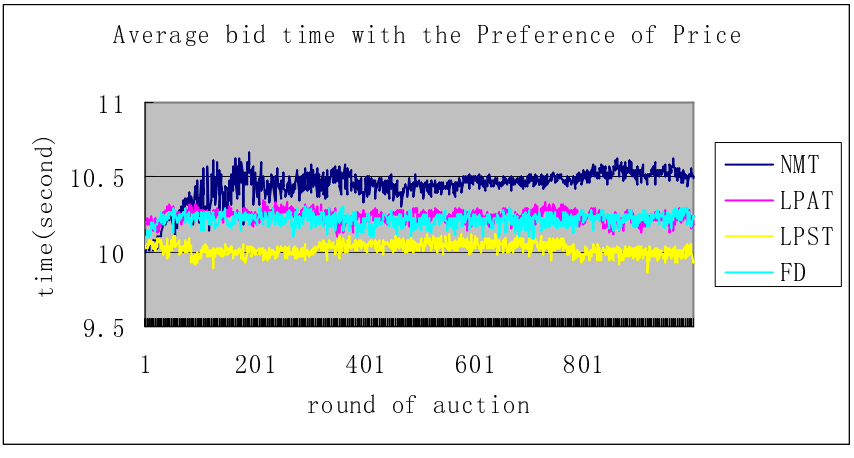


Fig. 3a. Average bid time with the Preference of Price

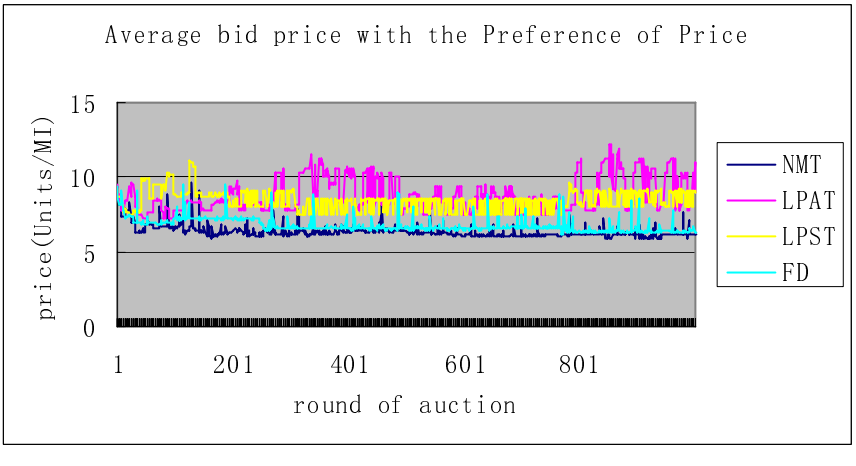


Fig. 3b. Average bid price with the Preference of Price

We can also see from Figure 3a, 3b that the market price trends to be under below 10 units / MI and reaches stable with the consumer' preference of price, and the result and reason are similar to the above.

4.2.2 Balance

At the same time, we can see the job distributions among the providers with different capability. Figure 4 and 5 show the results under different preferences and behaviors of consumers respective.

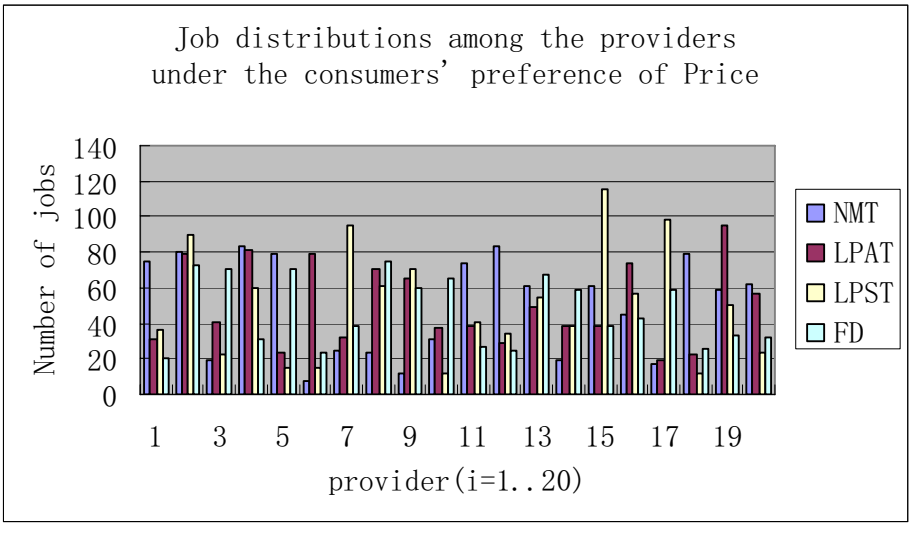


Fig. 4. Job distributions among different providers with different capability

For the convenient observation, we assign the increasing capability (number of processors), which is equally ranged between 30 and 600, to provider1, provider2... to provider20 sequent. We can know from Figure 4 that the jobs are randomly distributed to each provider approximately, which is affected little by the behaviors of the consumers. It is because that the consumers pay more attention to the price, the provider's capability has no obvious predominance in the market, so each provider has the same chance to catch a contract, and the disparity of jobs among the providers is not great.

We can know from Figure 5 that when the consumer takes the behavior of NMP or FD, most of the jobs are distributed to the providers with more capability, and the providers with less capability are assigned few jobs. However, when the consumer takes the behavior of LTAP or LTSP, the assigned jobs to the providers with less capability begin to increase, which reflects that the chance for the providers increases. It is because that the consumers focus on the deadline now, and the providers with more capability have obvious competence to contribute more resources on some job, so to take the position of monopoly. When the market capability is beyond the need of the

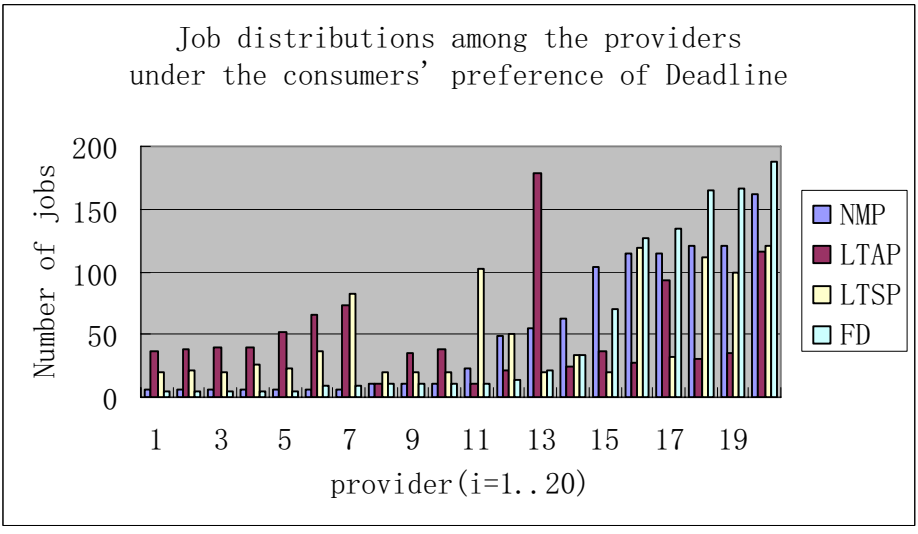


Fig. 5. Job distributions among different providers with different capability

consumers, the monopoly is not obvious, but we could conclude that once the whole market capability is in tension, the few providers would monopolize the market especially when the consumers take the behavior of NMP or FD. It is the diversity and dynamic of the consumer' preferences and behaviors, that the monopoly is not easy to be formed, which will be further researched and approved in the future.

4.2.3 Risk

The main threat to the market comes from three kinds of providers, they are the ones of new entering, the ones have ever exited from the market or violated the contracts, and the collusive ones.

For the new ones, they have the ability of burdening a new contract in the first few rounds, and have the interest to complete the contract, or else he would be punished with low reputation which would affect his future contract with the consumers taking the behavior of FD.

For the one who had ever violated the contracts, it has affection on the consumers who take other behaviors than the behavior of FD, which could be easily avoided by combining the FD to user's behavior of NMT, LPAT, LPST, NMP, LTAP or LTSP, by costing just a little more for consultation.

For the collusive ones, they usually sacrifice some of them to accumulate much credit and contracts to special ones, in order to attract more consumers. However, the collusion itself has no affection on the consumers, but affects the interest of the other providers, which could be identified by the market manager soon, and new incentive mechanisms should be introduced to resolve this problem, and will be in our future papers.

5 Conclusions and Future Works

In this paper, we have outlined a framework of service selection in the Grid that uses market mechanisms to attract the rational providers to contribute his resources for revenue, so as to form a stable and fair market, in which the consumers could select the satisfying service on-demand and the providers with different capability could survive. Four consumer behavior models have been talked in the framework, the first three models make the bids satisfy the consumers' demand as can as possibly, and the last model could decrease the consumers' risk in selecting provider. Another consumer behavior model could be easily added to the framework, and the model of FD could be combined with the other behavior models as a new model. Though only the computational service is talked about, the framework could be extended to support the other services, and the more complex case is that many markets compete for providers and consumers.

However, the framework brings any other problems. The first is the case of collusion, which should be further researched. The second is the job distribution among the providers when the whole market capability is under the demand of the consumers which is not considered in this paper.

References

1. Foster, I., Kesselman, C., Tueck, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. Int. J. Supercomputer Applications (2001)
2. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: Grid services for distributed system integration. IEEE Computer 35(6), 37–46 (2002)
3. Sarmenta, L., Chua, S., Echevarria, P., Mendoza, J.M.: Bayanihan computing.NET: Grid computing with XML Web services. In: CCGRID 2002. Proc. of the 2nd IEEE/ACM Int'l Conf. on Cluster Computing and the Grid, pp. 404–405. IEEE Computer Society Press, Berlin (2002)
4. Foster, I., Kesselman, C.: "Globus: A Metacomputing Infrastructure Toolkit. Intl J. Supercomputer Applications 11(2), 115–128 (1997)
5. Abramson, D., Giddy, J., Kotler, L.: High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Gird? In: IPDPS'2000, Mexico, IEEE Computer Society Press, Los Alamitos (2000)
6. Buyya, R., Abramson, D., Giddy, J.: Nimrod/G: An Architecture for a Resource Management and scheduling System in a Global Computational Grid. In: HPC ASIA'2000, China, IEEE Computer Society Press, Los Alamitos (2000)
7. Buyya, R., Abramson, D., Giddy, J.: An Economy Driven Resource Management Architecture for Globus Computational Power Grids. In: PTPTA'2000, Las Vegas (2000)
8. Buyya, R., Giddy, J., Abramson, D.: An evaluation of Economy-based Resource Trading and scheduling on Computing Power Grids for Parameter Sweep Applications. In: The second Workshop on Active Middleware Services AMS2000, conjunction with the HPDC2000 (2000)
9. Buyya, R.: Economic-Based distributed resource management and scheduling for grid computing. [Ph.D. Thesis], Monash University, Melbourne (2002)
10. Bajari, P., Hortacsu, A.: Winner's Curse, Reserve Prices and Endogenous Entry: Empirical Insights From eBay Auction (2000), http://www.stanford.edu/~bajari/wp/auction/ebay.pdf

11. Resnick, P., Zeckhauser, R., Swanson, J., Lockwood, K.: The Value of Reputation on eBay: A Controlled Experiment. Experimental Economics 9(2), 79–101 (2006)
12. Smith, R.: The contract net protocol: high level communication and control in distributed problem Solver. IEEE Transactions on Computers 29, 1104–1113 (1980)
13. Czajkowski, K., Foster, I., Kesselman, C., Sander, V., Tuecke, S.: SNAP: A Protocol for negotiating service level agreements and coordinating resource management in distributed systems. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2002. LNCS, vol. 2537, pp. 153–183. Springer, Heidelberg (2002)
14. Czajkowski, K., Dan, A., Rofrano, J., Tuecke, S., Xu, M.: Agreement-based Service Management (WS-Agreement). Draft Global Grid Forum Recommendation Document (2003)
15. Bohte, S.M., Gerding, E., La Poutre, H.: Market-based Recommendation: Agents that Compete for Consumer Attention. ACM Transaction on Internet Technology (2000)
16. He, Y., Liu, H., et al.: A preference Method with Fuzzy Logic in Service Scheduling of Grid Computing. In: Wang, L., Jin, Y. (eds.) FSKD 2005. LNCS (LNAI), vol. 3613, pp. 865–871. Springer, Heidelberg (2005)

Informative Gene Selection and Tumor Classification by Null Space LDA for Microarray Data

Feng Yue, Kuanquan Wang, and Wangmeng Zuo

Biocomputing Research Center, The School of Computer Science and Technology,
Harbin Institute of Technology, Harbin, 150001, China
csfyue@gmail.com, wangkq@hit.edu.cn, cswmzuo@gmail.com

Abstract. DNA microarray technology can monitor thousands of genes in a single experiment. One important application of this high-throughput gene expression data is to classify samples into known categories. Since the number of gene often exceeds the number of samples, classical classification methods do not work well under this circumstance. Furthermore, there are many irrelevant and redundant genes which will decrease classification accuracy, thus a gene selection process is necessary. More accurate classification result using these selected genes is expected. A novel informative gene selection and sample classification method for gene expression data is proposed in this paper. This method is based on Linear Discriminant Analysis (LDA) in the regular space and the null space of within-class scatter matrix. By recursively filtering genes which have smaller coefficient in the optimal projection basis vectors, the remaining genes are more and more informative. The results of experiments on leukemia dataset and the colon dataset show that genes in this subset have much less correlations and more discriminative power compared to those selected by classical methods.

1 Introduction

DNA microarray technology provides the opportunity to simultaneously measure the expression level of thousands or tens of thousands of genes. These high-throughput data can provide valuable insights in molecular level. Various techniques, such as clustering [1,2], classification [3,4] and prediction [5], are used to analyze gene expression data to exploit the hidden information. One important application of gene expression data is to uncover the distinction between different types of cancer or tumor and normal samples, and classify samples into known categories accurately and reliably. After firstly considered by Golub *et al.* 4., varieties of gene selection and sample classification methods have been proposed [6~10], and many comparisons between different methods are conducted as well [11~13].

Gene expression data from DNA microarrays are characterized by many measured variables (genes) on only a few observations (experiments) [6] Among all of these genes, most are redundant or irrelevant. Thus gene selection, i.e. dimensionality reduction via feature selection, is needed. Methods of gene selection can be roughly grouped into two categories: filtering approaches and wrapper approaches [14]. The

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 435–446, 2007.

former evaluate goodness of genes individually, and a score is assigned to each gene. Then the top *d* genes with the highest score are chosen. In contrast with that, the wrapper approaches evaluate each gene subset instead of each gene, and select best *d*-gene subset according to current classifier. Although the filter approaches are simple and efficient, the wrapper approaches usually show a more accurate behavior [14,15].

As two examples of filtering approach, Golub *et al.* [4] use the correlation coefficient ranking scheme to select informative genes for a two-class problem, while Dudoit *et al.* [11] use the ratio of between-group to within-group sums of squares to evaluate each gene. Despite their linear time complexity in terms of gene number, these filtering methods assess gene individually and can not discover the redundancy and correlations between genes. The gene subset selected by these methods may be highly correlated. Other methods can perform the dimensionality reduction from a holistic view, such as PCA-based method [16] and PLS-based method [6], but they can only reduce original high dimensional data to only a few gene components, which are difficult to interpret. For wrapper approaches, the gene selection process is dependent on the choice of classifier.

Recently, Li *et al.* [15] propose a recursive gene selection method and have tested it on SVM, ridge regression and a Rocchio-style classifier. It recursively discards genes of small weight according to the selected classifier training by remaining genes. Most recently, an incremental wrapper-based gene selection method is proposed by Ruiz *et al.* [10], and has been tested on Naïve Bayes, IB1 and C4.5 classifiers. First it evaluates the discriminant power of each gene individually, and then recursively improve prediction performance by adding a gene one time until best performance is obtained. Although these methods can achieve very good performance using only a small number of genes, it lacks of a holistic view of genes and only selects genes in a greedy manner.

In this paper, we propose a novel wrapper-based gene selection method using LDA in the original space and the null space. We use the optimal projection basis vectors which maximize the ratio of the between-class scatter to the within-class scatter to capture the predictive power of every gene. More informative genes are expected to have larger coefficient in projection basis vectors. After recursively selecting genes with large coefficient, the remaining gene subset has minimum redundant and irrelevant genes. Because at every iteration the optimal projection basis vectors is obtained using all the remaining gene expression data, this method analyzes data from a holistic view and can eliminate more correlation and redundant genes. Experimental results show that this method is superior to some state-of-the-art filter-based and wrapper-based approaches.

The organization of the rest of this paper is as follows: In Section 2, the Null Space LDA based informative gene selection method will be introduced. Experimental results on two microarray datasets, and comparisons with two well-known filtering approaches will be presented in Section 3. Finally, conclusion and our future work will be given in Section 4.

2 Informative Gene Selection by Null Space LDA

In this section, we shall describe in detail the proposed informative gene selection method, as well as the Null Space LDA method it based on.

2.1 Fisher's Linear Discriminant Analysis(LDA)

Let the training set be composed of C classes, where each class contains N_i sample, i=1,2,…C, and let X_m^i be a d-dimensional column vector which denotes the mth sample from the ith class. In this case, the within-class scatter matrix S_W, between-class scatter matrix S_B ,and total scatter matrix S_T are defined as,

$$S_W = \sum_{i=1}^{C} \sum_{m=1}^{N_i} (x_m^i - \mu_i)(x_m^i - \mu_i)^T , \tag{1}$$

$$S_B = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^T , \tag{2}$$

and

$$S_T = S_W + S_B = \sum_{i=1}^{C} \sum_{m=1}^{N_i} (x_m^i - \mu)(x_m^i - \mu)^T , \tag{3}$$

where μ is the mean of all samples, and μ_i is the mean of samples in the ith class.

In Fisher's linear discriminant analysis theory [17], the best classification accuracy can be achieved by projecting the data into a new subspace where the ratio of between-class scatter matrix to within-class scatter matrix is maximized:

$$W_{opt} = \arg\max_{W} \frac{\left| W^T S_B W \right|}{\left| W^T S_W W \right|} \tag{4}$$

where W_{opt} is the best projection basis vectors. It can be shown that the column vectors of W_{opt} is eigenvectors of $S_W^{-1} S_B$ 17.

2.2 Null-Space LDA

Classical Fisher's linear discriminant analysis, however, usually encounters the small sample size problem, where the dimension of the sample space is much larger than the number of the samples in the training set [18]. This can lead to the singularity of S_W. In order to deal with this problem, numerous methods have been proposed in the last years [18~21]. Among these methods, the PCA + Null Space LDA method proposed by Huang *et al.* [19] stands out for its efficiency. In this method, after removing the null space of S_T, which contains the intersection of the null spaces of S_B and S_W, without losing discriminant information, the best predictive performance can be obtained when S_B is maximized in null space of S_W. With this optimal projection basis vectors, all training samples in one class can be projected to one common vector, so it can achieve 100% prediction accuracy for the training samples. More concretely, let U be the matrix whose columns are all the eigenvectors of S_T corresponding to the nonzero eigenvalues, then we get

$$S_W^{'} = U^T S_W U \tag{5}$$

and

$$S_B^{'} = U^T S_B U \tag{6}$$

Let Q be the null space of $S_W^{'}$, then we get

$$S_W^{''} = Q^T S_W^{'} Q = (UQ)^T S_W (UQ) = 0 \tag{7}$$

and

$$S_B^{''} = Q^T S_B^{'} Q = (UQ)^T S_B (UQ) \tag{8}$$

UQ is a subspace of the whole null space of S_W, and is really useful for discrimination. Let V be the matrix whose columns are all the eigenvectors of $S_B^{''}$ corresponding to the nonzero eigenvalues, then the final LDA projection basis vectors is $W=UQV$ [19]. Using this projection basis vectors, $W^T S_W W = \mathbf{0}$, $W^T S_B W \neq \mathbf{0}$ and is maximized. So the ratio of between-class scatter matrix to within-class scatter matrix is maximized. Because all the distinctions of training samples belonging to the same class are removed, we will obtain the same unique vector for all samples of the same class. We refer to this vector as the common vector [21].

2.3 Informative Gene Selection

In this subsection, we propose to use linear discriminant analysis for informative gene selection. While discriminant analysis is mainly utilized for feature extraction by previous studies [19,21], it is considered as an informative gene selection method for the first time. After getting the optimal projection basis vectors, previous researches usually project all samples on these vectors and then perform classification task, while we extract the information provided by these vectors to select informative genes.

The key idea of informative gene selection using Null Space LDA is that in the optimal projection basis vectors, the absolute value of each coefficient reflects the importance of that gene: the expression data should be magnified or minified to project training samples into common vectors. Thus the magnitude of this coefficient can be used as an indicator of informative genes: the smaller the absolute value of coefficient, the weaker informative power this gene has. Discarding a large number of non-informative genes will only lose little discriminant information. Only top few genes which correspond to larger coefficient are retained as informative genes.

Instead of discarding all the non-informative genes at one time, a more preferable scheme is to discard only a small number of genes which are most non-informative. After that, the best projection basis vectors is recalculated using remaining genes. The informative power of remaining genes should be updated according to this new projection basis vectors. By recursively filtering genes which correspond to small coefficient, the remaining genes become more and more informative, and the redundant and irrelevant genes are discarded step by step. When number of genes is less than that of samples, Null Space LDA would be inapplicable and the general LDA could be used.

Then the process of gene selection is performed according to best projection basis vectors of LDA, in the same manner, until only one gene left. The full algorithm is shown below.

```
Recursive gene selection Algorithm
1   Let M be the number of genes and N be the number
    of samples.
2   Let m be the number of remaining genes currently
3   m ← M
4   While m ≥ 1
5     If  m > N
6       Calculate W_opt by PCA + Null space LDA
7     Else
8       Calculate W_opt by general LDA
9     Calculate the weight of every gene using W_opt
10    Sort genes according to their weight
11    Select top K genes of large weight and set m ← K
```

The number of genes discarded at each iteration can be a tunable parameter. In experiments we set this value to $m/2$ and have very good results. Practically, this method will work well unless K is too small compared to m, in which case too much information is discarded.

3 Experimental Results

3.1 Experiments on Leukemia Dataset

The leukemia dataset contains gene expression data of two types of acute leukemia: acute lymphoblastic leukemia(ALL) and acute myeloid leukemia(AML) [4]. Gene expression levels were measured by using Affymetrix high-density oligonucleotide arrays containing 7129 genes. The dataset consist of 47 cases of ALL (38 B-cell ALL and 9 T-cell ALL) and 25 cases of AML, and is split into training samples (38 samples, 27 ALL and 11 AML) and test samples (34 samples, 20 ALL and 14 AML). This dataset is available at http://www.genome.wi.mit.edu/MPR. Following Dudoit et al [11] we preprocess this dataset by thresholding, filtering, a base 10 logarithmic transformation and standardization. Finally a matrix of 3571 genes by 72 samples is obtained.

First we use Golub's correlation coefficient, Dudoit's BW metric, and PCA + Null Space method to select top 50 most informative genes, then classify 34 test samples by nearest neighbor classifier in original space and projection space using these 50 genes. For Null Space LDA method, both recursive and non-recursive scheme are tested. In recursive gene selection scheme, half of currently remaining genes, i.e. 1786, 893, 447, 224, 112, 56 genes, are selected. Final 50 genes are selected from 56 genes in the same manner.

The results are given in Table 1. As can be seen, using top 50 genes selected by Golub's and Dudoit's metric, NN classifier can achieve very good prediction performance (94.12% and 94.12%, respectively). Although using 50 genes selected by Dudoit's metric 97.1% accuracy can be obtained by null space LDA, it seems that genes selected by Golub's metric are not suitable for null space LDA (only a 70.6% accuracy is achieved).Using top 50 genes selected by both non-recursive and recursive null space LDA can obtain good performance (91.2% and 97.1%, respectively) and result of recursive scheme is better. It seems that genes selected by these methods are not suitable for NN classifier in original space, either (accuracy are only 85.3% and 76.5%, respectively).

Table 1. Classification accuracy using top 50 Genes selected by different method

	Golub's metric	*Dudoit's Metric*	*Null Space LDA (non-recursive)*	*Null Space LDA (recursive)*
NN classifier in original space	0.94118 (32/34)	0.94118 (32/34)	0.85294 (29/34)	0.76471 (26/34)
NN classifier in projection space	0.70588 (24/34)	0.97059 (33/34)	0.91176 (31/34)	0.97059 (33/34)

Then we examine the relevance of genes selected using different methods. Following Li *et al.* [15], we show the correlation matrix of 50 genes selected by each method (Figure 1). In each matrix, the features are sorted according to their ranks assigned by the classifiers. The (i, j) element of the matrix is the absolute value of the correlation coefficient between the i-th feature vector and the j-th feature vector in the training data. The intensity in those graphs reflects the magnitude of gene-gene correlation coefficients: the brighter the gray-level, the stronger the correlation for either positively or negatively correlated genes.

From Figure 1 we can see the strong correlations between genes selected by Golub's and Dudoit's metrics. Moreover, stronger correlations between groups of genes which are indicators of different tumor type are visible in (A). But genes selected by both non-recursive and recursive null space LDA method have much less correlations. Compared with non-recursive scheme, recursive gene selection do have some influences in further reducing the correlations.

Figure 2(A) illustrates the prediction accuracy when different number of genes are selected. In this experiment, half of remaining genes are retained. When the number of the reserved genes is less than that of samples (namely, 38), LDA is used instead of null space LDA. Note that the accuracy could achieve 94.1% when only one gene is used. Using 7, 4, 2 genes can also obtain very good results of which the maximum number of mistaken classified sample is 3. The best performance is only 1 classification error, which is obtained when 28 or 56 genes are used.

The best results reported in [10] are 93.03% when 2.50 genes (in average of 10 10-fold cross-validation experiments) are retained using Nearest Neighbor classifier and 84.64% when 1.10 genes (in average of 10 10-fold cross-validation experiments) are retained using C4.5 decision tree classifier. But in our method, the result is 94.1% when only one gene is used, which has better accuracy and uses less gene. While in [15], the accuracy using one gene is roughly 60% (estimated from the performance graph).

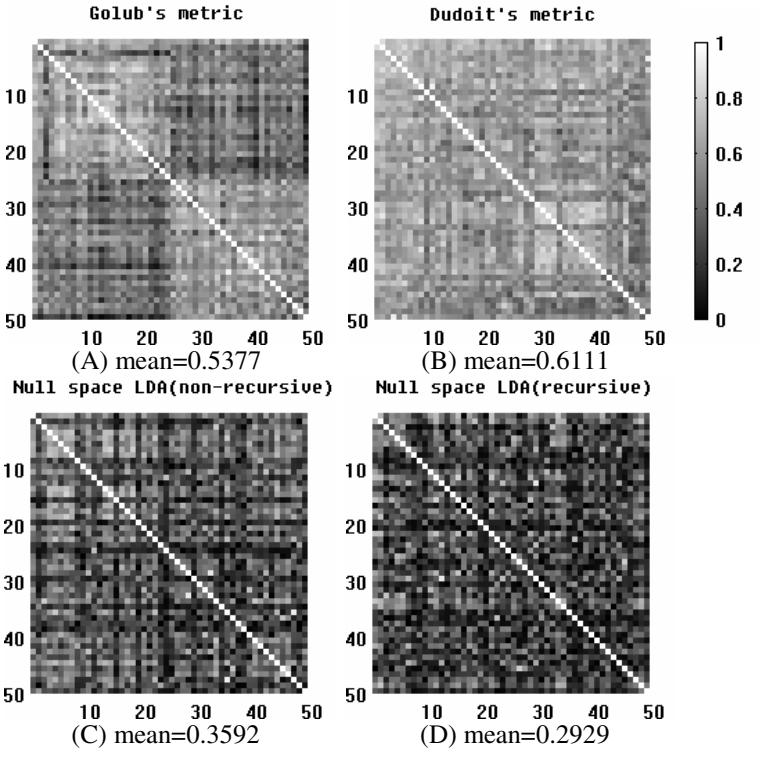


Fig. 1. Correlation matrix of 50 genes selected by different methods

While filtering half genes (Scheme I) seems arbitrary, we test other schemes to filter genes. The first is filtering only one gene at one time (Scheme II) and the second is filtering 100 genes each iteration until less than 100 genes are left, then filtering one gene at one time (Scheme III). When the number of remaining genes is close to the number of samples, the performance is instable. To solve this problem, we only retain 28 genes when 56 genes left, the same as filtering half genes scheme. After that genes are filtered one by one as before. Experimental results are show in Figure 2 (B) and (C), respectively.

Similar results are obtained as expected. Using less than 10 genes, the prediction accuracy can be consistently above 90%, even when only one or two genes are used. Besides that, other interesting tips can be learned from these graphs:

- When the number of the remaining genes drop from 56 to 28, the performance of our method is instable (green dashed line in Figure 2 (A) (B) and (C)). In Figure 2(A), the accuracy is not influenced, while in Figure 2 (B) and (C) the accuracy is decreased and increased, respectively. This instability may be due to the difference of projection basis vectors obtained by null space LDA and LDA, but the influence of it is very limited: after a few iterations using LDA, the accuracy will increase gradually (Figure 2 (B)).

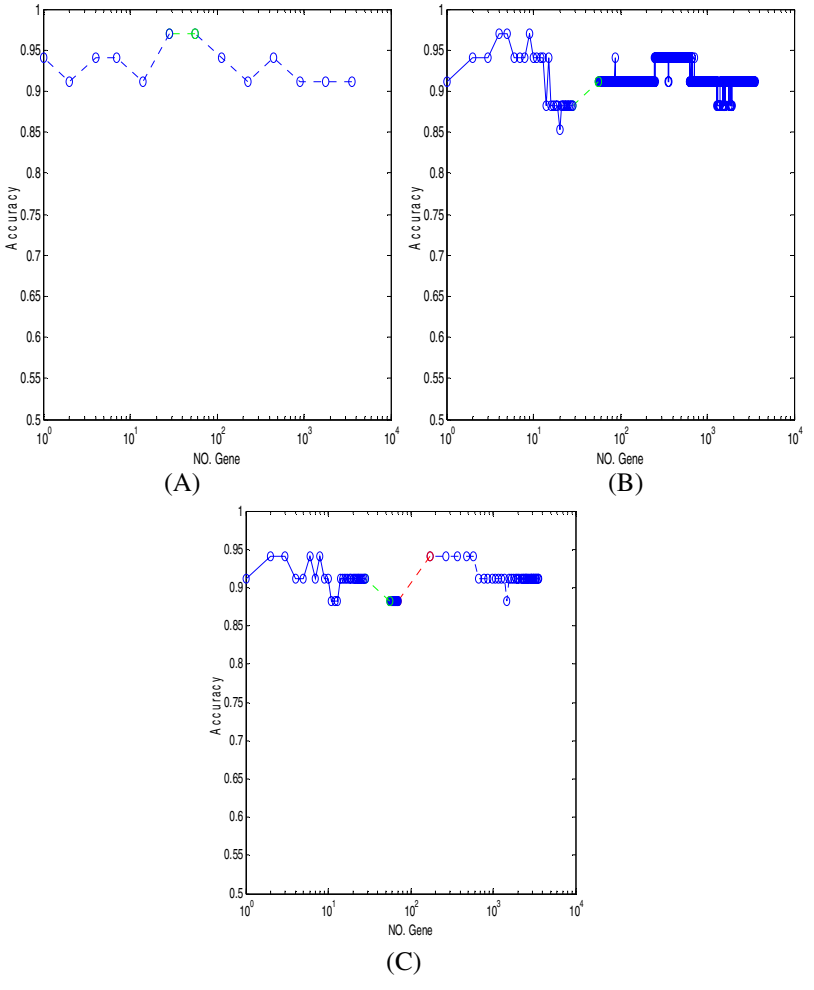


Fig. 2. Classification accuracy versus number of genes retained using null space LDA of different schemes. (A) Scheme I. Filter half genes at one time. (B) Scheme II. Filter a gene at one time. (C) Scheme III. Filter 100 genes at one time.

- When too many genes are dropped compared to remaining genes, the performance of our method will drop significantly. In Figure 2 (C) when we retain 71 genes from 171 genes, the accuracy decreases from 94.12% to 88.24% (red dashed line). The reason is that too much information is lost; the retained genes are not informative enough. Also, the decline of performance is not fatal: after a few iterations of filtering only one gene at one time, the accuracy will increase gradually (Figure 2 (B)). But if we always dropped too many genes, the performance will drop rapidly accordingly.

3.2 Experiments on Colon Dataset

The colon dataset reported by Alon et al. [2] is composed of 62 (40 tumor and 22 normal) samples of colon epithelial cells. The 'tumor' (40 samples) biopsies were collected from tumors, and the 'normal' (22 samples) biopsies were collected from healthy parts of the colons of the same patient. Gene expression levels were measured using high-density oligonucleotide arrays. Of about 6000 genes represented in these arrays, 2000 genes were selected based on the confidence of the measured expression levels.This dataset is available at http://microarray.princeton.edu/oncology/affydata/index.html.

We use Golub's correlation coefficient, Dudoit's BW metric, and the PCA + Null Space method to select top 100 and 200 most informative genes. Using Leave-One-Out cross validation we classify 62 samples by nearest neighbor classifier in original space and projection space using these 100 or 200 genes. For recursive Null Space LDA method, 100 genes are filtered one time, i.e. the number of remaining genes are 1900, 1800, 1700, … 300, 200, 100.

Table 2. Classification accuracy using top 100 Genes selected by different method

	Golub's metric	*Dudoit's Metric*	*Null Space LDA (non-recursive)*	*Null Space LDA (recursive)*
NN classifier in original space	0.80645 (50/62)	0.80645 (50/62)	0.83871 (52/62)	0.85484 (53/62)
NN classifier in projection space	0.62903 (39/62)	0.67742 (42/62)	0.82258 (51/62)	0.80645 (50/62)

Table 3. Classification accuracy using top 200 Genes selected by different method

	Golub's metric	*Dudoit's Metric*	*Null Space LDA (non-recursive)*	*Null Space LDA (recursive)*
NN classifier in original space	0.77419 (48/62)	0.79032 (49/62)	0.77419 (48/62)	0.80645 (50/62)
NN classifier in projection space	0.75806 (47/62)	0.70968 (44/62)	0.82258 (51/62)	0.83871 (52/62)

Experimental results are given in Table 2 and Table 3 respectively. From Table 2, we can see that Using top 100 genes selected by Golub's and Dudoit's metrics, predicted performance achieved by NN classifier are 80.65%. But these genes are not suitable for null space LDA classifier (only 62.90% and 67.74 are achieved). Using top 100 genes selected by both non-recursive and recursive null space LDA can also obtain good performance (82.26% and 80.65%, respectively) and result of the non-recursive scheme is better. But genes selected by these methods are more suitable for NN classifier in original space (accuracy are 83.87% and 85.48%, respectively). The results obtained by top 200 genes (shown in Table 3) are similar with that of top 100 genes using Golub and Dudoit's metric. But this time the advantage of using null space LDA method is evident. Both non-recursive and recursive scheme can obtain a much better performance.

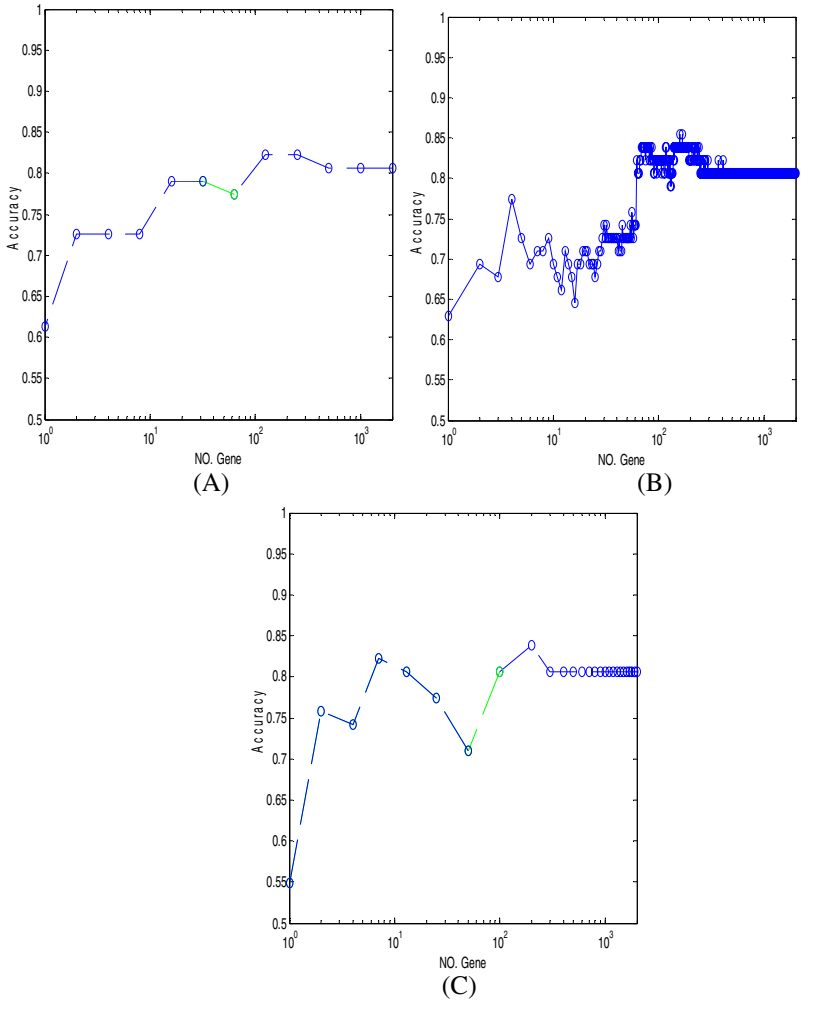


Fig. 3. Classification accuracy versus number of gene retained using null space LDA of different scheme. (A) Scheme I. Filter half genes at one time. (B) Scheme II. Filter a gene at one time. (C) Scheme III. Filter 100 genes at one time.

Because no explicit training and test set, we can not select consistent informative genes for all LOOCV experiments. Consequently we can not depict the graph of correlations of selected genes. But it can be expected that in each LOOCV experiment, informative genes selected by null space LDA have less correlations than those selected by Golub and Dudoit's metrics.

We also test other schemes to filter genes besides Scheme I. Experimental results are show in Figure 3 (B) and (C), respectively.

The results are similar with those obtained on leukemia dataset. When the number of remaining genes is close to the number of samples, the performance would be instable (green dashed line in Figure 3 (A) and (C)). Compared with results of leukemia

dataset, we can not get very good prediction performance using only 1 or 2 genes. Prediction accuracy over 80% can only obtained by using 100 informative genes or more. It is probably due to the complexity of this classification task.

4 Conclusion and Future Work

A novel informative gene selection and sample classification method for gene expression data is proposed in this paper, which can select a small number of informative genes from all genes. Compared with other state-of-art methods, it can produce more accurate classification result using these informative genes as features when classification tumor samples. Experimental results also show that recursive Null-Space LDA method is more accurate and more robust than non-recursive scheme. Our future work includes how to deal with the instability when the number of informative genes is close to the size of samples and how to incorporate the information provided by the regular space and the null space of total scatter matrix.

Acknowledgment

The work is supported in part by the NSFC foundation under the contracts No. 60571025, the 863 project under the contracts No. 2006AA01Z308.

References

1. Michael, B.E., Paul, T.S., Patrick, O.B., David, B.: Cluster analysis and display of genome-wide expression patterns. Proc. Natl. Acad. Sci. USA 95, 14863–14868 (1998)
2. Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D., Levine, A.J.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc. Natl. Acad. Sci. USA 96, 6745–6750 (1999)
3. Laura, J.V., Hongyue, D., Marc, J.V., Yudong, D.H., Augustinus, A.M., Mao, M., Hans, L.P., Karin, K., Matthew, J.M., Anke, T.W., George, J.S., Ron, M.K., Chris, R., Peter, S.L., Rene, B., Stephen, H.F.: Gene expression profiling predicts clinical outcome of breast cancer. Nature 415, 530–536 (2002)
4. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., BloomTeld, C.D., Lander, E.S.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science 286, 531–537 (1999)
5. Douglas, T.R., Uwe, S., Michael, B.E., Charles, M.P., Christian, R., Paul, S., Vishwanath, I., Stefanie, S.J., Matt, V.R., Mark, W., Alexander, P., Jeffrey, C.F., Deval, L., Dari, S., Timothy, G.M., John, N.W., David, B., Patrick, O.B.: Systematic variation in gene expression patterns in human cancer cell lines. Nature Genetics 24, 227–235 (2000)
6. Danh, V.N., David, M.R.: Tumor classification by partial least squares using microarray gene expression data. Bioinformatics 18, 39–50 (2002)
7. Antoniadis, S., Lambert, L., Leblanc, F.: Effective dimension reduction methods for tumor classification using gene expression data. Bioinformatics 19, 563–570 (2003)
8. Sun, M., Xiong, M.: A mathematical programming approach for gene selection and tissue classification. Bioinformatics 19, 1243–1251 (2003)

9. Guan, Z., Zhao, H.: A semiparametric approach for marker gene selection based on gene expression data. Bioinformatics 21, 529–536 (2005)
10. Roberto, R., José, C.R., Jesús, S.A.: Incremental wrapper-based gene selection from microarray data for cancer classification. Pattern Recognition (in press)
11. Dudoit, S., Fridlyand, J., Terence, P.S.: Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data. Journal of the American Statistical Association 97, 77–87 (2002)
12. Tao, L., Zhang, C., Mitsunori, O.: A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. Bioinformatics 20, 2429–2437 (2004)
13. Statnikov, A., Constantin, F.A., Tsamardinos, I., Hardin, D., Levy, S.: A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. Bioinformatics 21, 631–643 (2005)
14. Inza, I., Larranaga, P., Blanco, R., Cerrolaza, A.J.: Filter versus wrapper gene selection approaches in DNA microarray domains. Artificial Intelligence in Medicine 31, 91–103 (2004)
15. Li, F., Yang, Y.: Analysis of recursive gene selection approaches from microarray data. Bioinformatics 21, 3741–3747 (2005)
16. West, M., Blanchette, C., Dressman, H., Huang, F., Ishida, S., Spang, R., Zuzan, H., Olason, J., Marks, I., Nevins, J.: Predicting the clinical status of human breast cancer by using gene expression profiles. Proc. Natl. Acad. Sci. USA 98, 11462–11467 (2001)
17. Fisher, R.A.: The Use of Multiple Measures in Taxonomic Problems. Ann. Eugenics 7, 179–188 (1936)
18. Chen, L.F., Liao, H.Y., Ko, M.T., Lin, J.C., Yu, G.J.: A New LDA-Based Face Recognition System Which Can Solve the Small Sample Size Problem. Pattern Recognition 33, 1713–1726 (2000)
19. Yu, H., Yang, J.: A Direct LDA Algorithm for High-Dimensional Data with Application to Face Recognition. Pattern Recognition 34, 2067–2070 (2001)
20. Huang, R., Liu, Q., Lu, H., Ma, S.: Solving the Small Size Problem of LDA. Proc. 16th Int'l Conf. Pattern Recognition 3, 29–32 (2002)
21. Hakan, C., Marian, N., Mitch, W., Atalay, B.: Discriminative Common Vectors for Face Recognition. IEEE Trans. PAMI 27, 4–13 (2005)

Heuristic Search for 2D NMR Alignment to Support Metabolite Identification

Geun-Cheol Lee[1], Jeff de Ropp[2], Mark R. Viant[3], David L. Woodruff[2], and Ping Yu[2]

[1] Konkuk University; 1 Hwayang-dong; Gwangjin-Gu; Seoul; Korea
gclee@konkuk.ac.kr
[2] University of California, Davis; One Shields Avenue; Davis CA 95616; USA
jsderopp,dlwoodruff,pyu@ucdavis.edu
[3] The University of Birmingham, Birmingham, UK
m.viant@bham.ac.uk

Abstract. For the problem of aligning two-dimensional NMR spectra of biological samples to determine if metabolite standards in a database can be excluded as possible constituents, we develop heuristic search algorithms that offer tremendous time savings when compared to manual methods. Xi et al [15] consider this problem and use statistical methods to reduce the search space and enumerate it. In this paper we consider the case when the statistical model is not available due to lack of data. We describe a simulated annealing algorithm and an algorithm that hybridizes simulated annealing with a shift neighborhood and a variant of reactive tabu search with a large neighborhood. Computational experiments based on data from physical experiments demonstrates that the hybrid is more effective than its constituents for runs with limited CPU time but that simulated annealing and the hybrid are roughly equal for longer runs.

1 Introduction

For the problem of aligning two-dimensional NMR spectra of biological samples to determine if metabolite standards in a database can be excluded as possible constituents, we develop heuristic search algorithms that offer tremendous time savings when compared to present manual methods. Xi et al [15] consider this problem and use statistical methods to reduce the search space an enumerate it. In this paper we consider methods that are effective when the statistical model is not available due to lack of data. Also, it provides an interesting and important problem for testing metaheuristics.

A large neighborhood structure for shifting the spectra is used by a reactive tabu search [2]. We make use of a stopping criteria based loosely on the idea of freezing in simulated annealing (SA). The large neighborhood structure requires expensive computations, so it is not appropriate for SA itself. For an implementation of a simulated annealing that uses the cooling schedule of Johnson et al

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 447–458, 2007.

[6], we make use of a neighborhood structure that is computationally tractable enough to facilitate the sampling that is the basis of SA. Following Voß and Fink [13], we use SA as the escape mechanism in the reactive tabu search algorithm. The resulting hybrid algorithm is better and faster than either of its components for runs using limited CPU time when tested on data from physical experiments. For longer runs, there is little difference between the hybrid and simulated annealing.

Two dimensional Nuclear Magnetic Resonance (2D NMR) analysis [7] makes use of a technology where samples are placed in a magnetic field that forces orientation of the the nuclei of hydrogen atoms. Application of a radio frequency pulse forces the nuclei out of that alignment. The signals induced as the nuclei realign are recorded. When this is repeated with systematically varying time of the acquired signal, the result, after some transformation, is an x,y,z plot. Each peak on the main diagonal of the plot corresponds to a chemical group, with the z-value corresponding to the number of hydrogen nuclei present in the peak. In a COrrelation SpectroscopY (COSY) spectrum, which is our interest here, peaks on the off-diagonal indicate spin coupling of the two corresponding peaks on the main diagonal. Hence, topographic plots are assumed to be symmetric.

Many chemical compounds are identifiable by their signatures on these graphs, which makes the technology potentially very interesting for use in analysis of metabolites. At present, the identification of the metabolites is a major bottleneck in the efforts to automate metabolomic analysis. Depending on the type of 2D spectrum collected, the time to acquire a spectrum is on the order of minutes or hours but the time to identify compounds is on the order of days. Present methods of identification are primarily based on a person picking through the peaks visually.

One of the things that adds difficulty to automating this process for COSY spectra is that the peak for a given chemical group may be shifted slightly from one sample to the next. The primary causes of shifts in peak position are variations in sample pH (despite buffering) and sample temperature (despite fairly precise control of sample chamber temperature). The pH effect is particularly difficult to remove, and real metabolomic samples have a small variance in pH even in the presence of added buffer.

For two dimensional spectra, the presence of off-diagonal peaks makes the problem of compensating for shifts complex and computationally very expensive. With COSY spectra, it is valuable to rule out compounds using a simple model where the compound signatures are shifted. If the compound signatures cannot be shifted in reasonable fashion to match the signatures found for a sample, then no reasonable shifting of the sample peaks will result in a match either. The chemical compound signatures are, generally, many orders of magnitude simpler than the signatures from a biological sample so shifting them can be done much more quickly.

The consideration of peak shifts in 2D NMR spectra is complicated by the fact that two peaks on the diagonal are often coupled by an off-diagonal peak. Hence, shifting one peak can result in shifts of others. We have an algorithm that

matches the graph for compounds with the graph for a sample and finds near optimal shifts for the compounds so they can be ruled out as being above the detection limit in spectrum for the sample or retained for further consideration. We tested it by developing a database of 2D COSY NMR spectra of the 20 naturally occurring amino acids. Experiments searching for amino acid peaks in COSY data generated from samples reveal that we can rule out compounds using an optimization algorithm that requires less than a minute per compound.

In this paper we will concern ourselves with comparing z-values that are considered to be above the detection limit. Hence, the z-dimension is binary, which yields very simple and well-defined topographic plots that show the location of peaks that are above the detection limit. If a compound is present in a sample, then all of the peaks for the compound will be present above detection in the sample, perhaps after some shifting. In the following section we provide a formulation that allows us to rule out this possibility for some compounds. In §3 we describe algorithms to search for solutions; tests on the algorithms are described in §4. The paper closes with conclusions and directions for further research.

2 Formulation

We use the same formulation as Xi et al [15]. When describing the formulations, we use the terms *pixels* and x, y pairs interchangeably to refer to points in the graph with z values that are above the detection limit.

2.1 Data for a Compound

There are three related database views for each compound in the database.

- $\mathcal{G}$: the set of all x, y pairs that are above the detection limit for the compound; $|\mathcal{G}|$ gives the number of them.
- $\mathcal{O}$: a vector where each element corresponds to a peak and is the set of x, y pairs for the pixels that is the peak. The number of peaks for compound is given by $|\mathcal{O}|$. Note that $\cup_{i=1}^{|\mathcal{O}|} \mathcal{O}_i = \mathcal{G}$.
- $\mathcal{B}$: each entry is a database of information about a band on the x-axis. In a computer implementation, this would have the x-value for the left side of the band, the x-value for the right side, references to entries in $\mathcal{G}$ and $\mathcal{O}$ for all peaks in the band and references to entries in $\mathcal{O}$ that are for peaks that are not in the band, but that are spin coupled with the peak in the band on the diagonal axis. The number of bands is given by $|\mathcal{B}|$. Since off-diagonal peaks occur only when diagonal peaks are spin coupled, every band has exactly one on-diagonal peak.

2.2 Decision Variables

A vector of band shifts, b, of length $|\mathcal{B}|$ constitutes the decision vector. Each element gives the number of pixels that the corresponding band is shifted, with

a positive number understood to be a right shift and a negative number a left shift. Because of the symmetry, without loss of generality we consider shifting only the x bands. We constrain the shift vector to avoid having band boundaries cross. Furthermore, we assume a user specified limit, B, on the number of pixels that each band can be shifted in either direction. The set of all feasible shift vectors will be denoted by β.

The function $S(b, \cdot)$ returns the x, y list view of the compound given by its second argument modified to reflect the shift given by its first. E.g., $S(b, \mathcal{G})$ is the list of x, y pairs for the compound assuming shift vector b has been applied. The fact that all off-diagonal peaks are coupled with diagonal peaks and the presence of the list $\mathcal{B}$ makes this simple to implement. For example, a movement to the right of one pixel, causes all peaks in the band to move right one pixel. The peak on the diagonal moves up one pixel as do the off-diagonal peaks that are not in the band but are spin coupled with the diagonal peak in the band.

2.3 Data from an Experiment

Each experiment results in a set of x, y pairs for points that are above the detection limit. Call that list $\mathcal{E}$.

2.4 Objective Function

We now define two counting functions. In keeping with common slang, we will refer to these functions as counting the number of pixels or peaks that *hit*:

- $\delta(\mathcal{O}, \mathcal{E}, b)$: for a shift vector b this function returns the number entries (i.e., peaks) in $\mathcal{O}$ for which there is at least one x, y pair that is in both $S(b, \mathcal{O})$ and $\mathcal{E}$.
- $\gamma(\mathcal{G}, \mathcal{E}, b)$: for a shift vector b this function returns the number of x, y pairs that are in both $S(b, \mathcal{G})$ and $\mathcal{E}$.

If we treat the lists as sets, then $\gamma(\mathcal{G}, \mathcal{E}, b) \equiv |\{S(b, \mathcal{G}) \bigcap \mathcal{E}\}|$. Description of $\delta(\cdot)$ requires more notation. Let $S_i(b, \mathcal{O})$ be the set of x, y pairs that are above detection for peak i after shift vector b has been applied. We use $I(\cdot)$ as an *indicator function*, which takes the value one if its argument is true and zero otherwise. With this notation, $\delta(\mathcal{O}, \mathcal{E}, b) \equiv \sum_{i=1}^{|\mathcal{O}|} I(\{S_i(b, \mathcal{O}) \bigcap \mathcal{E}\} \neq \emptyset)$

Since the goal is to rule out the possibility that any shifting of the sample could result in a match for a particular database entry, we need an objective function that gives an ability to put a lower bound on this possibility. For this purpose, the first term in the objective function is $100\frac{\delta(\mathcal{O}, \mathcal{E}, b)}{|\mathcal{O}|}$, which is the percent of the peaks for which there is at least one hit under shift vector b. Database compounds with a score less than 100 for all reasonable b are unlikely to be in the sample.

The number of pixels that hit is of secondary importance. This information can help to prioritize compounds for further interpretation. To capture this in a simple, heuristic way, we add the term $\frac{\gamma(\mathcal{G}, \mathcal{E}, b)}{|\mathcal{G}|}$. Leaving this term scaled as a

fraction and the first term as a per cent results in a lexicographic ordering of the objective function values. A significant side benefit is that this term also helps guide the search through the large regions of the search space where the first term is constant. Only those compounds with an objective function value over one hundred could be in the sample. Very roughly, those with a higher value tend to be a better match than those with a lower value.

Maximization of these two terms, subject to the constraint that b remain feasible provides a fully specified optimization problem, which is to maximize $f(b) \equiv 100\frac{\delta(\mathcal{O},\mathcal{E},b)}{|\mathcal{O}|} + \frac{\gamma(\mathcal{G},\mathcal{E},b)}{|\mathcal{G}|}$ subject to $b \in \beta$.

3 Local Search Algorithms

As the name implies, local search algorithms are based on the idea of perturbations of a *current solution*. We sometimes refer to a perturbation of a solution as a *move*. All the solutions that can be reached in one move from a solution, b, are said to be in its *neighborhood*, $\mathcal{N}(b)$. We will concern ourselves here only with neighborhoods that preserve feasibility (i.e., $\mathcal{N}(b) \subset \beta$ for all $b \in \beta$).

Starting from some initial solution $b^{(0)}$, a *hill climbing* algorithm (which are sometimes called *greedy* or *steepest descent* when minimizing) proceed at iteration $k \geq 1$ as follows $b^{(k)} := argmax_{b \in \mathcal{N}(b^{(k-1)})} f(b)$ and terminates when for all $b \in \mathcal{N}(b^{(k-1)})$, $f(b) \leq f(b^{(k-1)})$. The algorithm terminates at a solution that is by definition a local maximum of $f(\cdot)$ with respect to the neighborhood.

Consider two neighborhood structures. An obvious neighborhood consists all those solutions that differ by one in one vector element (without violating constraints). Call this neighborhood N1. A natural one-dimensional characterization of moves that imply N1 is simply the band number for a move to the right and the negative of the band number for a move to left. A generalization of this neighborhood structure has a more natural two-dimensional characterization: a band number and the change in the vector element for the band. Call this neighborhood where one band can move any feasible distance N2.

3.1 Greedy Algorithms and Large Neighborhoods

It seems intuitive that algorithms based on N2 will dominate those based on N1, but to verify that we implemented a greedy algorithm based on each. Call these algorithms G1 and G2 respectively.

We also implemented greedy algorithms based on a band by band decomposition of N2. Algorithm D1 optimizes the value for each vector element independently, considering each only once in order. More formally, the algorithm proceeds as follows for bands $i = 1, \ldots, |\mathcal{B}|$ in order:

$$
\begin{array}{l}
b^{(i)} = argmax_{b_i} f(b) \\
\text{subject to:} \\
b_j = b_j^{(i-1)}, \;\; 1 \leq j < i, \; i < j \leq |\mathcal{B}| \\
b \in \beta
\end{array}
$$

Algorithm D2 simply repeats algorithm D1 using the result of the last execution as the starting point for the next until no further improvement is made by an execution of D1. Thus D1 can be thought of as defining a neighborhood over which greedy algorithm D2 operates. This is similar to the ideas expressed by the literature on very large neighborhoods [1] and compound neighborhoods [5] (although the label "compound" neighborhood is more descriptive than "large" in this case we will eschew it due to potential confusion with chemical compounds). We will refer to D1 both as an algorithm and as a neighborhood.

3.2 Simulated Annealing

Simulated Annealing SA is a metaheuristic for complex optimization problems [8]. We have adopted algorithm statement shown in Figure 1, which based on the work of Johnson et al [6], except that we make use of problem specific information to set the initial temperature. In the canonical version of SA, the initial temperature is set by trying temperatures until one is found that results in approximately the desired rate of move acceptance. We save computational effort by setting the initial temperature at $T^{(0)} \leftarrow 102 - f(0)$, where $f(0)$ is the objective value for the compound with no shifting. In the sequel, we refer to this implementation of simulated annealing with neighborhood N2 as algorithm SA.

The function Frozen(MinPercent) returns true if five temperatures in a row result in less than MinPercent percent acceptance of moves. For our experiments we fixed MinPercent at 2 as recommended by Johnson et al. The function BestCheck() simply keeps track of the best solution seen so far. The temperature is reduced by multiplying the current temperature by the parameter TempFactor. The number of iterations at each temperature is the parameter SizeFactor times the neighborhood size.

3.3 Tabu Search

Tabu Search (TS) algorithms [4,5] select moves according to a hill climbing scheme modified by a *tabu list* to force the search away from the reversal of the attributes of moves selected for recent iterations.

The form of our TS algorithm is shown in Figure 2. This is a conceptual description; the implementation is organized for greater efficiency. We applied tabu search using the compound neighborhood G1.

The functions Tabu() and UpdateMoveArray() work together. The MoveArray has a row for each band and a column for each shift of the band that has been tried. The entries give the change in the corresponding b vector element for the corresponding move. For the purpose of describing tabu restrictions, call this array A, so the change in b_i in iteration j is given by A_{ij}.

The function Tabu() implements tabu restrictions and aspiration criteria. We conducted preliminary experiments with two types of tabu restrictions. The first, T1, forbids moves at iteration k to vectors b where $b_i - b_i^{(k-1)} = -A_{ij}$ for $\max(1, k - \kappa) \leq j \leq k - 1$. With this restriction, it is not possible for a single move to directly reverse another move. This is generalized under tabu restriction

Begin with $b_i = 0, i = 1, \ldots, |\mathcal{B}|$
$T \leftarrow T^{(0)}$
REPEAT
 REPEAT SizeFactor $* |\mathcal{N}(b)|$ times
  Randomly Select $b' \in \mathcal{N}(b)$ (a neighbor)
  $\Delta \leftarrow \hat{f}(b') - \hat{f}(b)$ (change in Obj)
  IF $(\Delta \geq 0)$ OR $(\exp(\Delta/T) <$ URan(0,1)) (good enough?)
   $b \leftarrow b'$ (move)
   BestCheck(b)
 $T \leftarrow$ TempFactor $* T$ (cool)
UNTIL Frozen(MinPercent)

Fig. 1. Simulated Annealing Algorithm (SA)

T2, where a move is forbidden it would reverse the cumulative effect of any of the last κ moves. I.e., moves are forbidden if the resulting vector b has the property

$$b_i - b_i^{(k-1)} = -\sum_{\ell=j}^{k-1} A_{i\ell}$$

for $\max(1, k-\kappa) \leq j \leq k-1$. Our experiments indicated that T2 performs slightly worse than T1 so in the sequel, results are reported only for tabu restrictions based on T1.

We also define two types of aspiration criteria. Criterion A1 overrides tabu status if the move would result in the best solution found so far. Criterion A2 overrides tabu status if the move would result in the best solution seen as the result of change for the vector element under consideration.

The function FindBestOK() search neighborhood D1 subject to tabu restrictions. Hence, the function proceeds as follows for bands $i = 1, \ldots, |\mathcal{B}|$ in order:

$$\begin{aligned} &b^{(i)} = argmax_{b_i} f(b) \\ &\text{subject to:} \\ &b_j = b_j^{(i-1)}, \;\; 1 \leq j < i, \; i < j \leq |\mathcal{B}| \\ &b \in \beta \\ &\text{not Tabu(b)} \end{aligned}$$

The function EndCriteria() signals termination when there have been TabuStop solutions in a row without improvement.The function LTMem() implements so-called long term memory. A single parameter variant of reactive tabu search [2] is used. Solutions are hashed [14] and the hash values are stored. When a pattern of three hash values is repeated, the search is restarted with a random

```
Begin with b_i = 0, i = 1, ..., |B|
REPEAT
 b' ← FindBestOK()
 UpdateMoveArray(b, b')        (record changes)
 b ← b'                        (make the move)
 BestCheck(b)
 LTMem(b)
UNTIL EndCriteria()
```

Fig. 2. Tabu Search

value of $b \in \beta$. The function LTMem() also halves the value of TabuStop (if it is greater than 2) whenever a new best solution is found.

3.4 Hybrid

Given the notation that we have developed, it is now possible to provide a compact description of an effective hybrid algorithm. Algorithm HY combines TS and SA, and by implication neighborhoods D1 and N2. Following Voß and Fink [13], we use SA as the escape mechanism in the reactive tabu search algorithm TS. Instead of just restarting TS with a random value when a sequence of three repeated hash values are encountered, we execute SA starting with the current solution. The advanced start of SA typically causes much faster freezing than a random start. Upon termination of SA, TS is resumed from the last solution visited by SA.

4 Computational Experiments

In order to test the algorithms, two dimensional (2D) magnitude COSY NMR spectra were measured using an Avance DRX-500 spectrometer (Bruker, Fremont, CA) running XWINNMR software version 3.1. We recorded spectra for the twenty naturally occurring amino acids in order to create the database access functions described herein. In order to replicate these experiments, the following data are needed: Amino acid spectra were obtained for 10 mM solutions of the free amino acid in ^2D_2O, buffered to pH 7.4 with 0.2 M sodium phosphate. We also recorded 2D COSY spectra for two different samples from a shellfish (red abalone, *Haliotis rufescnes*) to create two different $\mathcal{E}$ structures: a sample of foot muscle and of digestive tissue. The abalone sample extraction and preparation are described in [12]. All data were obtained at 295K and referenced to internal TMSP (1 mM) at 0.00 ppm. Magnitude COSY spectra were collected with 1024 points in t2 and 256 points in t1 over a bandwidth of 14 ppm. The resulting

NMR spectra were processed in XWINNMR using standard methods and zero filled in t1 yielding a transformed dataset of 1024 by 1024 points.

The code for algorithms was executed by Matlab version 6 on a 1.79 GHz Athalon MP2200 processor running under Linux. We used $B = 10$ as user-specified bound on the shift function, which is $10/1024 \times 14 = 0.137$ppm. A constant detection limit was used for the compound database and another for the two samples. In both cases, the detection limit was obtained by visual inspection.

By running 20 amino acids against two samples, we obtain 40 runs that we can use for comparison. Detailed results are given in the Appendix. We summarize the results here.

All of the heuristic search algorithms are many orders of magnitude fast than processing "by hand." For lower CPU times the hybrid outperforms its constituent algorithms. A qualitatively important result not shown in the table is that the heuristic search algorithms find 11 acids that cannot be ruled out with fewer than 10 for the simple heuristic runs described in Table 1.

Some additional observations are as follows: The hybrid algorithm gets better results for low CPU times than SA or TS. The sensitivity to the cooling rate is not very strong, but slower cooling generally tends to improve the quality of the solution at the expense of more CPU time. The effect of the initial value of TabuStop is direct: the algorithm runs longer and therefore tends to find better solutions on average.

5 Conclusions

For the problem of aligning two dimensional NMR spectra to see if compounds in a database can be ruled out as being above the detection limit of the NMR device, we develop heuristic search algorithms that offer tremendous time savings over present manual methods. The hybrid algorithm performs well and has a parameterization that allows users to control the time/quality tradeoff in an intuitive way. In a few seconds, a compound can be ruled out of further consideration or its priority for further analysis estimated.

There are, of course, plenty of opportunities for more research in this area. Comparing spectra from samples to determine which peaks may be biomarkers generates problems that are closely related to those considered in this paper. One area of ongoing research is the problem of aligning spectra from two samples. For one dimensional spectra, there are simple, computationally fast methods for alignment that are used in practice [3,9]. The presence of coupling makes it impossible to apply these algorithms naively in the two-dimensional case. The algorithms that we have presented do not generalize to this problem for practical reasons: for the spectra generated from bio-samples (as opposed to those from a single compound) large sections of the diagonal are densely populated with peaks, many of which are superimposed or overlapping. Hence, the generation of the bands that can be moved independently is problematic. More sophisticated approaches based on correlations of individual peaks such as the one proposed by [11] are computationally too expensive to apply to 2D spectra in many settings.

A promising line of research for the sample alignment problem is the use of warping techniques that were developed for image processing [10].

In this paper, we have provided such algorithms with the intention of adding to the tools available for automated identification of compounds in NMR samples. A large neighborhood is shown to be effective for simple local search, and is the basis for an effective variant of reactive tabu search. However, a simple neighborhood seems to be just as effect when the CPU allocated to the problem is large. The most effective algorithm for short runs is provided by a hybrid with reactive tabu search based on a large neighborhood that uses simulated annealing with an appropriate shift neighborhood. This algorithm that makes use of two meta-heuristics based on different neighborhoods is shown to be much more effective than its constituents alone for such runs.

Acknowledgment

This publication was made possible in part by grant number 5 P42 ES04699 from the National Institute of Environmental Health Sciences, NIH. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the NIEHS, NIH. It is also supported by NIH grant RO1 HG003352-01A2. MRV is grateful to the Natural Environment Research Council, UK, for an Advanced Fellowship (NER/J/S/2002/00618).

References

1. Ahuja, R.K., Ergun, O., Orlin, J.B., Punnen, A.P.: A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics 123, 75–102 (2002)
2. Battiti, R., Tecchiolli, G.: The Reactive Tabu Search. ORSA Journal on Computing 6, 126–140 (1994)
3. Forshed, J., Schuppe-Koistinen, I., Jacobsson, S.P.: Peak Alignment of NMR Signals by Means of a Genetic Algorithm. Analytica Chimica Acta 487, 189–199 (2003)
4. Glover, F.: Tabu Search - Part I. ORSA Journal on Computing 1, 190–206 (1989)
5. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publisher, Boston, Dordrecht, London (1997)
6. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning. Operations Research 37, 865–892 (1989)
7. Keeler, J.: (2002), `http://www-keeler.ch.cam.ac.uk/lectures/` *understanding/chapter7.pdf*
8. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220(4598), 671–680 (1983)
9. Lee, G.-C., Woodruff, D.L.: Beam Search for Peak Alignment of NMR Aignals. Analytica Chimica Acta 513(2), 413–416 (2004)
10. Tomasi, G., van den Berg, F., Andersson, C.: Correlation Optimized Warping and Dynamic Time Warping as Preprocessing Methods for Chromatographic Data. Journal of Chemometrics 18, 231–241 (2004)

11. Stoyanova, R., Nicholls, A.W., Nicholson, J.K., Lindon, J.C., Brown, T.R.: Automatic alignment of individual peaks in large high-resolution spectral data set. J. Magn. Reson. 170, 329–355 (2004)
12. Viant, M.R., Rosenblum, E.S., Tjeerdema, R.S.: NMR-Based Metabolomics: A Powerful approach for Characterizing the Effects of Environmental Stressors on Organism Health. Environ. Sci. Technol. 37, 4982–4989 (2003)
13. Voß, S., Fink, A.: Efficient Meta-heuristics Approaches for Ring Load Balancing. In: Proceedings of the 9th International Conference on Telecommunication Systems, pp. 243–250. Southern Methodist University, Dallas (2001)
14. Woodruff, D.L., Zemel, E.: Hashing Vectors for Tabu Search. Annals of OR 41, 123–137 (1993)
15. Xi, Y., de Ropp, J.S., Viant, M., Yu, P.: Automated Screening for Metabolites in Complex Mixtures using 2D COSY NMR Spectroscopy. Metabolomics 2, 221–233 (2006)

Appendix: Experimental Results

By running 20 amino acids against two samples, we obtain 40 runs of each algorithm. Table 1 summarizes the first set of results. The column labelled "Avg. Obj. Val." gives the average over the 40 runs of the objective function value found during each. The column "Avg. CPU (sec)" gives the average execution time. Although this paper is concerned with the optimization problem, it is useful to examine the column labelled "Over 100" because this gives the number of amino acids that cannot be ruled out (hence avoiding a Type I error).

Table 1. Averages for 40 Alignments by Simple Heuristics: 20 amino acids against two abalone tissue samples

Algorithm	Avg. Obj. Val.	Avg. CPU (sec)	Over 100
None	45.2		6
G1	49.2	0.9	8
G2	64.4	2.8	8
D1	61.8	0.7	8
D2	63.0	1.5	9

Since SA, TS, and HY are stochastic algorithms, we replicated them using 10 different seeds for the pseudo-random stream. The results with a wide variety of parameter combinations of each algorithm are given in tables that are available from the authors. Table 2 gives some of the results for the hybrid algorithm with the given neighborhood.

The run with the SA parameters 16 and 0.95 was not replicated because of the excessive run time. These parameter values are recommended by Johnson et al., which we used to verify that SA is not as effective as HY even when given a very long run time. The hybrid algorithm, HY was always run with a tabu tenure of 7 and the value of SizeFactor is always 1, so the parameters given are the aspiration method, the initial value of TabuStop and the SizeFactor.

Table 2. Replicated Experiments: Statistics Concerning the Averages of 10 Different Random Seeds for 40 Alignments

	Objective Val.		CPU (sec)	
Algorithm	Avg.	Std. Dev. of Avgs.	Avg.	Std. Dev. of Avgs.
HY A2 32 0.5	68.7	0.00	8.53	0.3
HY A1 32 0.6	68.7	0.00	9.45	0.3
HY A1 32 0.7	68.6	0.31	10.70	0.4
HY A1 64 0.5	68.7	0.00	17.47	0.2
HY A1 64 0.6	68.7	0.00	19.73	0.4
HY A1 64 0.7	68.7	0.00	22.28	0.6
HY A1 128 0.5	69.2	0.41	35.55	0.5
HY A1 128 0.6	69.4	0.35	39.50	1.0
HY A1 128 0.7	69.5	0.26	47.08	1.0
HY A1 128 0.8	69.6	0.24	56.10	1.9
HY A2 32 0.5	65.7	0.71	4.57	0.1
HY A2 32 0.6	66.0	0.00	4.53	0.1
HY A2 32 0.7	66.0	0.00	4.55	0.2
HY A2 64 0.5	67.6	0.63	6.76	0.2
HY A2 64 0.6	67.7	0.48	7.19	0.2
HY A2 64 0.7	68.0	0.00	7.81	0.2
HY A2 128 0.5	68.7	0.00	10.94	0.2
HY A2 128 0.6	68.6	0.32	11.61	0.2
HY A2 128 0.7	68.7	0.00	12.72	0.4
HY A2 256 0.5	68.7	0.00	18.82	0.7
HY A2 256 0.6	68.7	0.00	19.73	0.4
HY A2 256 0.7	68.7	0.00	21.71	0.7
HY A2 512 0.5	68.7	0.00	32.25	0.4
HY A2 512 0.6	68.7	0.00	36.71	1.1
HY A2 512 0.7	68.7	0.00	40.16	0.3
HY A1 32 0.8	68.7	0.00	12.22	0.7
HY A1 64 0.8	68.7	0.00	25.85	0.5
HY A2 64 0.8	68.0	0.00	7.76	0.2
HY A2 128 0.8	68.7	0.00	13.40	0.6
HY A1 256 0.7	69.6	0.21	85.31	1.1
HY A1 256 0.8	69.7	0.001	112.34	2.0

A New Succinct Representation of RMQ-Information and Improvements in the Enhanced Suffix Array*

Johannes Fischer and Volker Heun

Inst. für Informatik, Ludwig-Maximilians-Universität München
Amalienstr. 17, D-80333 München
{Johannes.Fischer,Volker.Heun}@bio.ifi.lmu.de

Abstract. The Range-Minimum-Query-Problem is to preprocess an array of length n in $O(n)$ time such that all subsequent queries asking for the position of a minimal element between two specified indices can be obtained in constant time. This problem was first solved by Berkman and Vishkin [1], and Sadakane [2] gave the first *succinct* data structure that uses $4n + o(n)$ bits of additional space. In practice, this method has several drawbacks: it needs $O(n \log n)$ bits of intermediate space when constructing the data structure, and it builds on previous results on succinct data structures. We overcome these problems by giving the first algorithm that *never* uses more than $2n + o(n)$ bits, and does not rely on rank- and select-queries or other succinct data structures. We stress the importance of this result by simplifying and reducing the space consumption of the Enhanced Suffix Array [3], while retaining its capability of simulating top-down-traversals of the suffix tree, used, e.g., to locate all *occ* positions of a pattern p in a text in optimal $O(|p| + occ)$ time (assuming constant alphabet size). We further prove a lower bound of $2n - o(n)$ bits, which makes our algorithm asymptotically optimal.

1 Introduction

Given an array A of n real numbers or other elements from a totally ordered set, a natural question is to ask for the position of a minimal element between two specified indices l and r. Queries of this form are known under the name of *range minimum queries* (RMQ), and the result is denoted by $\text{RMQ}_A(l, r)$. There are several variants of the problem, the most prominent being the one where the array is static and known in advance, which is the issue of this article. It has been shown by Berkman and Vishkin [1] that a preprocessing in $O(n)$ time is sufficient to answer all RMQs in $O(1)$ time. This algorithm has been rediscovered and simplified by Bender and Farach-Colton [4]. The main motivation of both papers is the strong connection between RMQs and a different fundamental algorithmic problem in trees, namely the task of computing the *lowest common ancestor* (LCA) of two specified nodes in constant time, first noted by Gabow et al. [5]. They showed

* This work was partially funded by the German Research Foundation (DFG).

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 459–470, 2007.

that LCA-queries on a tree T correspond to a *restricted* version of RMQ on an array that is obtained by writing down the heights of the nodes visited during an Euler-Tour through T (the details will be explained later). The fact that the general RMQ-problem can be reduced to an instance of LCA, which is in turn reduced to the restricted version of RMQ, is the basis for their RMQ-algorithms.

A drawback of these methods is that, because they employ arrays of size n that store numbers up to the computer's word size, their space consumption is $O(n \log n)$ bits. It has been noted that in many cases this is far from optimal, and a flood of algorithms under the term *succinct* (meaning that their space complexity is $O(n)$ *bits* instead of *words*) has been developed, starting with Jacobson's succinct representation for labeled trees [6].

The only known result on succinct data structures for the RMQ-problem is due to Sadakane [2]. His solution requires $4n+o(n)$ bits of space and makes heavy use of succinct data structures for rank- and select-queries on binary sequences (see [7]), and of succinct data structures for related problems [8]. Another drawback is that the solution in [2] requires the intermediate construction of several labeled trees, which use $O(n \log n)$ bits of space in the worst case.

1.1 Contributions of Our Work

We present a new succinct data structure for the RMQ-problem that uses $2n + o(n)$ bits of extra space and answers range minimum queries in $O(1)$ time. It is a practicable and direct method, meaning that (1) at each stage of its construction it never requires more space than the final data structure, (2) it does not rely on any other results on succinct data structures, and (3) it is easy to implement.

As a direct application of our new storage scheme for RMQ, we show that it leads to simplifications and improvements in the Enhanced Suffix Array (ESA) [3], a collection of arrays which can be used as a space-conscious alternative to the suffix tree, e.g., for locating all occ occurrences of a pattern p in a text in optimal $O(|p| + occ)$ time. Note that despite the significant progress that has been made in the field of compressed indexes [7,9], none of these indexes achieves this time bound, even if the size $|\Sigma|$ of the alphabet is assumed to be constant. So the ESA is still the method of choice. We show that one of the arrays from the ESA (previously occupying $n \log n$ bits) can be replaced by our storage scheme for RMQ, thus reducing its space to $2n + o(n)$ bits. We further emphasize the practicability of our method by performing tests on realistically-sized input arrays which show that our method uses less space than the most space-conscious *non*-succinct data structure [10], and that the query time is competitive.

Finally, a lower bound of $2n - o(n)$ bits is shown, making our data structure asymptotically optimal.

1.2 Applications of RMQ

We briefly sketch the most important applications of RMQ.

Computing Lowest Common Ancestors in Trees. For a static tree T to be preprocessed, $\text{LCA}_T(v, w)$ returns the deepest node in T that is an ancestor of both v and w. It has been noted by Gabow et al. [5] that this problem can be reduced to RMQ as follows: store the heights of the nodes in an array H in the order in which they are visited during an in-order tree traversal of T. Also, in $I[j]$ remember the node from which the height $H[j]$ has come. Finally, let R be the inverse array of I, i.e., $I[R[j]] = j$. Then $\text{LCA}_T(v, w)$ is given by $I[\text{RMQ}_H(R[v], R[w])]$. This is simply because $\text{LCA}_T(v, w)$ is the shallowest node visited between v and w during the in-order tree traversal.

Computing Longest Common Extensions of Suffixes. This problem has numerous applications in approximate pattern matching and is defined for a static string t of size n: given two indices i and j, $\text{LCE}_t(i, j)$ returns the length of the longest common prefix of t's suffixes starting at position i and j; i.e., $\text{LCE}_t(i, j) = \max\{k : t_{i,\ldots,i+k-1} = t_{j,\ldots,j+k-1}\}$. It is well-known that $\text{LCE}_t(i, j)$ is given $\mathsf{LCP}[\text{RMQ}_{\mathsf{LCP}}(\mathsf{SA}^{-1}[i] + 1, \mathsf{SA}^{-1}[j])]$, where LCP is the *LCP-array* [11] for t.

Document Retrieval Queries. The setting of document retrieval problems is as follows: For a static collection of n text documents, on-line queries like "return all d documents containing pattern p" are posed to the system. Muthukrishnan [12] gave elegant algorithms that solve this and related tasks in optimal $O(|p|+d)$ time. The idea behind these algorithms is to "chain" suffixes from the same document and use RMQs to ensure that each document containing p is visited at most twice. Sadakane [2] continued this line of research towards succinctness, again using RMQs.

Maximum-Sum Segment Queries. Given a static array A of n real numbers, on-line queries of the form "return the sub-interval of $[l, r]$ with the highest sum" are to be answered; i.e., $\text{MSSQ}(l, r)$ returns the index pair (x, y) such that $(x, y) = \arg\max_{l \le x \le y \le r} \sum_{i=x}^{y} A[i]$. This problem and extensions thereof have very elegant optimal solutions based on RMQs due to Chen and Chao [13]. The fundamental connection between RMQ and MSSQ can be seen as follows: compute an array of prefix sums $C[i] = \sum_{k=0}^{i} A[k]$ and prepare it for range minimum and maximum queries. Then if $C[x]$ is the minimum and $C[y]$ the maximum of all $C[i]$ in $[l - 1, r]$ and $x < y$, then $(x + 1, y)$ is the maximum-sum segment in $[l, r]$. The more complicated case where $x > y$ is also broken down to RMQs.

2 Definitions and Previous Results

The *Range Minimum Query* (RMQ) problem is formally defined as follows: given an array $A[0, n - 1]$ of elements from a totally ordered set (with order relation "$\le$"), $\text{RMQ}_A(l, r)$ returns the index of a smallest element in $A[l, r]$, i.e., $\text{RMQ}_A(l, r) = \arg\min_{k \in \{l,\ldots,r\}}\{A[k]\}$. (The subscript A will be omitted if the context is clear.) The most naive algorithm for this problem searches the array from l to r each time a query is presented, resulting in a $\Theta(n)$ query time in the worst case. As mentioned in the introduction, we consider the variant where

A is first preprocessed in order to answer future queries faster. The following definition will be central for both our algorithm and that of [1].

Definition 1. *A* Cartesian Tree *of an array $A[l,r]$ is a binary tree $\mathcal{C}(A)$ whose root is a minimum element of A, labeled with the position i of this minimum. The left child of the root is the Cartesian Tree of $A[l,i-1]$ if $i > l$, otherwise it has no left child. The right child is defined analogously for $A[i+1,r]$.*

Note that $\mathcal{C}(A)$ is not necessarily unique if A contains equal elements. To overcome this problem, we impose a *strong* total order "$\prec$" on A by defining $A[i] \prec A[j]$ iff $A[i] < A[j]$, or $A[i] = A[j]$ and $i < j$. The effect of this definition is just to consider the 'first' occurrence of equal elements in A as being the 'smallest'. Defining a Cartesian Tree over A using the $\prec$-order gives a *unique* tree $\mathcal{C}^{\text{can}}(A)$, which we call the *Canonical Cartesian Tree*. Note also that this order results in unique answers for the RMQ-problem, because the minimum under "$\prec$" is unique. A linear-time algorithm for constructing $\mathcal{C}^{\text{can}}(A)$ is given in [4].

In the rest of this paper the space is analyzed in *bit*-complexity. For the sake of clarity we write $O(f(n) \cdot \log(g(n)))$ for the number of bits needed by a table, where $f(n)$ denotes the number of entries in the table, and $g(n)$ is their maximal size. For example, a normal integer array of size n which takes values up to n uses $O(n \cdot \log n)$ bits of space.

2.1 Berkman and Vishkin's Algorithm

This section describes the solution to the general RMQ-problem as a combination of the results obtained in [1,5,4]. We follow the simplified presentation from [4].

$\pm$1RMQ is a special case of the RMQ-problem, where consecutive array elements differ by exactly 1. The solution starts by reducing RMQ to $\pm$1RMQ as follows: given an array $A[0,n-1]$ to be preprocessed for RMQ, build $\mathcal{C}^{\text{can}}(A)$. Then perform an Euler Tour in this tree, storing the labels of the visited nodes in an array $E[0,2n-2]$, and their respective heights in $H[0,2n-2]$. Further, store the position of the first occurrence of $A[i]$ in the Euler Tour in a representative array $R[0,n-1]$. The Cartesian Tree is not needed anymore once the arrays E, H and R are filled, and can thus be deleted. The paper then shows that $\text{RMQ}_A(l,r) = E[\pm 1\text{RMQ}_H(R[l],R[r])]$. Note in particular the doubling of the input when going from A to H; i.e., H has $n' := 2n-1$ elements.

To solve $\pm$1RMQ on H, partition H into *blocks* of size $\frac{\log n'}{2}$.[1] Define two arrays A' and B of size $\frac{2n'}{\log n'}$, where $A'[i]$ stores the minimum of the ith block in H, and $B[i]$ stores the position of this minimum in H.[2] We now want to preprocess A' such that out-of-block queries (i.e., queries that span over several blocks in H) can be answered in $O(1)$. The idea is to precompute all RMQs whose length is a power of two. For every $0 \le i < 2n'/\log n'$ and every $1 \le j \le \log(2n'/\log n')$ compute the position of the minimum in the sub-array $A'[i,i+2^j-1]$ and store the result in $M[i][j]$. Table M occupies

[1] For a simpler presentation we often omit floors and ceilings from now on.
[2] Array A' is just conceptual because $A'[i]$ is given by $H[B[i]]$.

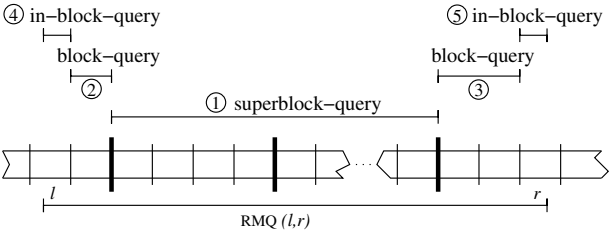


Fig. 1. How a range-minimum query $\text{RMQ}(l, r)$ can be decomposed into at most five different sub-queries. Thick lines denote the boundaries between superblocks, thin lines denote the boundaries between blocks.

$O(\frac{2n'}{\log n'} \log \frac{2n'}{\log n'} \cdot \log \frac{2n'}{\log n'}) = O(n \cdot \log n)$ bits of space and can be filled in $O(n)$ time by using the formula $M[i][j] = \arg\min_{k \in \{M[i][j-1], M[i+2^{j-1}][j-1]\}} \{A'[k]\}$. To answer $\text{RMQ}_{A'}(i, j)$, select two *overlapping* blocks that exactly cover the interval $[i, j]$, and return the position where the overall minimum occurs. Precisely, let $l = \lfloor \log(j - i) \rfloor$. Then $\text{RMQ}(i, j) = \arg\min_{k \in \{M[i][l], M[j-2^l+1][l]\}} \{A'[k]\}$.

It remains to show how in-block-queries are handled. This is done with the so-called Four-Russians-Trick, where one precomputes the answers to all possible queries when the number of possible instances is sufficiently small. The authors of [4] noted that due to the ± 1-property there are only $O(\sqrt{n'})$ blocks to be precomputed: we can virtually subtract the initial value of a block from each element without changing the answers to the RMQs; this enables us to describe a block by a ± 1-vector of length $2^{1/2 \log n' - 1}$. For each such block precompute all $\frac{1}{2} \frac{\log n'}{2} (\frac{\log n'}{2} + 1) = O(\log^2 n')$ possible RMQs and store them in a table $P[1, 2^{1/2 \log n' - 1}][1, \frac{\log n'}{4} (\frac{\log n'}{2} + 1)]$ with a total size of $O(\sqrt{n'} \log^2 n' \cdot \log \frac{\log n'}{2}) = o(n)$ bits. To index table P, precompute the *type* of each block and store it in array $T[1, \frac{2n'}{\log n'}]$. The block type is simply the binary number obtained by comparing subsequent elements in the block, writing a 0 at position i if $H[i+1] = H[i] + 1$ and 1 otherwise. Because tables M, E and R are of size $O(n)$, the total space needed is $O(n \cdot \log n)$ bits.

Now, to answer $\text{RMQ}(l, r)$, if l and r occur in different blocks, compute (1) the minimum from l to the end of l's block using arrays T and P, (2) the minimum of all blocks between l's and r's block using the precomputed queries on A' stored in table M, and (3) the minimum from the beginning of r's block to r, again using T and P. Finally, return the position where the overall minimum occurs, possibly employing B. If l and r occur in the same block, just answer an in-block-query from l to r. In both cases, the time needed for answering the query is constant.

3 Our New Algorithm

This section describes our new algorithm for the RMQ-problem. The array A to be preprocessed is (conceptually) divided into superblocks $B'_1, \ldots, B'_{n/s'}$ of

size $s' := \log^{2+\epsilon} n$, where B'_i spans from $A[(i-1)s']$ to $A[is'-1]$. Here, ϵ is an arbitrary constant greater than 0. Likewise, A is divided into (conceptual) blocks $B_1, \ldots, B_{n/s}$ of size $s := \log n/(2+\delta)$. Again, $\delta > 0$ is a constant. For the sake of simplicity we assume that s' is a multiple of s. We will preprocess long queries by a two-level step due to Sadakane [8], and short queries will be precomputed by a combination of the Four-Russians-Trick (as presented in [15]) with the method from [10]. The general idea is that a query from l to r can be divided into at most five sub-queries (see also Fig. 1): one *superblock-query* that spans several superblocks, two *block-queries* that span the blocks to the left and right of the superblock-query, and two *in-block-queries* to the left and right of the block-queries. From now on, we assume that the $\prec$-relation is used for answering RMQs, such that the answers become unique.

3.1 A Succinct Data Structure for Handling Long Queries

We first wish to precompute the answers to all RMQs that span over at least one superblock. Define a table $M'[0, n/s'-1][0, \log(n/s')]$. $M'[i][j]$ stores the position of the minimum in the sub-array $A[is', (i+2^j)s'-1]$. As in Sect. 2.1, $M'[i][0]$ can be filled by a linear pass over the array, and for $j > 0$ we use a dynamic programming approach by setting $M'[i][j] = \arg\min_{k \in \{M'[i][j-1], M'[i+2^{j-1}][j-1]\}} \{A[k]\}$.

In the same manner we precompute the answers to all RMQs that span over at least one block, but *not* over a superblock. These answers are stored in a table $M[0, n/s-1][0, \log(s'/s)]$, where $M[i][j]$ stores the minimum of $A[is, (i+2^j)s-1]$. Again, dynamic programming can be used to fill table M in optimal time.

3.2 A Succinct Data Structure for Handling Short Queries

We now show how to store all necessary information for answering in-block-queries in a table P.

Theorem 1 ([15]). *Let A and B be two arrays, both of size s. Then $\text{RMQ}_A(l,r) = \text{RMQ}_B(l,r)$ for all $0 \le l \le r < s$ if and only if $\mathcal{C}^{can}(A) = \mathcal{C}^{can}(B)$.* □

It is well known that the number of binary trees with s nodes is C_s, where C_s is the s'th *Catalan Number* defined by $C_s = \frac{1}{s+1}\binom{2s}{s} = 4^s/(\sqrt{\pi}s^{3/2})(1 + O(s^{-1}))$. Because the Cartesian tree is a binary tree, this means that table P does not have to store the in-block-queries for all n/s *occurring* blocks, but only for $4^s/(\sqrt{\pi}s^{3/2})(1+O(s^{-1}))$ *possible* blocks. We use the method described in [10] to represent the answers to all RMQ-queries inside one block.

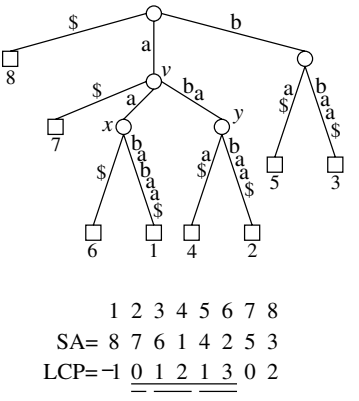
Computing the Block Types. In order to index table P, it remains to show how to fill array T; i.e., how to compute the types of the blocks B_i occurring in A in linear time. Thm. 1 implies that there are only C_s different types of arrays of size s, so we are looking for a surjection

$$t : \mathcal{A}_s \to \{0, \ldots, C_s - 1\}, \text{ and } t(B_i) = t(B_j) \text{ iff } \mathcal{C}^{\text{can}}(B_i) = \mathcal{C}^{\text{can}}(B_j)\ , \qquad (1)$$

Input: block B_j
Output: $t(B_j)$, the type of B_j

```
1  let R[0, s − 1] be an array
2  R[0] ← −∞
3  q ← s, N ← 0
4  for i ← 1, . . . , s do
5      while R[q + i − s − 1] > B_j[i − 1]
       do
6          N ← N + C_{(s−i)q}
7          q ← q − 1
8      end
9      R[q + i − s] ← B_j[i − 1]
10 end
11 return N
```

Fig. 2. An algorithm to compute the type of a block



	1	2	3	4	5	6	7	8
SA=	8	7	6	1	4	2	5	3
LCP=	−1	0	1	2	1	3	0	2

Fig. 3. The suffix tree (top) and the suffix- and LCP-array (bottom) for string $t = aababaa\$$

where $\mathcal{A}_s$ is the set of arrays of size s. The reason for requiring that B_i and B_j have the same Canonical Cartesian Tree is given by Thm. 1 which tells us that in such a case both blocks share the same RMQs. The algorithm in Fig. 2 shows how to compute the block type directly. It makes use of the so-called *ballot numbers* C_{pq}, defined by

$$C_{00} = 1, C_{pq} = C_{p(q-1)} + C_{(p-1)q}, \text{ if } 0 \le p \le q \ne 0, \text{ and } C_{pq} = 0 \text{ otherwise.} \quad (2)$$

The basic idea of the algorithm in Fig. 2 is that it simulates a walk in a certain graph, as explained in [15].

Theorem 2 ([15]). *The algorithm in Fig. 2 computes a function satisfying the conditions given in (1) in $O(s)$ time.* □

3.3 Space Analysis

Table M' has dimensions $n/s' \times \log(n/s') = n/\log^{2+\epsilon} n \times \log(n/\log^{2+\epsilon} n)$ and stores values up to n; the total number of bits needed for M' is therefore $n/\log^{2+\epsilon} n \times \log(n/\log^{2+\epsilon} n) \cdot \log n = o(n)$. Table M has dimensions $n/s \times \log(s'/s) = (2+\delta)n/\log n \times \log((2+\delta)\log^{1+\epsilon} n)$. If we just store the offsets of the minima then the values do not become greater than s'; the total number of bits needed for M is therefore $O(n/\log n \times \log(\log^{1+\epsilon} n) \cdot \log(\log^{2+\epsilon} n)) = o(n)$. To store the type of each block, array T has length $n/s = (2+\delta)n/\log n$, and because of Thm. 1 the numbers do not get bigger than $O(4^s/s^{3/2})$. This means that the number of bits to encode T is

$$\frac{n}{s} \cdot \log(O(4^s/s^{3/2})) = \frac{n}{s}(2s - O(\log s)) = 2n - O(n/\log n \log\log n) = 2n - o(n)\ .$$

Finally, by Sect. 3.2, and as [10] allows to store all queries inside of one block in $O(s \cdot s)$ bits, table P can be stored in

$$O\left(\frac{4^s}{s^{3/2}} s \cdot s\right) = O\left(n^{2/(2+\delta)}\sqrt{\log n}\right) = O\left(n^{1-\frac{1}{2/\delta+1}}\sqrt{\log n}\right) = o(n/\log n)$$

bits. Thus, the total space needed is $2n + o(n)$ bits. It is interesting that the leading term $(2n)$ comes from table T, i.e., from remembering the type of all blocks occurring in A. One can wonder if this is really necessary. The following theorem says that, asymptotically, one cannot do better than this, provided that the array is *only* used for minimum evaluations. To model this situation, we introduce the so-called *min-probe model*, where we only count evaluations of $\arg\min\{A[i], A[j]\}$, and all other computations and accesses to additional data structures (but *not* to A) are free (note the analogy to the cell-probe model [16]).

Theorem 3. *For an array A of size n one needs at least $2n - o(n)$ additional bits to answer $\text{RMQ}_A(l, r)$ in $O(1)$ for all $0 \le l \le r < n$ in the min-probe model.*

Proof. Let $\mathcal{D}_A$ be the additional data structure. To evaluate $\text{RMQ}_A(l, r)$ in $O(1)$, the algorithm can make $O(k)$ argmin-evaluations (constant k). The algorithm's decision on which $i \in [l, r]$ is returned as the minimum is based on the outcome of the $K := 2^k$ possible outcomes of these argmin-evaluations, and possibly other computations in $\mathcal{D}_A$. Now suppose there are less than $C_n/(K+1)$ different such $\mathcal{D}_A$'s. Then there exists a set $\{A_0, \ldots, A_K\}$ of $K+1$ arrays of length n with $\mathcal{D}_{A_i} = \mathcal{D}_{A_j}$ for all i, j, but with pairwise different Cartesian Trees. Because the algorithm can return different answers for at most K of these arrays, for at least two of them (say A_x and A_y) it gives the same answer to $\text{RMQ}(l, r)$ for all l, r. But A_x and A_y have different Cartesian trees, so there must be at least one pair of indices l', r' for which $\text{RMQ}_{A_x}(l', r') \neq \text{RMQ}_{A_y}(l', r')$. Contradiction. So there must be at least $C_n/(K+1)$ different choices for $\mathcal{D}_A$; thus the space needed to represent $\mathcal{D}_A$ is at least $\log(C_n/(K+1)) = 2n - o(n) - O(1)$ bits. □

4 Improvements in the Enhanced Suffix Array

Suffix trees are a very powerful tool for many tasks in pattern matching. Because of their large space consumption, a recent trend in text indexing tries to replace them by adequate array-based data structures. Kasai et al. [17] showed how algorithms based on a bottom-up traversal of a suffix tree can be simulated by a parallel scan of the suffix- and LCP-array. The Enhanced Suffix Array (ESA) [3] takes this approach one step further by also being capable of simulating top-down traversals of the suffix tree. This, however, requires the addition of another array to the suffix- and LCP-array, the so-called *child-table*. Essentially, the child table captures the information on how to move from an internal node to its children. This table requires $O(n)$ words (or $O(n \cdot \log n)$ bits). We show in this section that the RMQ-information on the LCP-array can be used as an alternative representation of the child-table, thus reducing the space requirement

to $O(n/\log n)$ words (precisely, to $2n + o(n)$ bits). Note that our representation is not only less space-consuming than [3], but also much simpler. Throughout this section, t is a text of length n.

4.1 Enhanced Suffix Arrays

In its simplest form, the ESA consists of the suffix- and LCP-array for t. The basic idea of the ESA is that internal nodes of the suffix tree correspond to certain intervals (so-called ℓ-intervals) in the LCP-array (recall the definition of SA and LCP in Sect. 1.2):

Theorem 4 ([3]). *Let T, SA, LCP be t's suffix tree, suffix- and LCP-array, respectively. Then the following is equivalent:*

1. *There is an internal node in T representing a sub-word ϕ of t.*
2. *There exist $1 \leq l < r \leq n$ s.t. (a) $\mathsf{LCP}[l] < |\phi|$ and $\mathsf{LCP}[r+1] < |\phi|$, (b) $\mathsf{LCP}[i] \geq |\phi|$ for all $l < i \leq r$ and $\phi = t_{\mathsf{SA}[q]..\mathsf{SA}[q]+|\phi|-1}$, and (c) $\exists\ q \in \{l+1, \ldots, r\}$ with $\mathsf{LCP}[q] = |\phi|$.*

The pair of indices satisfying point 2 of the above theorem are said to form a $|\phi|$-interval $[l:r]$ (denoted as $|\phi|$-$[l:r]$), and each position q satisfying (c) is called a $|\phi|$-index. For example, in the tree in Fig. 3, node v corresponds to the 1-interval [2:6] in LCP and has 1-indices 3 and 5.

Let ℓ-$[l:r]$ be any such interval, corresponding to node v in T. Then if there exists an $\ell' > \ell$ such that there is an ℓ'-interval $[l':r']$ contained in $[l:r]$, and no super-interval of $[l':r']$ has this property, then ℓ'-$[l':r']$ corresponds to an internal child of v in T. E.g., in Fig. 3, the two child-intervals of 1-[2:6] representing internal nodes in T are 2-[3:4] and 3-[5:6], corresponding to nodes x and y. The connection between ℓ-indices and child-intervals is as follows [3, Lemma 6.1]:

Lemma 1. *Let $[l:r]$ be an ℓ-interval. If $i_1 < i_2 < \cdots < i_k$ are the ℓ-indices in ascending order, then the child intervals of $[l:r]$ are $[l:i_1-1], [i_1:i_2], \ldots, [i_k:r]$. (Singleton intervals are leaves!)*

With the help of Lemma 1 it is possible to simulate top-down traversals of the suffix tree: start with the interval 0-$[1:n]$ (representing the root), and at each interval calculate the child-intervals by enumerating their ℓ-indices. To find the ℓ-indices in constant time, the authors of [3] introduce a new array $C[1,n]$, the so-called child-table, occupying $n \log n$ bits of space.

4.2 An RMQ-based Representation of the Child-Table

The following lemma is the key to our new technique:

Lemma 2. *Let $[l:r]$ be an ℓ-interval. Then its ℓ-indices can be obtained in ascending order by $i_1 = \mathrm{RMQ}_{\mathsf{LCP}}(l+1, r)$, $i_2 = \mathrm{RMQ}_{\mathsf{LCP}}(i_1+1, r), \ldots$, as long as $\mathsf{LCP}[\mathrm{RMQ}_{\mathsf{LCP}}(i_k+1, r)] = \ell$.*

Input: pattern $p = p_{1..m}$ to be found in t
Output: interval of p in SA or negative answer

```
1  c ← 0, found ← true, l ← 1, r ← n
2  repeat
3      [l, r] ← getChild(l, r, p_{c+1})
4      if [l : r] = ∅ then return "not found"
5      if l = r then M ← m
6        else M ← min{LCP[RMQ_LCP(l + 1, r)], m}
7      found ← (p_{c+1..M−1} = t_{SA[l]+c..SA[l]+M−1})
8      c ← M
9  until l = r ∨ c = m ∨ found = false
10 if found then return [l : r]
11   else return "not found"
```

```
11 function getChild (l, r, a)
12 r_old ← r
13 r ← RMQ_LCP(l + 1, r_old)
14 ℓ ← LCP[r]
15 repeat
16     if t_{SA[l]+ℓ} = a then
17         return [l : r − 1]
18     end
19     l ← r
20     r ← RMQ_LCP(l + 1, r_old)
21 until l = r_old ∨ LCP[r] > ℓ
22 if t_{SA[l]+ℓ} = a then
23     return [l : r_old]
24 else return ∅
```

Fig. 4. How to locate a pattern of length m in a text in $O(m)$ time

Proof. Because of point 2(b) in Thm. 4, the LCP-values in $[l+1{:}r]$ cannot be less than ℓ. Thus any position in this interval with a minimal LCP-value must be an ℓ-index of $[l{:}r]$. On the other hand, if $\mathsf{LCP}[\mathrm{RMQ}_{\mathsf{LCP}}(i_k+1, r)] > \ell$ for some k, then there cannot be another ℓ-index in $[i_k+1{:}r]$. Because RMQ always yields the position of the *leftmost* minimum if this is not unique, we get the ℓ-indices in ascending order. □

With Lemma 1 this allows us to compute the child-intervals by preprocessing the LCP-array for RMQ. As an example, we can retrieve the 1-indices of 1-$[2{:}6]$ as $i_1 = \mathrm{RMQ}_{\mathsf{LCP}}(3, 6) = 3$ giving interval $[2{:}2]$ (corresponding to leaf 7 in Fig. 3), $i_2 = \mathrm{RMQ}_{\mathsf{LCP}}(4, 6) = 5$ giving $[3{:}4]$ (corresponding to node x). Because $\mathsf{LCP}[\mathrm{RMQ}_{\mathsf{LCP}}(6, 6)] = \mathsf{LCP}[6] = 3 > 1$, there are no more 1-indices to be found, so the last child-interval is $[5{:}6]$ (corresponding to y).

Theorem 5. *Any algorithm based on a top-down traversal of a suffix tree can be replaced by data structure using* $|\mathsf{SA}| + 4n + o(n)$ *bits without affecting its time bounds, where* $|\mathsf{SA}|$ *denotes the space consumed by the suffix array.*

Proof. With Lemmas 1 & 2, preparing the LCP-array for RMQ is enough for retrieving the child-intervals. Because of Thm. 3, this requires $2n + o(n)$ bits. [8] has shown that the LCP-array can be stored in $2n + o(n)$ bits as well, while retaining constant access to $\mathsf{LCP}[i]$. With Thm. 4, the claim follows. □

4.3 Application to Pattern Matching

For a given pattern p of length m, the task is to answer in $O(m)$ time whether p is a substring of t, and to locate all occ occurrences of p in $O(m+occ)$ time. With a plain suffix array these time bounds cannot be achieved (they are $O(m \log n)$ and $O(m \log n + occ)$, respectively). Note that the ESA is still the method of choice for this task, as all known compressed indexes [7,9] have asymptotic worse matching or locating times, even for the alphabet size $|\Sigma|$ being a constant.

The algorithm to locate a pattern p is shown in Fig. 4. The invariant of the algorithm is that *found* is `true` if $p_{1..c}$ occurs in t and has the interval $[l:r]$ in SA. In each step of the loop, method `getChild`(l, r, a) is used to find the subinterval of $[l:r]$ that stores the suffixes having $p_{1..c+1}$ as a prefix. This is done exactly as described in Sect. 4.2. Because $|\Sigma|$ is a constant, function `getChild` takes constant time. (Note that for large $|\Sigma| \in \omega(\log n)$ one would actually drop back to binary search, so even for large alphabets `getChild` never takes more than $O(\log n)$ time.) The if-statement in lines 5–6 distinguishes between internal nodes ($l > r$) and leaves ($l = r$). The actual pattern matching is done in line 7 of the algorithm. Because c is increased by at least 1 in each step of the loop, the running time of the whole algorithm is $O(m)$.

To retrieve all *occ* occurrences of p, get p's interval $[l:r]$ in SA by the method explained above, and then return the set of positions $\{\mathsf{SA}[l], \mathsf{SA}[l+1], \ldots, \mathsf{SA}[r]\}$. This takes $O(occ)$ additional time.

5 Implementation Details

We implemented the algorithm from Sect. 3 in C++ (available at the first author's home page). A good trade-off between time and space is to fix the block size s to 2^3 and the superblock size s' to 2^8, and to introduce an "intermediate" block-division of size $s'' = 2^4$. As the intermediate blocks consist of two blocks of size s, their RMQs can be answered with two look-ups to table P, and therefore do not have to be stored at all. The advantage of this intermediate layer is that it reduces the space of table M'. In total, our implementation uses $\frac{7}{8}n$ bytes.

We compared our algorithm with the most space-conscious algorithm for RMQ due to Alstrup et al. [10], which uses only $2n + o(n)$ additional *words* of space. We could not compare with the succinct method [2] because it is not yet publicly available. We performed some tests on random arrays of length up to $n = 2^{27} \approx 1.34 \times 10^6$ (i.e., the array uses space up to $2^{27} \times 4 = 2^{29} = 512$MB). We found that our method is not only much less space consuming than [10] (by a factor of ≈ 8), but also faster in constructing the index (by a factor of ≈ 2.5). This shows that the extra work from Sect. 3 pays off. The query time for short queries ($\log n/2$) is about the same for both methods (with a slight advantage for our method), whereas for long queries ($n/100$) our method is slowed down by only a factor of two, which can be coped with given the reduction in space.

6 Conclusions

We have presented a direct and easy-to-implement data structure for constant-time RMQ-retrieval that uses $2n + o(n)$ bits of additional space, which is asymptotically optimal. This led to direct improvements in the Enhanced Suffix Array. Tests confirmed the practical utility of our method. We finally note that our algorithm is also easy to implement on PRAMs (or real-world shared-memory machines), where with n/t processors the preprocessing runs in time $\Theta(t)$ if $t = \Omega(\log n)$, which is work-optimal.

References

1. Berkman, O., Vishkin, U.: Recursive star-tree parallel data structure. SIAM J. Comput. 22(2), 221–242 (1993)
2. Sadakane, K.: Space-efficient data structures for flexible text retrieval systems. In: Bose, P., Morin, P. (eds.) ISAAC 2002. LNCS, vol. 2518, pp. 14–24. Springer, Heidelberg (2002)
3. Abouelhoda, M.I., Kurtz, S., Ohlebusch, E.: Replacing suffix trees with enhanced suffix arrays. J. Discrete Algorithms 2(1), 53–86 (2004)
4. Bender, M.A., Farach-Colton, M., Pemmasani, G., Skiena, S., Sumazin, P.: Lowest common ancestors in trees and directed acyclic graphs. J. Algorithms 57(2), 75–94 (2005)
5. Gabow, H.N., Bentley, J.L., Tarjan, R.E.: Scaling and related techniques for geometry problems. In: Proc. of the ACM Symp. on Theory of Computing, pp. 135–143. ACM Press, New York (1984)
6. Jacobson, G.: Space-efficient static trees and graphs. In: Proc. FOCS, pp. 549–554. IEEE Computer Society Press, Los Alamitos (1989)
7. Navarro, G., Mäkinen, V.: Compressed full-text indexes. ACM Computing Surveys (to appear, 2007) Preliminary version available at `http://www.dcc.uchile.cl/~gnavarro/ps/acmcs06.ps.gz`
8. Sadakane, K.: Succinct representations of lcp information and improvements in the compressed suffix arrays. In: Proc. SODA, ACM/SIAM, pp. 225–237 (2002)
9. Sadakane, K.: Compressed suffix trees with full functionality. Theory of Computing Systems (to appear, 2007), Preliminary version available at `http://tcslab.csce.kyushu--u.ac.jp/~sada/papers/cst.ps`
10. Alstrup, S., Gavoille, C., Kaplan, H., Rauhe, T.: Nearest common ancestors: A survey and a new distributed algorithm. In: Proc. SPAA, pp. 258–264. ACM Press, New York (2002)
11. Manber, U., Myers, E.W.: Suffix arrays: A new method for on-line string searches. SIAM J. Comput. 22(5), 935–948 (1993)
12. Muthukrishnan, S.: Efficient algorithms for document retrieval problems. In: Proc. SODA, ACM/SIAM, pp. 657–666 (2002)
13. Chen, K.-Y., Chao, K.-M.: On the range maximum-sum segment query problem. In: Fleischer, R., Trippen, G. (eds.) ISAAC 2004. LNCS, vol. 3341, pp. 294–305. Springer, Heidelberg (2004)
14. Tarjan, R.E., Vishkin, U.: An efficient parallel biconnectivity algorithm. SIAM J. Comput. 14(4), 862–874 (1985)
15. Fischer, J., Heun, V.: Theoretical and practical improvements on the RMQ-problem, with applications to LCA and LCE. In: Lewenstein, M., Valiente, G. (eds.) CPM 2006. LNCS, vol. 4009, pp. 36–48. Springer, Heidelberg (2006)
16. Yao, A.C.-C.: Should tables be sorted? J. ACM 28(3), 615–628 (1981)
17. Kasai, T., Lee, G., Arimura, H., Arikawa, S., Park, K.: Linear-time longest-common-prefix computation in suffix arrays and its applications. In: Amir, A., Landau, G.M. (eds.) CPM 2001. LNCS, vol. 2089, pp. 181–192. Springer, Heidelberg (2001)

Lagrangian Relaxation and Cutting Planes for the Vertex Separator Problem*

Victor F. Cavalcante and Cid C. de Souza

Institute of Computing, State University of Campinas, C.P. 6176, 13084-970
Campinas, SP, Brazil
{victor.cavalcante,cid}@ic.unicamp.br

Abstract. In this paper we propose an algorithm for the vertex separator problem (VSP) based on Lagrangian Relaxation and on cutting planes derived from valid inequalities presented in [3]. The procedure is used as a preprocessing for the branch-and-cut (B&C) algorithm implemented for VSP in [12], aiming to generate an initial pool of cutting planes and a strong primal bound for the latter. Computational experiments show that the exact method obtained in that way outperforms the pure B&C algorithm recently introduced by de Souza and Balas in [12]. Besides, we show that the Lagrangian phase is a very effective heuristic for the VSP, often producing optimal solutions extremely fast.

1 Introduction

A *vertex separator* in an undirected graph is a subset of the vertices, whose removal disconnects the graph in at least two nonempty connected components. Recently, Balas and de Souza [3,12] studied the vertex separator problem (VSP) which can formally be stated as follows.

INSTANCE: a connected undirected graph $G = (V, E)$, with $|V| = n$, an integer $1 \leq b \leq n$ and a cost c_i associated with each vertex $i \in V$.

PROBLEM: find a partition of V into disjoint sets A, B, C, with A and B nonempty, such that *(i)* E contains no edge (i, j) with $i \in A$, $j \in B$, *(ii)* $\max\{|A|, |B|\} \leq b$, *(iii)* $\sum_{j \in C} c_j$ is minimized.

The sets A and B are called the *shores* of the separator C. A separator C that satisfies (i) but violates (ii) is termed *infeasible*; one that satisfies (i) and (ii) is *feasible*; and a separator that satisfies (i), (ii), (iii) is optimal. Unless otherwise specified, the term separator is used here to denote a feasible one. The VSP is $\mathcal{NP}$-hard and has widespread applicability in network connectivity. We refer to [3] for further discussion on applications, including one in Linear Algebra.

Based on the Integer Programming (IP) model and the strong valid inequalities introduced by Balas and de Souza, we propose an algorithm that combines Lagrangian relaxation with cutting plane techniques to solve the VSP. Our method belongs to a class of Lagrangian Relaxation algorithms where constraints of certain families of inequalities may only be explicitly dualized when

* Research supported by grants CNPq-307773/2004-3 and FAPESP-03/09925-5.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 471–482, 2007.

they become violated at some Lagrangian relaxation solution (see [5,6,8,10]). These so-called Relax-and-Cut (R&C) algorithms appear as a promising alternative approach to strengthen Lagrangian relaxation bounds.

Related Work. Relax-and-Cut algorithms are presented in several recent works in literature [5,6,7,8,9,10]. These algorithms use a dynamic inequality dualization scheme that renders viable the application of Lagrangian Relaxation to models with an exponential number of inequalities. Indeed, a similar approach for the traveling salesman problem [1] date from the early 80's.

The VSP described in this paper was first defined in [3] by Balas and de Souza were a polyhedral investigation of the VSP is conducted. They introduced several classes of strong valid inequalities for the polytope associated to the problem. The same authors reported extensive computational experiments with a branch-and-cut (B&C) algorithm based on these inequalities in [12]. A similar problem is discussed in [11] by Borndörfer *et al.* This problem can be considered a generalization of the VSP in the sense that the partitioning can be done in more than two sets of vertices. However, contrarily to the VSP, solutions where one of the shores remains empty are allowed.

Our contribution. The contribution of this paper is twofold and is related to the different usages we made of the R&C algorithm. First, we consider the performance of the B&C algorithm developed in [12] when the R&C is used as a preprocessing phase. We found out that the resulting method is a very powerful technique to tackle hard instances of VSP, outperforming the pure B&C algorithm in most of the benchmark instances. Thus, our algorithm can be viewed as the best exact method available so far for the VSP. Also, the R&C algorithm can be used as a heuristic for the VSP. We show that it produces solutions of very high quality for the problem, often optimal ones. Besides, our Lagrangian heuristic is much faster than the Linear Programming heuristic proposed in [12].

The paper is organized as follows. Section 2 presents the results in [3,12] that are relevant to our work, namely, an IP formulation of VSP and a class of valid inequalities used as cutting planes in the R&C algorithm. Section 3 briefly reviews the Lagrangian relaxation technique and the subgradient method (SM). A general description of the R&C framework is also given and the specific algorithm we developed for the VSP is described in section 4, including details of our Lagrangian heuristic for the problem. Section 5 reports on the computational results obtained for test instances gathered from the literature. Finally, in section 6, we draw some conclusions and discuss possible extensions of this study.

2 An IP Model for VSP and a Class of Valid Inequalities

We now describe the mixed IP model presented in [3,12] on which our Lagrangian relaxation is based. For every vertex $i \in V$, two binary variables are defined: $u_{i1} = 1$ if and only if $i \in A$ and $u_{i2} = 1$ if and only if $i \in B$. For $S \subseteq V$ and $k \in \{1, 2\}$, let $u_k(S)$ denote $\sum(u_{ik} : i \in S)$, and $u(S) = u_1(S) + u_2(S)$. An IP formulation for the VSP is given by

$$\max \sum_{i \in V} c_i(u_{i1} + u_{i2})$$

$$u_{i1} + u_{i2} \leq 1, \qquad \forall\, i \in V \tag{1}$$

$$u_{i1} + u_{j2} \leq 1,\ u_{j1} + u_{i2} \leq 1, \qquad \forall\, (i,j) \in E \tag{2}$$

$$u_1(V) \geq 1, \tag{3}$$

$$u_2(V) \leq b, \tag{4}$$

$$u_1(V) - u_2(V) \leq 0, \tag{5}$$

$$u_{i2} \geq 0,\ u_{i1} \in \{0,1\}, \qquad \forall\, i \in V. \tag{6}$$

Inequality (1) forces every vertex to belong to at most one shore. Inequalities (2) prohibits the extremities of an edge to be on distinct shores. Inequalities (3) to (5) limit the size of the shores and, at the same time, reduce the symmetry of the model by forcing the size of shore A to be bounded by that of shore B. As observed in [12], if the u_{i1} variables are integer for all $i \in V$, the integrality of the u_2 variables can be dropped from the formulation. Though this observation is not taken into account by our Lagrangian relaxation, it is relevant for IP solvers.

Balas and de Souza [3] call a valid inequality for VSP *symmetric* if, for all $j \in V$, the coefficients of the variables u_{j1} and u_{j2} in the inequality are the same. Besides, they show that vertex separators are intimately related to vertex dominators. A vertex dominator is a subset of vertices of the graph such that all the remaining vertices are adjacent to at least one of them. The dominator is said to be connected if the subgraph induced by its vertices is connected. Balas and de Souza then stated the following property: every separator and every connected dominator have at least one vertex in common. From this observation, they derived a class of symmetric inequalities associated with connected dominators, the so-called `CD` inequalities. If $S \subset V$ is a connected dominator, the `CD` inequality for S is given by

$$u(S) \leq |S| - 1. \tag{7}$$

Inequality (7) is clearly valid for the VSP polytope P, where $P = \text{conv}\{u \in \{0,1\}^{2|V|} : u \text{ satisfies (1)–(6)}\}$. It is non dominated only if S is *minimal* (with respect to vertex removal). However, necessary and sufficient conditions for `CD` inequalities to define facets are not known in general. But, in [12], they are shown to be very effective in computations.

According to [12], for unit costs, one can adapt the separation routine to search for more stringent `CD` inequalities. These inequalities are valid for all vectors $u \in P$ satisfying $u(V) \geq z_{LB} + 1$, but chop off several feasible solutions with smaller costs. Their usage preserves optimality and is conditioned to the existence of a lower bound z_{LB}. We call them *conditional cuts*, in an analogy to what is done for the set covering problem in [2]. For the VSP, these cuts are obtained computing $\alpha = max\{z_{LB} - b + 1, 1\}$ and searching minimal dominators

that cover at least $k = |V| - \alpha + 1$ vertices (k-dominators). Thus, given a lower bound z_{LB} for the optimum, the separation routine can be changed to identify minimal connected dominators that *cover* at least k vertices. Conditional cuts are used both in the B&C algorithm in [12] and in our implementation.

3 Relax-and-Cut (R&C) Algorithms

We now review the basics on Lagrangian relaxation and relax-and-cut algorithms that are relevant to us. Denote by X a subset of $\mathbb{B}^n = \{0,1\}^n$ and let

$$Z = \max \{cx : Ax \leq b,\ x \in X\} \tag{8}$$

be a formulation for a $\mathcal{NP}$-hard combinatorial optimization problem. In association with (8) one has $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$, where m and n are positive integral values representing, respectively, the number of constraints and the number of variables involved. Assuming that m is an exponential function of n, let Z' denote the formulation obtained after removing constraints $Ax \leq b$ from (8). Also, assume that Z' can be solved in polynomial or pseudo-polynomial time in the problem size.

A Lagrangian relaxation of (8) is obtained by bringing the term $\lambda(b - Ax)$ into the objective function of Z', where $\lambda \in \mathbb{R}^m_+$ is the corresponding vector of Lagrange multipliers. The resulting *Lagrangian relaxation Problem* (LRP) is

$$Z(\lambda) = \max \{cx + \lambda(b - Ax) :\ x \in X\} = \max \{(c - \lambda A)x + \lambda b :\ x \in X\}. \tag{9}$$

It is a known fact that $Z(\lambda) \geq Z$ and, therefore, the tightest possible upper bound on Z, attainable through $LRP(\lambda)$, is given by an optimal solution to the *Lagrangian dual problem* (LDP) $Z_D = \min_{\lambda \in \mathbb{R}^m_+} \{\max \{(c - \lambda A)x + \lambda b : x \in X\}\}$. In the literature, several methods exist to compute the LDP. Among these, due to its simplicity and the acceptable results it returns, the subgradient method (SM) is the most widely used [4]. A brief review of that method follows since the R&C algorithm we suggest here for the VSP is deeply based on SM.

SM is an iterative algorithm which solves a succession of LRPs like the one in (9). It starts with a feasible vector λ^0 of Lagrangian multipliers and, at iteration k, generates a new feasible vector λ^k of multipliers and an associated LRP.

At iteration k, let $\bar{x}^k$ be an optimal solution to (9) with cost $Z(\lambda^k)$ and let Z^k_{LB} be a known lower bound on (8). An associated subgradient vector (for the m relaxed constraints) is then computed as $g^k_i = (b_i - a_i \bar{x}^k),\ i = 1, 2, \ldots, m$. That vector is then used to update λ^k. To that order, a *step size* θ^k must be computed first. Typically, for a real valued parameter $\pi^k \in (0, 2]$, formula

$$\theta^k = \frac{\pi^k (Z(\lambda^k) - Z^k_{LB})}{\sum_{i=1}^m (g^k_i)^2} \tag{10}$$

is used [4]. Finally, once θ^k is obtained, λ^k is updated as

$$\lambda^{k+1}_i = \max \{0; \lambda^k_i - \theta^k g^k_i\},\ i = 1, 2, \ldots, m. \tag{11}$$

Notice that the straightforward use of formulas (10-11) may become troublesome when a *huge* number of dualized inequalities exist. An alternative is then to modify SM according to the R&C scheme discussed below.

In the literature two strategies to implement R&C algorithms are discussed. They differ, basically, on the moment at which the new inequalities are dualized. In a Delayed Relax-and-Cut (DRC), several executions of SM are made and the cuts found during one such execution are added only at the beginning of the next one. In a Non Delayed Relax-and-Cut (NDRC), typically a single SM run is done and cuts are dualized along the iterations as they are found. See [5,6,8,9,10] for details. In a comparison carried out in [9], NDRC performed better than DRC. Therefore, in our implementation, we adopt the NDRC strategy.

Clearly, if there are exponentially many inequalities in (8), the use of traditional Lagrangian relaxation becomes impracticable. Alternatively the R&C scheme proposes a dynamic strategy to dualize inequalities. In this process, one should be able to identify inequalities that are violated at $\bar{x}^k$. To do so, likewise polyhedral cutting-plane generation, a separation problem must be solved at every iteration of SM. Thus, one tries to find at least one inequality violated by the current LRP solution. The inequalities thus identified are candidates to be dualized. It is worth noting that separation problems arising in R&C algorithms may be easier than their polyhedral cutting-plane algorithm counterparts. That applies since LRP normally has integral valued solutions (cf. [10]).

4 A Relax-and-Cut Algorithm for the VSP

Different Lagrangian relaxations can be devised from the formulation given in Section 2. We decided to start with a simple one in which the constraint sets (1) and (2) are dualized by means of the vector multipliers $\lambda \in \mathbb{R}_+^{|V|}$, $\beta^1 \in \mathbb{R}_+^{|E|}$ and $\beta^2 \in \mathbb{R}_+^{|E|}$, respectively. The resulting LRP is given by

$$\begin{aligned} \mathrm{LRP}(\lambda, \beta^1, \beta^2) = \max \{ & \sum_{i \in V} (\bar{c}_{i1} u_{i1} + \bar{c}_{i2} u_{i2} + \lambda_i) + \sum_{\substack{(i,j) \in E \\ i<j}} (\beta^1_{i,j} + \beta^2_{i,j}) : \\ & u_{kl},\ k \in V,\ l = 1,2 \text{ satisfy (3)–(6)} \} \end{aligned} \tag{12}$$

where $\bar{c}_{k1} = c_k - \lambda_k - \sum_{\substack{(k,j) \in E \\ k<j}} \beta^1_{k,j} - \sum_{\substack{(i,k) \in E \\ i<k}} \beta^2_{i,k}$ and $\bar{c}_{k2} = c_k - \lambda_k - \sum_{\substack{(i,k) \in E \\ i<k}} \beta^1_{i,k} - \sum_{\substack{(k,j) \in E \\ k<j}} \beta^2_{k,j}$ are the Lagrangian costs of, respectively, u_{k1} and u_{k2}. Notice that (12) can be solved in $O(|V| \log |V|)$ time by sorting the variables according to their Lagrangian costs and after performing a few simple calculations.

On the other hand, the computation of good primal bounds is important for the computation of the step size (10) in the SM and to assess the duality gap along the iterations of the algorithm. To compute lower bounds for the VSP, we devise a simple greedy heuristic. Initially, the set L containing the vertices that are candidates to be part of the shores is built. This excludes the universal vertices (those which are adjacent to all other vertices) since, obviously, they must be in all separators. The heuristic chooses arbitrarily two nonadjacent

vertices of L and assigns them to different shores so that, in the end, they will not be empty. It proceeds by assigning vertices to shores, prioritizing the assignments corresponding to the variables with higher weighted Lagrangian costs. The weighting method, chosen after some experimentation, is implemented by multiplying the original cost of the variable associated to a vertex v by the degree of v ($\delta(v)$), as seen in step 5. All the assignments are made so as to maintain the compatibility and to respect the maximum size of the shores. As a final step, a local search subroutine is ran in an attempt to improve on the solution produced by the heuristic. These steps are summarized below.

Lagrangian heuristic $(G = (V, E), \overline{c})$
1. $L \leftarrow V \backslash \{$universal vertices in $G\}$;
2. $v \leftarrow \{$any vertex in $L\}$;
3. Initialize shore A: $A \leftarrow \{v\}$ and $L \leftarrow L \backslash \{v\}$;
4. Initialize shore B: $B \leftarrow \{$first vertex in $L \backslash \text{Adj}(v)\}$ and $L \leftarrow L \backslash B$;
5. `for` $k = 1, 2$ `do`:
   `for` all $i \in L$, compute $w_{u_{ik}} \leftarrow \bar{c}(u_{ik}) * \delta(i)$;
   Let S_k be the list of variables u_{ik} sorted nonincreasingly by $w_{u_{ik}}$;
6. `while` $|A| < b$ `or` $|B| < b$ `do`
   $f_1 \leftarrow \{$vertex corresponding to the first variable in $S_1\}$;
   $f_2 \leftarrow \{$vertex corresponding to the first variable in $S_2\}$;
   `if` $\bar{c}(u_{f_1,1}) > \bar{c}(u_{f_2,2})$ `then`
     $A \leftarrow A \cup \{f_1\}$; $S_1 \leftarrow S_1 \backslash \{u_{f_1,1}\}$;
     `for` all $j \in \text{Adj}(f_1)$ `do` $S_2 \leftarrow S_2 \backslash \{u_{j,2}\}$;
   `else`
     $B \leftarrow B \cup \{f_2\}$; $S_2 \leftarrow S_2 \backslash \{u_{f_2,2}\}$;
     `for` all $j \in \text{Adj}(f_2)$ `do` $S_1 \leftarrow S_1 \backslash \{u_{j,1}\}$;
   `if` $|A| = b$, $\bar{c}(u_{f_1,1}) \leftarrow -\infty$; /* avoids new vertices in A */
   `if` $|B| = b$, $\bar{c}(u_{f_2,2}) \leftarrow -\infty$; /* avoids new vertices in B */
7. Compute the separator: $C \leftarrow V \backslash \{A \cup B\}$
8. `Local Search`(G, A, B, C, c)
9. `return` (A, B, C)

We now turn to the improvement heuristic. The local search starts by enlarging the separator with as many vertices of the shores belonging to its adjacency as possible (steps 1–5). Then vertices are transferred from the separator back to the shores in step 6 in an arbitrary order. However, the choice of the destination shore is made so as to increase the chances of future moves from the separator to the shores. This is evaluated via the simple computations in steps 6.i to 6.m. The overall complexity of the Lagrangian heuristic, including the local search procedure, is $O(|V| \log |V| + |E|)$.

Local Search (G, A, B, C, c) *(initializations)*
1. Let A_C be the vertices in A that have neighbors in C;
2. Let B_C be the vertices in B that have neighbors in C;
3. `if` $A = A_C$ `then` $A_C \leftarrow A_C \backslash \{$arbitrarily chosen vertex of A$\}$;
4. `if` $B = B_C$ `then` $B_C \leftarrow B_C \backslash \{$arbitrarily chosen vertex of B$\}$;
5. $A' \leftarrow A \backslash A_C$; $B' \leftarrow B \backslash B_C$; $C' \leftarrow C \cup A_C \cup B_C$;

Local Search (G, A, B, C, c) *(main loop)*
6. `for` every vertex $v \in C'$ `do`:
6.a `if` $|A'| = b$ and $|B'| = b$ `then break`;
6.b `if` $|\text{Adj}(v) \cap A'| \neq \emptyset$ and $|\text{Adj}(v) \cap B'| \neq \emptyset$, `then continue`;
6.c $C' \leftarrow C' \backslash \{v\}$;
6.d `if` $\text{Adj}(v) \subset C'$ `then`
6.e `if` $|A'| = b$ `then` $B' \leftarrow B' \cup \{v\}$;
6.f `else`
6.g `if` $|B'| = b$ `then` $A' \leftarrow A' \cup \{v\}$;
6.h `else`
6.i $n_A \leftarrow 0$; $n_B \leftarrow 0$;
6.j `for` all $w \in \text{Adj}(v)$ `do`
6.k $n_A \leftarrow n_A + |\text{Adj}(w) \cap A|$; $n_B \leftarrow n_B + |\text{Adj}(w) \cap B|$;
6.l `if` $n_A > n_B$ `then` $A' \leftarrow A' \cup \{v\}$;
6.m `else` $B' \leftarrow B' \cup \{v\}$;
6.n `else`
6.o `if` $\text{Adj}(v) \subset A' \cup C'$ and $|A'| < b$ `then` $A' \leftarrow A' \cup \{v\}$;
6.p `else` /* $\text{Adj}(v) \subset B' \cup C'$ */
6.q `if` $|B'| < b$ `then` $B' \leftarrow B' \cup \{v\}$;
7. `if` $\sum_{i \in C} c_i > \sum_{i \in C'} c_i$ `then` $A \leftarrow A'$, $B \leftarrow B'$, $C \leftarrow C'$.

Another ingredient of our R&C algorithm is a fast separation routine that looks for violated CD inequalities at $\bar{u} = (\bar{u}_1, \bar{u}_2)$, the optimal solution of the current LRP in (12). A high level description of our procedure is given below where, for any $S \subset V$, $\text{Adj}(S)$ refers to the set of all vertices in $V \backslash S$ which are adjacent to at least one vertex in S. The routine starts by constructing the subgraph $G_{\bar{u}} = (W, F)$ of the input graph $G = (V, E)$ which is induced by the vertices $i \in V$ with $\bar{u}_{i1} + \bar{u}_{i2} \geq 1$. It is easy to see that, if W is a dominator and $G_{\bar{u}}$ is connected then the CD inequality associated to W is violated by $\bar{u}$. Unfortunately, the converse is not true in general but holds when constraints (1) are satisfied. Thus, our routine can be viewed as a heuristic. Step 5 of the algorithm tries to strengthen the inequality since the minimality of the dominator is a necessary condition for a CD inequality to be facet defining. It checks if the removal of a limited number of vertices preserves the connectedness of the graph induced by W and the dominance property. Clearly, in this way, the separation routine can be implemented to have $O(|V| + |E|)$ time complexity.

CD-Separation(G)
1. Construct $G_{\bar{u}} = (W, F)$;
2. Determine n_{CC}, the number of connected components of $G_{\bar{u}}$;
3. `if` $n_{CC} = 1$ `then` /* $G_{\bar{u}}$ is connected */
4. `if` $V \subseteq (W \cup \text{Adj}(W))$ `then` /* W is a dominator of V */
5. Turn W into a minimal CD;
6. `return` the CD inequality $u(W) \leq |W| - 1$;
7. `else return FAIL`; /* no new cut is returned for dualization */

In our R&C algorithm the separation procedure is called at each SM iteration. Since we implemented two greedy ways to obtain minimal CD inequalities, at most two cuts are produced at each SM iteration. Every new cut separated is stored in a pool and dualized in a Lagrangian fashion. The relaxation in (12) is then modified to incorporate this constraint. As a result, the term $\sum_{k=1}^{|pool|} \mu_k(|S_k| - 1 - u(S_k))$ is added to the cost function of (12), where $\mu \in \mathbb{R}_+^{|pool|}$ is the vector of multipliers of the CD inequalities that are currently dualized. The management of the cut pool is quite simple: a new inequality is inserted only if it is not identical to another inequality already in the pool or in the original formulation.

When the the R&C algorithm stops, the conditional cut generator (CCG) is executed. This procedure tries to transform every CD inequality in the pool into a more restrictive conditional cut, according to the discussion in Section 2. For that, z_{LB}, the best lower bound at the end of the R&C algorithm, is used.

In our implementation, the R&C algorithm and CCG form a preprocessing phase for the B&C described in [12]. The output from this phase is a set of, possibly conditional, CD cuts which are added *a priori* to the model given as input of B&C. Moreover, B&C is also given the values of z_{LB}, which may help to prune the enumeration earlier, and of $\alpha = z_{LB} - b + 1$ which, according to Section 2, is applied to limit the size of the smallest shore in equation (3). We denote by R&C&B&C, the algorithm resulting from this combination of R&C and B&C. Figure 1 depicts the flow diagram of R&C&B&C.

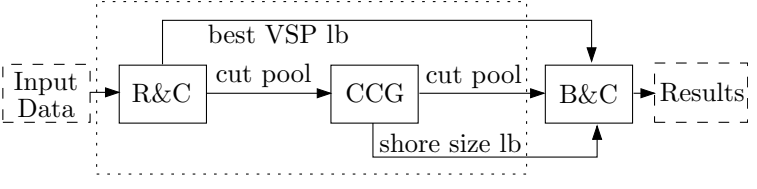


Fig. 1. Flow Diagram of R&C&B&C

5 Computational Results

In this section we report the computational tests carried out with the R&C algorithm implemented for the VSP. Tests were ran on a Pentium IV machine having 1GB of RAM and operating under Linux OS. The codes are written in C and C++, using resources of the Standard Template Library. XPRESS Optimizer 16.01.05 was used as the IP solver.

Our experiments were made on a subset of instances taken from [12]. Initially, from the more than 140 instances, we select those whose graph densities were higher than 40%. Among those, we kept only the ones which required more than a minute of CPU time to be solved by the branch-and-bound (B&B) algorithm of XPRESS (default configuration). We end up with 31 instances for our tests, all of which, with cost vector equals to the the sum vector. Instances from [12] can be downloaded from www.ic.unicamp.br/~cid/Problem-instances/VSP.html.

The motivation to select only high density graphs comes from : (a) these are the most difficult cases for standard B&B algorithms; (b) experimental results reported in [12] revealed some difficulties of the LP solver when high density cuts are added to the problem matrix, resulting on a severe degradation of the performance of cutting-plane algorithms and (c) in high density graphs, connected dominators tend to be small and, therefore, the corresponding CD inequalities are of low density. Thus, the use of cutting-plane algorithms is justifiable and likely to be more effective for high density graphs.

The following settings were used for the basic parameters of the subgradient algorithm: (a) in equation (10), π is initially set to 2 and multiplied by 0.5 each 100 consecutive iterations without improvement on the upper bound; (b) the Lagrangian heuristic is called at each SM iteration; (c) the local search heuristic is called only when the cost of the current solution differs from those of the ones previously found. Cost repetition is easily identified in our case since there are only $O(|V|)$ possible values for the cost function; (d) the algorithm stops when the limit of 2000 SM iterations is reached or when, $\pi^k \leq 0.00001$ in equation (10), what occurs first.

Computational experiments were performed aiming to assess three aspects of our approach: (i) the quality of the dual bounds produced by our R&C algorithm; (ii) the quality of the primal bounds produced by our Lagrangian heuristic; (iii) the effectiveness of using our approach as a preprocessing tool that provide an IP solver with a tighter formulation.

The latter aspect deserves some discussion. First, it should be noticed that the IP model from section 2 is rather weak. In the linear relaxation, setting all variables to 1/2 satisfies all the constraints for any sufficiently large value of b (which is the case for all instances in our dataset). This gives the worst dual bound one could come up with: n ! Thus, poor dual bounds are expected unless strong cuts are added to the formulation. Results reported in [12] show that CD inequalities fulfill this requirement. However, a drawback to use such inequalities comes from the fact that the corresponding separation problem is $\mathcal{NP}$-hard in general. The authors had then to resort to a heuristic procedure to perform the task. Their heuristic is of quadratic-time complexity, therefore, more expensive than our linear-time complexity heuristic used to separate integral points.

Moreover, inspecting the behavior of the B&C algorithm developed in [12], which we had access to, we noticed that a couple of CD inequalities needed to be separated and added to the model before good dual bounds are computed. Thus, it would be very helpful if we could quickly generate a set of initial CD cuts. That is precisely the role to be played by the R&C algorithm in the preprocessing phase. Besides this, the lower bound from the Lagrangian heuristic can be used as an incumbent to accelerate the pruning of the search tree.

Four algorithms were tested, namely: our R&C, the standard B&B from XPRESS, the B&C from [12] and the hybrid method, denoted by R&C&B&C, that uses R&C as a preprocessing for the B&C algorithm. In R&C&B&C, the B&C algorithm starts with the IP model strengthened by: (a) the CD cuts and (b) the incumbent solution returned by the preprocessing phase and (c) a

Table 1. Computational results for VSP instances

Instance					R&C					B&B		B&C			R&C&B&C	
name	d (%)	n	m	Opt	ub	lb	tb(s)	t(s)	Pool	t(s)	node	t(s)	node	tb(s)	t(s)	node
dim.miles1000	0.40	128	6560	110	122	109	<0.01	3.09	830	85.48	467	**15.51**	9	9.84	17.98	9
dim.DSJC125.9	0.90	125	14047	22	63	**22**	<0.01	4.4	1116	> 1800.00	72631	**890.96**	29809	13.85	1017.17	27935
mat.L125.fs_183_1	0.44	125	6909	98	135	96	0.05	2.04	16	225.19	2354	29.42	24	15.31	**27.88**	33
mat.bcsstk04	0.68	132	11968	84	91	**84**	<0.01	3.74	912	> 1800.00	17584	74.77	61	5.75	**11.23**	3
mat.arc130	0.93	130	15656	88	112	**88**	0.01	5.5	1138	**177.95**	127	225.36	83	12.77	237.41	83
mat.L120.cavity01	0.42	120	6064	99	117	**99**	0.13	3.24	345	239.83	2385	**9.15**	9	4.59	10.3	11
mat.L120.fidap021	0.43	120	6236	98	116	**98**	<0.01	2.13	349	98.07	776	12.56	15	3.22	**4.32**	11
mat.L120.rbs480a	0.46	120	6666	88	105	**88**	0.04	2.34	505	1193.02	14220	186.34	343	44.1	**82.95**	116
mat.L100.rbs480a	0.52	100	5200	73	82	**73**	<0.01	2.3	537	261.47	4831	19.28	57	0.30	**15.07**	55
mat.L120.e05r0000	0.59	120	8474	90	107	**90**	<0.01	2.82	539	246.59	1299	8.50	9	0.49	**6.94**	5
mat.L100.wm1	0.60	100	6012	74	102	73	0.4	1.63	49	71.14	974	19.67	21	11.17	**15.21**	21
mat.L120.fidap022	0.60	120	8734	84	93	**84**	<0.01	2.95	634	1008.47	10797	**32.66**	45	0.52	81.2	133
mat.L100.wm2	0.61	100	6178	76	103	**76**	<0.01	1.69	6	188.39	1925	12.74	14	8.66	**8.52**	14
mat.L100.fidapm02	0.62	100	6280	69	73	**69**	<0.01	2.04	663	426.09	6695	5.84	7	0.35	**4.27**	7
mat.L120.fidap001	0.63	120	9084	82	93	**82**	<0.01	3.76	464	> 1800.00	30274	**30.77**	29	15.6	54.81	5
mat.L100.e05r0000	0.64	100	6406	70	83	**70**	<0.01	1.81	353	209.06	3497	6.02	9	0.33	**4.06**	3
mat.L80.fidapm02	0.65	80	4196	53	**53**	**53**	<0.01	1.21	503	131.64	11323	3.48	11	1.88	**2.36**	5
mat.L120.fidapm02	0.65	120	9372	86	97	**86**	0.1	2.96	537	723.72	6602	27.73	21	5.60	**7.19**	5
mat.L100.fidap001	0.68	100	6858	64	81	**64**	<0.01	2.62	285	1185.9	55503	20.3	63	4.29	**9.47**	35
mat.L100.fidap022	0.68	100	6818	62	78	**62**	0.01	2.37	369	965.03	36353	43.63	131	4.96	**31.23**	107
mat.L80.fidap022	0.76	80	4886	41	75	**41**	<0.01	1.81	186	277.68	21621	11.91	135	0.24	**3.63**	137
mat.L100.fidap027	0.81	100	8128	69	85	**69**	<0.01	2.55	248	83.70	655	5.90	5	3.51	**4.14**	1
mat.L100.fidap002	0.82	100	8214	66	89	**66**	<0.01	2.52	184	203.21	3381	**4.56**	3	0.88	4.82	5
mat.L120.fidap002	0.82	120	11892	68	107	**68**	0.01	3.58	225	1439.09	12803	34.29	51	4.68	**8.74**	5
mat.L120.fidap027	0.85	120	12222	83	101	**83**	0.01	3.74	414	226.79	3758	12.36	7	7.79	**9.86**	5
miplib.l152lav.p	0.40	97	3829	61	98	54	<0.01	1.58	122	639.63	39489	73.24	292	50.87	**56.76**	245
miplib.lp4l.p	0.46	85	3373	50	82	46	0.01	1.37	94	411.68	34485	43.31	358	40.98	**36.48**	332
miplib.air03.p	0.61	124	9502	75	125	74	2.46	2.87	241	496.60	6373	158.67	124	68.64	**115.75**	113
miplib.misc03.p	0.63	96	5884	52	88	**52**	0.05	2.42	802	1387.42	55610	130.18	4411	8.73	**123.16**	3227
Σ										> 18002.84	458792	2149.11	36156		2012.91	32666

constraint stating that universal nodes in the input graph must belong to the separator. The results of our experiments are compiled in table 1. Five groups of columns are present in the table. The first describes the instance characteristics: name, density (d), the number of vertices (n) and edges (m) and the optimum (`Opt`). The four remaining groups give results of the different algorithms: `ub` (`lb`) for upper (lower) bounds, `t` for the computation time and `tb` for the time needed to find the best primal bound. Column head `Pool` for R&C gives the number of violated cuts added to the model before the execution of B&C.

We observe that, though the instances `dim.DSJC125.5` and `miplib.misc07` were initially selected to be part of our dataset, their results are not shown in table 1. This exclusion took place cause none of the tested algorithms computed `dim.DSJC125.5` and only R&C&B&C was able to solve `miplib.misc07` within the fixed time limit of 30 minutes for each run. Our approach needed 1658 seconds to prove optimality and the Lagrangian heuristic found its best solution with cost 115 (only one unit away from the optimum) in just 0.23 seconds !

We first analyse the dual bounds generated by R&C. If no cuts were used, the theoretical bound for the Lagrangian relaxation would be n as discussed earlier. Comparing column `ub` for R&C with n, we see that, by adding `CD` cuts, R&C is able to produce better bounds than the trivial one in 24 out of the 29 instances. This suggests that the preprocessing phase in R&C&B&C pays off as far as dual bounds are concerned. However, comparing columns `ub` for R&C and `Opt`, one sees that huge integrality gaps are still to be closed. Instance `mat.L80.fidapm02` was the only one that was solved to optimality by R&C.

On the primal side, excellent results were achieved. As seen in column `lb` of R&C, in 23 out of 29 cases our simple Lagrangian heuristic found an optimal solution. The average error of the heuristic was of only 0.94% and the maximum

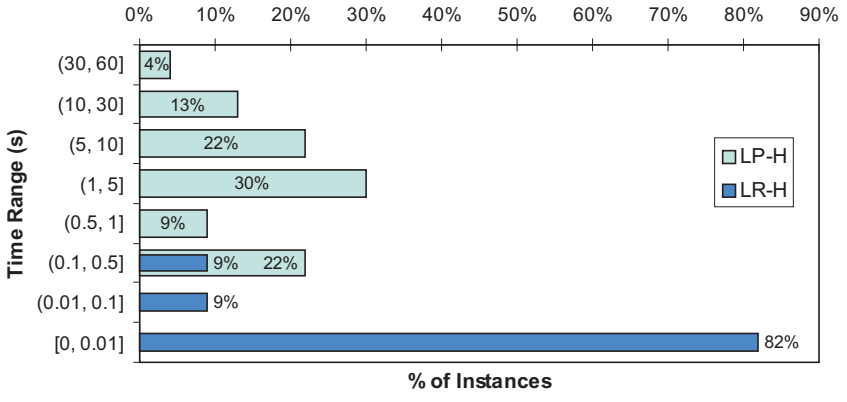


Fig. 2. Time to optimum for Lagrangian (LR-H) and LP-based (LP-H) heuristics

error was 9.8% for instance `miplib.l152lav.p`. For a better appreciation of the performance of the Lagrangian heuristic (LR-H), we compare the columns `tb(s)` from R&C and from B&C. This comparison can be visualized by inspecting the histogram in figure 2, where we restricted ourselves to the cases for which LR-H found the optimum. This histogram reveals that LR-H finds optimal solutions much quicker than the LP based heuristic (LP-H). Besides, it shows that in 82% of the cases, the optimum was found in less than 0.01 seconds and, for all instances, LR-H reached the optimum in less than 0.05 seconds.

Finally, we analyse the behavior of R&C&B&C and compare its performance with that of B&C and B&B. In our analysis, percentages are computed relative to the number of instances tested that could be solved by at least one of the approaches, say, 30. B&B is by far the worst alternative but curiously reached the optimum much sooner than its competitors in instance `mat.arc130`. Back to the other algorithms, one sees that R&C&B&C is faster than B&C in 23 out of the 30 instances (77%). Besides, the total time spent by B&C in the 29 instances solved by both methods (see last row of table 1) is 6.8% larger than that of R&C&B&C. As for the number of nodes explored during the enumeration, one can see that R&C&B&C visits less nodes than B&C, with a total reduction of 9.6% over these 29 instances. Overall, we can say that R&C&B&C is better than B&C code of [12], beating the latter in the majority of the tested instances.

6 Conclusions and Future Works

In this paper we developed an R&C algorithm for the VSP and tested it on a set of instances from the literature. The primal bounds produced by a simple Lagrangian heuristic proved to be very effective, rapidly reaching the optimum in many cases. Though the dual bounds were not as good, the R&C algorithm proved to be an excellent tool for preprocessing, quickly providing inequalities to strengthen the initial IP model, which allow the B&C algorithm of [12] to

solve the large majority of the tested instances faster than it does when working alone. To our knowledge, this turns our R&C&B&C algorithm the best exact method available for the VSP to date. Future works include the study of different relaxations, better primal heuristics and the use of other known inequalities for the VSP polytope discussed in [3] as cutting planes in the R&C algorithm.

References

1. Balas, E., Christofides, N.: A restricted lagrangean approach to the traveling salesman problem. Mathematical Programming 21, 19–46 (1981)
2. Balas, E., Ho, C.: Set covering algorithms using cutting planes, heuristics, and subgradient optimization. Mathematical Programming Study 12, 37–60 (1980)
3. Balas, E., de Souza, C.C.: The vertex separator problem: a polyhedral investigation. Mathematical Programming 103, 583–608 (2005)
4. Beasley, J.E.: Modern Heuristic Techniques, ch. 6. Blackwell Scientific Press, Oxford (1993)
5. Calheiros, F., Lucena, A., de Souza, C.: Optimal rectangular partitions. Networks 41, 51–67 (2003)
6. Guignard, M.: Lagrangean relaxation. In: Lòpez-Cerdá, M.A., García-Jurado, I. (eds.) Top, Madrid, Spain. Sociedad de Estadística e Investigación Operativa, vol. 11, pp. 151–228 (2003)
7. Kliewer, G., Timajev, L.: Relax-and-cut for capacitated network design. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 47–58. Springer, Heidelberg (2005)
8. Lucena, A.: Steiner problem in graphs: Lagrangean relaxation and cutting-planes. In: COAL Bulletin. Mathematical Programming Society, vol. 21 (1992)
9. Lucena, A.: Non delayed relax-and-cut algorithms. Annals of Operations Research 140(1), 375–410 (2005)
10. Martinhon, C., Lucena, A., Maculan, N.: Stronger k-tree relaxations for the vehicle routing problem. European Journal on Operational Research 158, 56–71 (2004)
11. Ferreira, C.E., Borndörfer, R., Martin, A.: Decomposing matrices into blocks. SIAM Journal on optimization 9, 236–269 (1998)
12. de Souza, C.C., Balas, E.: The vertex separator problem: algorithms and computations. Mathematical Programming 103, 609–631 (2005)

Finding Pure Nash Equilibrium of Graphical Game Via Constraints Satisfaction Approach

Min Jiang[1,2]

[1] Wuhan University, Wuhan Hubei 430072, China
mail.minjiang@gmail.com
[2] City University of Hong Kong, Hong Kong, China

Abstract. Considerable progress has been made in recent years in complexity analysis of Nash equilibrium, so we restrict our attention to seek it from the empirical perspective in this paper. Based on a new description format of game - stimulate – response pair proposed in the paper, we put forward a constraints satisfaction-based algorithm on this data structure to compute pure Nash equilibrium of graphical game. And then, we discuss how to employ search strategies when finding the equilibrium in different types. To evaluate our algorithm, we use a comprehensive game generator - GAMUT, to produce a mass of data and take a famous tool - gambit as competitor. At last the experimental result is presented.

1 Introduction

The classical theory of game was invented by John Von Neumann and Oskar Morgenstern in their foundational book [13]. It studies the rational behavior [1] among n ($n \geq 2$) players using the strict mathematical method. Game theory provides a powerful tool to predict and analyze the state that rational players form and the outcome due to the strategies they adopt in a complex situation.

In the past half a century, the economics and mathematics communities have already carried on deep research on the game theory and have made lots of compelling achievements. Equilibrium is one of the most important concepts in computational game theory. There are different definitions of equilibrium according to different conditions, e.g., Does cooperation exists between players? Is game carried on once or repeatedly? Does every player know any opponent's available strategies? Among all the equilibriums, Nash equilibrium may be the most important, which was put forward by John F. Nash in his famous paper [9].

In recent years, the researchers in computer science also put more and more enthusiasm to this topic. For example, e-commerce [26], large-scale network [27,28] , algorithmic mechanism design [29] and so on. Computational game theory, a brand-new field, is taking shape under the influence of a lot of relevant disciplines. It origins from the classical game theory but has its own focus. Just as Michael Kearns says:"draws strongly on classical game theory, but differs in its

[1] It means that a player chooses his own strategy according to the principle of maximizing his payoff.

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 483–494, 2007.

focus. As the name suggests, the emphasis is on computational and algorithmic issues". All of those real-life problems push the researchers to design fast and accurate solutions.

We follow this direction and restrict our attention to finding Pure Nash equilibrium from the practical point of view.

The motivation and contribution of this paper is to find the pure Nash equilibrium of graphical game by using constraints satisfaction problem (CSP) approach, which has proved successful to settle many hard problems. Such that, to open a door to explore the problem by virtue of some impactful methods.

Follow by presenting a CSP based on algorithm, we discuss how to speed up the finding process through different heuristic strategies according to the distinct equilibrium we want to seek. For accelerating the searching speed, we propose a description format – stimulate – response pair to depict a game. By this depiction, the major operation in seeking process can be reduced to find the intersection on set. Clearly, our objective will benefit from this onefold and simple execution command.

By way of being compared with other algorithms, we use a comprehensive testbed - **GAMUT** to produce a real-world dataset and employ a famous tool, **GAMBIT**, as the contrast.

The rest of the paper is organized as follows: Section 2 introduces some concepts of game and equilibrium and presents a new description format - stimulate – response pair. In section 3 we describe a reduction from graphical game to CSP and put forward an algorithm to realize it. In section 4, we discuss how to accelerate searching speed according to different types of equilibrium. Then the experimental results are presented in Section 5. In section 6, some related works in the area is described.

2 Game Theory and Nash Equilibrium

2.1 Nash Equilibrium

Nash equilibrium describes a stable situation among n players, in which every player doesn't want to change his strategy unilaterally. That is to say, any player can not gain more payoff through changing his own strategy.

For any given game, Nash has already proved in his paper that there exists at least one situation - Nash equilibrium. Generally speaking, Nash equilibrium is the abbreviation for mixed Nash equilibrium. if everyone can make a choice randomly according to some probability distribution it fixes in advance, the equilibrium in this situation is mixed. Pure Nash equilibrium is considered as a special case for mixed Nash equilibrium.

In this paper, the equilibrium is pure in the sense that each participator could choose one and only one strategy, on which all of his effort would focus. A simple example may help the audience to catch the concept of Nash equilibrium.

Suppose this game has two players, and each player has three strategies. We use the table (Table 1.) to describe the payoff of two players. The first one is player A's payoff and the second one is player B's payoff. In this game, it has

Table 1. Payoff Matrix of A Game

	B:1	B:2	B:3
A:1	2,2	**23,45**	8,10
A:2	**40,25**	0,0	5,15
A:3	10,5	15,5	**10,10**

three Nash equilibrium. If every player is rational, the combination of their choice will be one among the three pairs.

For example: if player A chooses his strategy 1, player B knows it in this Situation, he must choose his strategy 2 to maximize his payoff. But at the same time, player A also knows that if player B's choice is 2, his best response will be his strategy 1. This means that in this situation, no one wants to change his choice unilaterally.

2.2 Graphical Game

In spite of the fact that Nash equilibrium certainly exists which has been certified in Nash's paper, in a particular field, e.g., computer science, what people concerns is how to compute it. The researchers have presented many algorithms to solve this problem. There are two comprehensive surveys [16,17] which summarize the temporal approaches, algorithms and tools. But we know that a game described in strategy form[2] (or normal form) demands exponential space requirement with the increase of the number of players, so we cannot hope to have an effective algorithm.

The first problem needed to be solved is how to represent a game compactly so as to avoid the exponential space requirement. The experience and fact tell us that it is effective to use the locality of a complex problem to conquer it.

The researcher in computational game community follows this direction. Generally speaking, the locality of a game can be divided into two types: locality of action and locality of player.

The first one means one player shows his influence upon other participants through the strategy that he chooses. The representative of this type is Rasenthal's *congestion game* [20] and *local effect game* [19] defined by K. Keyton-Brown et al. Congestion game describes a scenario, where every player is only interested in a subset of available resources. We can use the following formulate to describe the payoff of player i: $P_i(SR_i, n) = \sum_{r \in SR_i} F_r(n_r)$,where $SR_i \subseteq R$, n_r is the number of players who are interested in resource r and F_r is an arbitrary function. Local effect game highlights the asymmetry of the influence of locality, namely, strategy a affects strategy b if the payoff of the player taking strategy b depends on a function $F_{a,b}$ of the number of players choosing strategy a. Please note that, the asymmetry is embodied by $F_{a,b} \neq F_{b,a}$. Because a strategy does not have an influence on all the other strategies, the local effect game can represent a game succinctly.

[2] It means to describe a game via tabular format.

The representative of the latter kind, locality of player, is the widely-studied Graphical Game. As the name suggests, it depicts a game with a graph where nodes represent players and edge means that the two players can affect each other [3]. The assumption it is based on is that every player often has direct relations only with few players, and his influence on the global situation is spread out through these players. If the number of his neighbors is K - $|Neighbor(i)| = K$ and $K \ll n$, then the size of every player's payoff matrix can be reduced at once. So it is a good idea to represent a game compactly.

Definition 1. *A graphical game is a 3-tuple* $GG =< Graph, S, U >$*, where* $Graph$ *is a undirected graph* $G =< V, E >$ *;* S *is a set of strategies, and it is very similar to the classical definition;* $U = \{U_1, \ldots, U_i, \ldots, U_n\}$ *is a set of payoff functions, where* $U_i = \prod_{j \in Neighbor(i)} S_j \rightarrow \mathbf{N}$*, and it maps player* i*'s strategy and those of his neighbor to a value. Commonly, payoff matrix is used to represent* S *and* U*.*

In the present paper, we put forward a new description format called **stimulate – response pair** to describe the effect among player i and his neighbors. A stimulate – response pair is $sr_{ij} =< \{S_{neighbor-i}\}, \{s_{ij}\}, outcome >$, in which $\{S_{neighbor-i}\}$ means a combination of strategies of player i's neighbors (not including player i). $\{s_{ij}\}$ denotes a strategy j player i chooses when his neighbors play $\{S_{neighbor-i}\}$. *Outcome* is player i's payoff in this situation. So, every player's payoff matrix can be transformed to some simulate-response pairs.

Sharp-eyed readers will find that this representing format is a transformation of payoff matrix, but it is easy to be understood and implemented. Please note that, we use set to describe the elements in this format. It means we don't care the difference of scheduling when the players choose the strategies. Through it, the operation of finding Nash equilibrium is reduced to the operation on set. Obviously, this reduction makes the search process more intuitionistic and faster.

For the convenience of implementing, we define two functions on it:

- **Stimulate**$(sr_{ij}) = \{\{S_{neighbor-i}\} | \{S_{neighbor-i}\} \in sr_{ij}\}$, which returns the combination of strategies of player i's neighbor in sr_{ij};
- **Response**$(sr_{ij}) = \{\{s_{ij}\} | \{s_{ij}\} \in sr_{ij}\}$, which returns the strategy of player i in sr_{ij}

Through the above definitions, we reformulate a graphical game as follows:

Definition 2. *A graphical game* $GG =< Graph, SR >$*, where* $SR = \{SR_1, \ldots, SR_n\}$*,* $SR_i = \{sr_{i1}, ..., sr_{in}\}$*.* SR_i *can be considered as the classical payoff matrix and* sr_{ij} *represents the* j*th stimulate – response pair of player* i*.*

Obviously, though the above definitions have the same description ability, each has its strong points. In the following part of this paper, we will use them alternately according to the actual need without puzzling readers.

[3] Surely, we can use different concepts in graph theory, i.e., directed graph or hyper graph, to meet our requirements.

Definition 3. *Pure Nash Equilibrium.We say a profile SP is pure Nash equilibrium. For every player i and all his own available (pure) strategies S_i in SP: $U_i(SP) \geq U_i(i : s^{'}), s_i^{'} \in S_i$. This means no player can increase his payoff through changing his strategy from s_i to $s_i^{'}$ in SP.*

Our attention is restricted to the computation of pure Nash equilibrium, so the definition and algorithms about mixed Nash equilibrium was referred to [16].

We know that game theory is based on the assumption that every player is intelligent and rational. Intelligence means that any player knows his opponents know everything about this game, at the same time he possesses the ability to reason out the current situation as other players do. Rationality means that every player adopts his strategy for the sake of maximizing his payoff. According to the two points the definition of the best response is given as follows:

Definition 4. *$BR_i(SP)$ is player i's best response in a profile SP. If $BR_i(SP) = \{s_{in} | s_{in} \in S_i, \forall s_i^{'} \in S_i : U_i(S_{-i}; s_{in}) \geq U_i(S_{-i}; s_i^{'})\}$.*

Intuitively speaking, $BR_i(SP)$ is a set of player i's strategies, which let player i get his maximal payoff in the profile SP. Obviously, it is very easy to use stimulate – response pair to describe it again. For the limitation of the paper, we don't do that. We can imagine that when a rational player makes a decision, he always chooses a strategy as response among his best responses according to other opponents' choices.

2.3 Complexity Issues

Like other hard problems, the study concerning Nash equilibrium can be divided into theory analysis and practical computing. In recent years, exciting breakthroughs have been made in the complexity analysis of Nash equilibrium. The theoreticians have proved that 4-**Nash** [14] , 3-**Nash** [1,5] and 2-**Nash** [2] are **PPAD**-complete [4], especially the last proof presented by X.Chen and X.Deng which settled the big open problem puzzling relevant communities for a long time.

For pure Nash equilibrium, Gottlob et al. [8] have proved that determining whether a game has a pure Nash equilibrium is NP-complete, and hardness holds even if the game is described by graphical form, which has 3-bounded neighborhood, and where each player is allowed to play at most three strategies.

All of those proofs show it is a hard problem to find a Nash equilibrium, even if the condition is restricted.

3 A CSP's Solution to Pure Nash Equilibrium

3.1 Constraint Satisfaction Problem (CSP)

Many computational problems from different application fields can be considered as constraint satisfaction problems. For example, the problems of scheduling

[4] Because the decision form of existence of a mixed Nash equilibrium is always yes, speaking of NP-completeness would be meaningless.

a collection of tasks [21], laying out a chip [22], even to understand the visual image [23], can all be done in this way. ACM(Association for Computing Machinery) identified it as one of the strategic directions in computer research. Many facts[24] tell us CSP is a powerful tool to settle some hard problems, especially, combinational optimization problem. So we hope to find Pure Nash equilibrium by using CSP approach.

At beginning, let we recall the statements about CSP in the [25] first. A CSP is a 3-tuple $CSP =< X, D, C >$, where $X = \{X_1, ..., X_n\}$ is a set of variables, and D_i is a non-empty domain of X_i; $C = \{C_1, ..., C_m\}$, and every C_i involves some subsets of the variables and specifies the allowable combinations of the values for those subsets.

A search process for a CSP's solution can be constructed as follows:

1. Initial state: no variable is assigned to a value;
2. Successor function: assign a value to an unassigned variable, which ensures that it does not conflict with the previously assigned variables;
3. Check whether all constraints are satisfied;
4. Cost of path: assign a fixed cost for every search step.

CSP is a problem which was deeply studied, and there are a huge amount of literature and algorithms. Although it is hard to solve theoretically, i.e., the famous **3-SAT** problem is a special case, sometimes we still can find the solution effective in practice. Another predominance of CSP is that the simple yet powerful description ability makes one possess the capability to handle sophisticated problem even if he is not very familiar with it.

3.2 From Game to CSP

In this section, we will discuss how to transform finding pure Nash equilibrium in a given game into finding a solution of CSP. Note that, the players in all the games we discussed in the paper are finite and every player has finite strategies. We present our reduction as follows:

1. Every variable X_i denotes player i;
2. The domain of every variable is the stimulate – response pairs;
3. **Unary constraint:** if this constraint is satisfied, it means the strategies player i chooses are the best response to his neighbors. When applying this constraint, all the stimulate – response pairs will be removed except that it is the best response to a certain "stimulation";
4. **Binary constraint:** It means that a legal assignment lets two adjacent players have corresponding best responses.
   In the algorithm presented in this paper, we use stimulate – response pair to implement this constraint. Assume that, the player j is the player i's neighbor. $SR_i(SR_j)$ is a set of stimulate – response pairs of player i (player j) , which is the result of applying the unary constraint. If there are

$((\text{Re}sponse(sr_{im}) \cap \text{Re}ceive(sr_{jn})) \neq \phi) \wedge ((\text{Re}ceive(sr_{im}) \cap \text{Re}sponse(sr_{jn})) \neq \phi)$, we say this binary constraint is satisfied. The meaning of this unary constraint is that two adjacent players have his own strategy respectively; each strategy is the best response to his opponent's strategy which is also a best response according to itself. In other words, there is a circle of best response between two adjacent players.

5. **Constraints propagation:** assign a value to each variable in turn, with the requirement that the current assignment should not conflict with the previous assignments.

Clearly, if an assignment to every variable can satisfy all the above constraints, it is a pure Nash equilibrium for its corresponding game, otherwise the game doesn't have a pure Nash equilibrium.

3.3 An Algorithm on Graphical Game

We present our algorithm as follows.

Algorithm 1. PNE-CSP (sketch)

```
Data: Graphical Game GG with stimulate – response pair
Result: pure Nash equilibrium or "not exist"
begin
    Step.1 Arbitrarily select a node as root, and do breadth-first search, form a
    sequence X_1, X_2, ..., X_n; Step.2 for i = 1 To n do
        APPLY-UNARY-CONSTRAINTS(X_i);
    Step.3 for i = n Downto 2 do
        Add all Neighbors of X_i to queues;
        while not empty the queues do
            temp = Move-First(queues);
            APPLY-BINARY-CONSTRAINTS(temp,X_i);
            if |Domain(temp)| = 0 then
                Return "not exist"
    Step.4 for i = 1 To n do
        assign any value for X_i is consistent with the value assigned for X_j,
        where X_j is the parent of X_i.
end
```

For the limitation of space, we just describe the framework of the algorithm. Please note that the underlying graph of our algorithm is undirected and tree-like. The undirected graph means we deem the effect of neighbors is symmetrical.

Remark 1. **step 1.** To makes the parent of each node precedes it except root. The reason will be explained later.

Step 2. To removes all the strategies a rational player will not choose, so the search space is reduced.

Step 3. The most important part of it is *Apply-Binary-Constraints*(X_i, X_j), and the result of it ensures every value remained in the domain of X_i can find a corresponding value in the domain of X_j, which guarantees the binary constraint between them is satisfied. The order of those operations is from leaf to root. The ordering from step 1. assures that the latter operation will not affect the previous results in this step. If the domain of any variable is empty, the search process is terminated. It means there is no pure Nash equilibrium for the corresponding game.

Step 4. A classical constraints propagation. We can use different methods to speed up search process with respect to the equilibrium we want to find.

4 Discussion

Through the above analysis, we know that if one wants to find an equilibrium without using its distinguishing features, the result may not be the best. In other words, when computing an equilibrium, we should take full advantage of its features to accelerate the speed of searching.

For example, if we want to find a pure Nash equilibrium, which, as we know, may not exist, we should choose the variable with fewest "legal" values. Because it picks a variable that is most likely to cause a failure as fast as possible, the search tree will be pruned soon. This method is called "Minimum Remaining Values" (MRV) or "Fail-Fist" in CSP's terms. In other cases, if we only want to seek a mixed Nash equilibrium, the strategy will be changed. On the contrary, we choose the variable with most "legal" value if possible. The reason is that mixed Nash equilibrium certainly exists, so we can find it as soon as possible without backtracking. However, it is not advisable to choose this strategy to look for all the mixed Nash equilibriums.

On all accounts, concerning the problem of finding equilibrium with intelligent approaches, we must use the heuristic strategies with special purposes to get satisfying results.

5 Experimental Results

Our experiment is designed to evaluate the performance between Gambit, a most famous tool, and our algorithm on finding pure Nash equilibrium on graphical game. The dataset used in this experiment is produced by GAMUT and the underlying graph is n-try tree. We compare the results of 2 programs which run on 20 different cases. Case 1 denotes 3 persons - 30 actions; Case 2 denotes 3 person - 31 actions; ...; Case 20 is 3 person - 49 actions. Our program employs JCL package, which is developed by EPFL [5]. For possessing the bin-constraint needed in our algorithm, we extend it. Please note, its performance is not the reason that we choose JCL. Strictly speaking, it is not the most fast one, but it is the only

[5] http://liawww.epfl.ch/JCL/index.htm

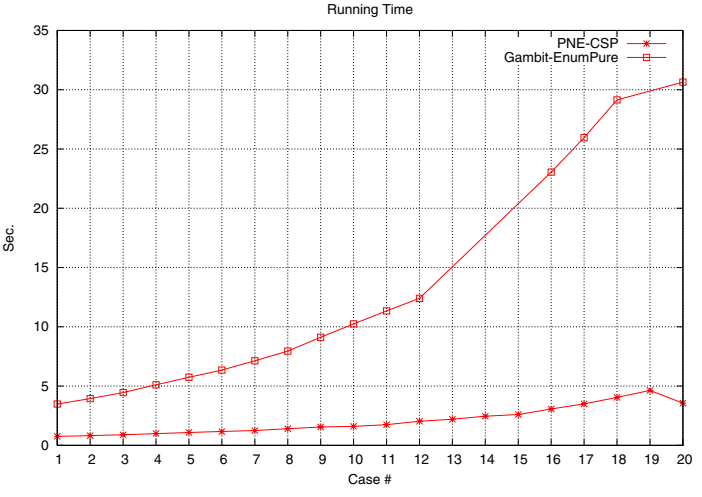


Fig. 1. The Running Time

package that we find can provide the continuous CSP function at present. Despite not using this feature in the paper, it has established the foundation for our future work. In the present paper, all programs are implemented in java 5.0, and the version of GAMBIT is 2006.1.20. The running environment are linux fc 5.0, Pentium(R) 1.86G with 1G memory. The running time is presented in Figure 1.

The result shows that when the size of game largens, it is not obvious that the increase of consumption in time of our algorithm and the difference between two methods is easy to be found.

6 Related Work

The computation of equilibrium is an amazing but hard problem, which has recently employed many methods related to different fields. Among all the existing algorithms, Lemke-Howson's algorithm is the most famous one. It is a complementary pivoting algorithm, where an arbitrary selection of an action for the first player determines the first pivot, then every successive pivot is determined by this state, until an equilibrium is found. Gambit is an implementation of Lemke-Howson method. As a toolkit, Gambit consists of some different programs with several uses. Gambit-enumpure is used for computing pure Nash equilibrium.

Another well-known program is Gametracer, which includes two algorithms, Global Newton Method (GNM) and Iterative Polymatrix Approximation (IPA), both due to Srihari Govindam and Robert Wilson [3]. GNM perturbed a game to another one that has a known equilibrium, and then traced the solution back to the original game as the magnitude of the perturbation approaches zero. IPA approximates the game with a bimatrix game, one in which the interaction between each pair of players is not affected by any other players. A player's

payoff is the sum of his payoffs which is decided by the interactions with other players.

In this paper, we are interested in graphical game, which is defined by Kearns et al. in [12]. We consider that probabilistic graphical models motivated graphical games and the algorithm to do probabilistic inference on trees led to the idea of a similar message passing algorithm for computing Nash equilibrium. In Kearns's paper, they proposed three algorithms to compute Nash equilibrium. The first is an abstract algorithm, the second based on the first tries to find approximation Nash equilibrium in polynomial time and the last is an exact but inefficient version.

Another paper by the same authors [11] tries to efficiently compute an exact Nash equilibrium in tree-like graphical game. It is a pity that Elkind et al. in [7] prove that this algorithm is incorrect and any 2-pass message passing algorithm of this type has no hope to solve the problem in polynomial time even on bounded-degree tress with pathwidth 2. An algorithm proposed in [10] uses message passing algorithm to compute Nash equilibrium in general (loopy) graphs, which is also based on 2-pass message passing algorithm.

Although pure Nash equilibrium is computationally trivial, it is a very primitive, intuitive, and straightforward concept of rationality. In very recent, Daskalakis et al.[18] proposed a reduction from graphical games to Markov random fields so that pure Nash equilibria in the former can be found by statistical inference on the latter. They use junction tree algorithm to achieve it.

Surprisingly, to the best of our knowledge, we are aware of little work on pure equilibrium based on CSP approach.

[6] uses the approach to find approximation Nash equilibrium. They define a function called regret function which denotes that any player should not exceed ϵ. The constraint C_i for a player ensures that he regrets to being no more than a ϵ. Until now, the most compelling result by using CSP method to seek mixed Nash equilibrium in normal form is presented in [15], where they employed a simple search approach and tested 22 types of game. But we still think the result remains to be improved and the reason has been discussed in Section 4 in detail. The authors of this paper have developed a comprehensive game generator - **GAMUT**, which can produce many different types of game.

Please note that, the difference between our approach and the above CSP-based approaches is that we take profile but not strategy as value of a domain and we use stimulate – response pair to depict it. Through this approach, we convert major part of comparing operations between numbers to elements of set. Clearly, it will accelerate the calculating speed.

7 Conclusion and Future Work

Equilibrium is the most important concept in computational game theory. In recent years, the complexity analysis of this problem has made some exciting breakthroughs, so it seems more important to find it at a fast speed in practice.

In this paper, we put forward a description format, stimulate – response pair, to depict a game, which can convert the major part of comparison between the real numbers into the comparison between the elements of set. Clearly, it will accelerate the search speed. At the same time, we propose an algorithm which can reduce finding pure Nash equilibrium on graphical game to seeking a solution of CSP. The experiment shows that our algorithm can find the Nash equilibrium correctly and has some advantages over Gambit in some extent. We hope this approach open a door to explore this hard problem from a different viewpoint.

Meanwhile, we realize that the time requirement of this kind of problem will increase exponentially with the increase of the description length of it. On the contrary, that means the time requirement will be decreased exponentially with the description length cut down. It gives us an indication to solve this problem. In the future, we will continue to study how to rapidly find mixed Nash equilibrium or approximation Nash equilibrium from the empirical perspective.

References

1. Chen, X., Deng, X.: 3-NASH is PPAD-Complete. In: ECCC 2005, TR05-134 (2005)
2. Chen, X., Deng, X.: Settling the Complexity of 2-player Nash Equilibrium. In: ECCC 2005, TR05-140 (2005)
3. Blum, B., Shelton, C., Koller, D.: A continuation method for Nash equilibria in structured games. International Joint Conference on Artificial Intelligence (IJCAI), 757–764 (2003)
4. Papadimitriou, C.: Algorithms, games and the internet. In: STOC, pp. 749–753. ACM Press, New York (2001)
5. Daskalakis, C., Papadimitriou, C.H.: Three-Players Games are Hard. In: ECCC 2005, TR05-139 (2005)
6. Koller, D., Vickrey, D.: Multi-agent algorithms for solving graphical games. In: AAAI 2002. Eighteenth National Conference on Artificial Intelligence, pp. 345–351 (2002)
7. Elkind, E., Goldberg, L.A., Goldberg, P.W.: Nash Equilibria in Graphical Games on Trees Revisited. In: Electronic Commerce (EC), pp. 100–109. ACM Press, New York (2006)
8. Gottlob, G., Greco, G., Scarcello, F.: Pure Nash equilibria: hard and easy games. In: TARK. Theoretical Aspects Of Rationality And Knowledge, pp. 215–230 (2003)
9. Nash, J.F.: Non-cooperative games. Annals of Mathematics (54), 286–295 (1951)
10. Oriz, L., Kearns, M.: Nash Propagation from Loopy Graphical Games. Advances in Neural Information Processing Systems, 817–824 (2003)
11. Littman, M., Kearns, M., Singh, S.: An Efficient Exact Algorithm for Singly Connected Graphical Games. NIPS, 817–823 (2001)
12. Kearns, M., Littman, M., Singh, S.: Graphical Models for Game Theory. In: UAI-01. Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence, pp. 226–253. Morgan Kaufmann, San Francisco (2001)
13. Morgenstern, O., von Neumann, J.: Theory of Games and Economic Behavior. Princeton University Press, Princeton, NJ (1980)
14. Goldberg, P.W., Papadimitriou, C.H.: Reducibility Among Equilibrium Problems. In: STOC'06, Seattle, WA, pp. 61–70. ACM Press, New York (2006)

15. Porter, R., Nudelman, E., Shoham, Y.: Simple Search Methods for Finding a Nash Equilibrium. In: AAAI 2004. Proceedings of the Nineteenth National Conference on Artificial Intelligence, pp. 664–669 (2004)
16. McKelvey, R.D., McLennan, A.: Computation of Equilibria in Finite Games, vol. 1. Elsevier, Amsterdam (1996)
17. von Stengel, B.: Computing Equilibria for Two-person Games, vol. 3. North-Holland Press, Amsterdam (2002)
18. Daskalakis, C., Papadimitriou, C.H.: Computing Pure Nash Equilibria in Graphical Games via Markov Random Fields. In: Eletronic Commerce (EC), pp. 91–99. ACM Press, New York (2006)
19. Leyton-Brown, K., Tennenholtz, M.: Local-Effect Games. IJCAI, 772–780 (2003)
20. Rosenthal, R.W.: A class of games possessing pure-strategy Nash equilibria. International Journal of Game Theory, 65–67 (1973)
21. Dhar, V., Ranganathan, N.: Integer Programming vs Expert Systems: An Experimental Comparison. Communications of the ACM (33), 323–336 (1990)
22. de Kleer, J., Sussman, G.J.: Propagation of Constraints Applied to Circuit Synthesis. Circuit Theory and Applications, 127–144 (1980)
23. Davis, A.L., Rosenfeld, A.: Cooperating Processes for Low Level Vision: A Survey. Articial Intelligence (17), 245–263 (1981)
24. Kumar, V.: Algorithms for Constraint Satisfaction Problems: A Survey. AI magazine, 32–44 (1992)
25. Russell, S., Norvig, P.: Artificial Intelligence: a Modern Approach. Prencite Hall.
26. Feigenbaum, J., Papadimitriou, C.H., Shenker, S.: Sharing the Cost of Muliticast Transmissions. Journal of Computer and System Sciences (63), 21–41 (2001)
27. Korilis, Y.A., Lazar, A., Orda, A.: Achieving Network Optima using Stackelberg Routing Strategies. IEEE/ACM Transactions of Networking (5), 161–173 (1997)
28. Roughgarden, T.: On the Severity of Braess Paradox: Designing Networks for Selfish Users is Hard. Journal of Computer and System Sciences, 922–953 (2006)
29. Nisan, N., Ronen, A.: Algorithmic Mechanism Design. Games and Economic Behavior 35(1-2), 166–196 (2001)

A New Load Balanced Routing Algorithm for Torus Networks

Jiyun Niu[1], Huaxi Gu[2], and Changshan Wang[1]

[1] School of Computer Science, Xidian University, Xi'an, China 710071
[2] State key lab of ISN, Xidian University, Xi'an, China 710071
njy_001@163.com, hxgu@xidian.edu.cn

Abstract. Originated from gas diffusion phenomenon, a new load balanced routing algorithm named Gas Diffusion Based Load balanced Routing (GDLR) is proposed for torus networks. GDLR estimates global congestion in the network by the number of potential deadlocked packets. Based on this information, the probabilities of all possible ports a packet may use are accordingly determined. Subsequently, packets are sent out with the stochastic policy, which leads to data load spreading with consequent automatic load balancing. Finally, simulations have been carried out on 2D torus networks by using OPNET software. The results obtained show that GDLR achieves a better performance than other popular algorithms such as Dimension Order, Duato, and GAL.

Keywords: Interconnection networks, Routing, Torus.

1 Introduction

Direct interconnection networks have received wide applications, from processor memory interconnect [1], I/O interconnect [2], to switch and router fabrics [3]. Torus network is one of the typical networks [5].

Routing algorithms, which determine how a packet travels from its source to destination, are crucial for the network performance. An efficient routing algorithm can improve the network throughput and reduce the average latency. Routing algorithms generally fall into two categories: deterministic and adaptive. The former always provides the same road between a source-destination pair, while the latter offers many optional paths. Dimension order (DOR) routing [4], very popular in practice for its simple hardware implementation, is a typical representative of deterministic routing algorithms. It routes a packet along productive dimensions (A productive dimension is the dimension in which a packet has not reached its destination.) in a predefined order. Many commercial products use dimension order routing, such as Intel Paragon, MIT J-machine and Cray T3D [5, 6]. Though it can distribute the traffic evenly under the uniform traffic, DOR performs poorly under congested conditions due to its deterministic way to route packets regardless of the current network state.

Adaptive routing was proposed to overcome the performance limitations of deterministic routing. A minimal adaptive routing can route packets along any of the

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 495–503, 2007.

shortest paths in the topology. Duato proposed such an adaptive algorithm in [7]. The algorithm requires two types of channels, represented as Class A and Class B. Class A contains at least two virtual channels for Torus (one for mesh), in which deterministic routing is employed. By contrast, in the virtual channels belonging to class B fully adaptive routing is used. The packets can adaptively choose any virtual channel available from class B. Only if all the virtual channels of class B are busy can a packet turn to channels that belong to class A.

The routing algorithms mentioned above attempt to provide low latency on local traffic. However, they perform poorly under some adversarial traffic, such as hotspot or tornado traffic. Therefore, people try to propose algorithms that can yield satisfactory performance under different traffic. Valliant's randomized algorithm [8] uses a random node as an intermediate destination so as to give good performance on worst-case traffic. However, it performs poorly on local traffic due to its complete loss of locality. Arjun proposes a load balanced routing algorithm (GAL in [9]) that senses global congestion by injection queue length at the source node to decide the directions to route in each dimension. Though GAL matches the throughput of minimal algorithms on local patterns and of Load balanced algorithms on difficult patterns, it is complex to implement and slow to adapt to changes in traffic. What is more, it has very high latency once it starts routing traffic non-minimally. ACQ [10] makes improvements on it. It uses channel queues rather than the injection queue length to estimate global congestion. It achieves good performance on adversarial traffic but gives poor performance on benign traffic.

In this paper, we introduce Gas Diffusion based load balanced routing (GDLR)-a minimal, adaptive routing algorithm for k-ary n-cube networks. GDLR uses time-out criteria to detect deadlocks, and the detected deadlocked packets are absorbed into local node and retransmitted at a later time. GDLR estimates network congestion by the number of potential deadlock packets rather than the queue length used by GAL and ACQR. What is more, GDLR routes a packet according to the congestion degree of local ports. In this way, GDLR balances the traffic load in each direction to achieve a good performance on different traffics. The rest of the paper is organized as follows.

Section 2 gives a detailed description of the GDLR algorithm. We measure the performance of GDLR in section 3 and compare its performance with several existing algorithms through simulation. Finally we give some conclusions in section 4.

2 Gas Diffusion Based Load Balanced Routing Algorithm

2.1 GDLR Algorithm

GDLR is obtained from the inspiration of gas diffusion phenomenon. As is known, when there are differences in the density of the gas in different locations, the high-density gas will diffuse to the location where the gas is of low density. This process continues until the density of the gas in the two places becomes equal. When mapped to routing issue, the phenomenon turns out to be that packets turn from more congested ports to less congested ones, until the congestion degree of all the ports becomes approximately same.

Unlike GAL or ACQ, it exploits the number of potential deadlocked packets, which is more accurate, rather than the queue length to sense the network congestion. Based on this local information, it determines the probability of all possible ports packets may use. The less congested one port is, the more routing probability it owns. In other words, it is inclined to give more opportunity to less congested ports. The underlying strategy is the greedy policy. Each node tries to send out packets as quickly as possible, then the average ETE delay of the network will be reduced and the network throughput will be increased. However, in order not to follow the network fluctuations to reach a good performance in the long run, some congested ports are also used with low probability. A detailed description of the algorithm implemented on 2D-Torus is shown as below.

Algorithm GDLR

// D: the set of the productive dimensions for a certain packet. For 2D-Torus, there are at most two productive dimensions, 0 and 1.
// ||D||: total members of the set D.
// P_0 : the probability of routing in dimension 0.
// P_1 : the probability of routing in dimension 1.
// P: = { P_0 , P_1 }, the set of routing probabilities of all productive dimensions.
// M_0 : the number of the potential deadlocked packets in the port representing productive dimension 0.
// M_1 : the number of the potential deadlocked packets in the port representing productive dimension 1.
// M: = { M_0 , M_1 }, the set of the numbers of the potential deadlocked packets in ports representing all productive dimensions.
// α_0 : the factor to update the routing probability of the port representing productive dimension 0.
// α_1 : the factor to update the routing probability of the port representing productive dimension 1.

P_0 : =0.5, P_1 : =0.5
for each packet in the input subqueue
    Calculate the set D for the packet
    if ||D||=1
        if D={0}
            P_0 : =1, P_1 : =0
        else
            P_0 : =0, P_1 : =1
        end if
    else
        calculate the set M for the packet

if $M_0 > M_1$

$P_0 := P_0 - \alpha_0$, $P_1 := P_1 + \alpha_0$

end if

if $M_0 < M_1$

$P_0 := P_0 + \alpha_1$, $P_1 := P_1 - \alpha_1$

end if

end if

for each member P_j in P

if $P_j > 1$

$P_j := 1$

endif

if $P_j < 0$

$P_j := 0$

endif

route the packet in dimension 0 with the probability P_0, dimension 1 with P_1

end for

The determination of the factor α for updating the routing probability of a certain port requires much technique. The simplest strategy is to set the value of α as a constant:

$$\alpha = C,\ C \in (0,1) \tag{1}$$

that is, independently from the conditions of network. Another way is to adaptively determine the value of α based on the congestion conditions of local ports. In our experiments, the latter method was applied, and the algorithm showed moderately good performance. The formula used is showed below.

$$\alpha_0 = \frac{M_0 - M_1}{M_0 + M_1} \tag{2}$$

$$\alpha_1 = \frac{M_1 - M_0}{M_0 + M_1} \tag{3}$$

This model is very simple but effective. However, it is not proper for high-dimension networks where a packet may have more than two productive dimensions. A solution of this problem is given in section 2.3. And more elaborate models may be produced to improve the performance.

2.2 Deadlock Detection and Recovery

Deadlocks can be detected by a simple time-out criterion, similar to those suggested in [11, 12]. Given a threshold Tout, if a packet cannot be sent out within Tout period,

then it is supposed to be a potential deadlocked packet to be ejected from the network. Instead of killing the detected deadlocked packet [12], GDLR uses a software-based recovery mechanism to absorb the packet and retransmit it later.

Theorem 1: GDLR is livelock free and it can resolve every detected deadlock.

Proof: GDLR makes all the packets choose minimal path, thus providing livelock freedom.

Suppose that a deadlock has been detected. GDLR uses software-based deadlock recovery mechanism to deliver a potential deadlocked packet to local node, thus freeing the resources it occupies. Other packets forming the deadlock cycle can use the freed resources and continue their travel. So far, the deadlock has been resolved.

2.3 Extensions

We only give a description of the algorithm implemented on 2D torus before. Here we try to apply the algorithm to higher dimensional torus network and other popular networks.

Given a k-ary n-cube network and a set D of $d(d \le n)$ productive dimensions for a certain packet to be routed, D = $\{D_0, D_1 ... D_{d-1}\}, D_j \in [0, n-1], 0 \le j \le d-1$, as well as a set M = $\{M_0, M_1 ... M_{d-1}\}$, with $M_j (0 \le j \le d-1)$ representing the number of the potential deadlocked packets in the port for the productive dimension $D_j (0 \le j \le d-1)$. Then, we accordingly need to determine a set A = $\{A_0, A_1 ... A_{d-1}\}$ with A_j $(0 \le j \le d-1)$, the factor for updating the probability of routing in productive dimension $D_j (0 \le j \le d-1)$, initialized by the value 0, as well as a set P = $\{P_0, P_1 ... P_{d-1}\}$ with P_j $(0 \le j \le d-1)$ representing the routing probability of the productive dimension $D_j (0 \le j \le d-1)$. The determination of set A requires experience, and the set P should meet the requirement P_j

$$\sum_{j=0}^{k-1} P_j = 1 \tag{4}$$

Here we just give one way to figure out the set A and P.

$$P_j = P_{ini} - A_j,\ P_{ini} = \frac{1}{\| D \|},\ 0 \le j \le d-1,\ \| D \| = d \tag{5}$$

$$A_j = \frac{M_j - M_{avg}}{M_{sum}}, \text{ Where } M_{avg} = \frac{M_{sum}}{\| D \|}, M_{sum} = \sum_{j=0}^{k-1} M_j \tag{6}$$

However, further research is required to decide whether this way will yield the best performance.

3 Simulation Study

3.1 Evaluation Methodology

In this section, we give a comparison of GDLR with other popular algorithms including Dimension Order routing algorithm, Duato's algorithm, and GAL. The simulations have been carried out on 8-ary 2-cube network as well as 16-ary 2-cube network by OPNET simulation software [13]. Due to space constraints, we only present the results of 8-ary 2-cube network. The results obtained for 16-ary 2-cube network are of similar trends.

Only virtual cut through switching mechanism [14] is used in the simulations, but the algorithms are also suitable for wormhole switching [15] and store-and-forward switching [16, 17]. The simulations are based on the following assumptions if not specially stated. Packet length distribution is a specific distribution SP (Size and Percent) based on the IP (Internet Protocol) packet size and percentages sampled over a two-week period [18]: 40 bytes (56% of all traffic), 1500 bytes (23%), 576 bytes (16.5%) and 52 bytes (4.5%), which makes the results more convincing.

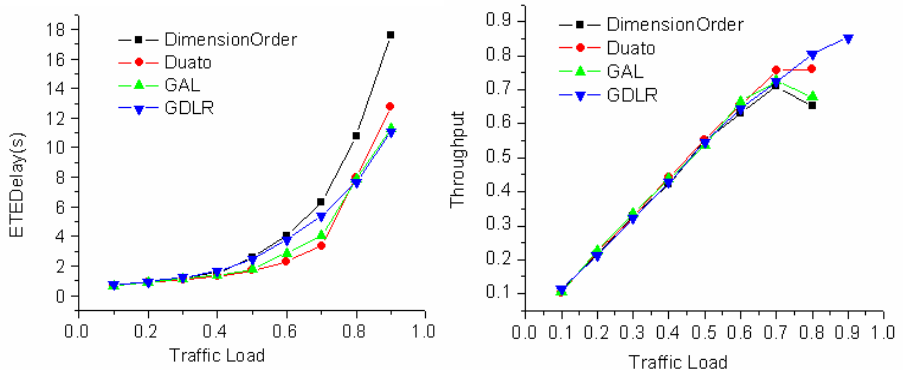
The traffic patterns used include the uniform, the hotspot and the tornado traffic pattern. In the uniform traffic pattern, each node sends packets to all other nodes in the network with the same probability. In the hotspot traffic pattern, one or more nodes are designated as the hotspot nodes, which receive hotspot traffic in addition to regular uniform traffic. In the tornado traffic pattern, the node (i, j) only sends packets to node (i, (j+[k/2]-1) mod k), with k being the radix of the network.

The nodes operate asynchronously. They generate packets at the time intervals following the negative exponential distribution. And packets arriving at a destination node are consumed immediately. For a fair comparison, four virtual channels are used for each algorithm.

The performance of a routing algorithm is evaluated in terms of two main metrics: average ETE (End To End) delay and normalized network throughput. The average ETE delay is defined as the mean time from the time a packet is generated to the time it reaches its destination, while the normalized network throughput is equal to the current network throughput divided by network capacity.

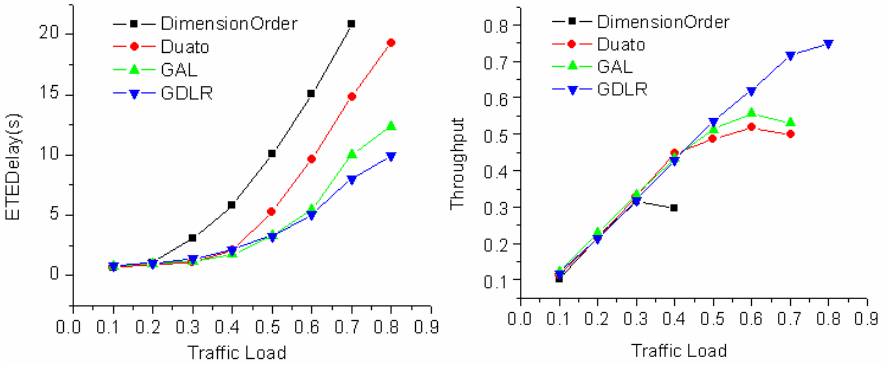
3.2 Simulation Results

Fig.1 shows the results of the four algorithms under uniform traffic. For traffic loads less than 0.4 where little congestion is present, the four algorithms yield almost identical latency and throughput. However, as traffic loads increase, the gap between them is broadened. Dimension Order, the deterministic minimal routing algorithm, reaches its saturation point firstly. GAL and Duato are both adaptive algorithms and give approximately the same performance. GDLR gives a good performance under high traffic load by using the probability method to route a packet.



(a) ETE Delay (b) Throughput

Fig. 1. Performances of the four algorithms under uniform traffic

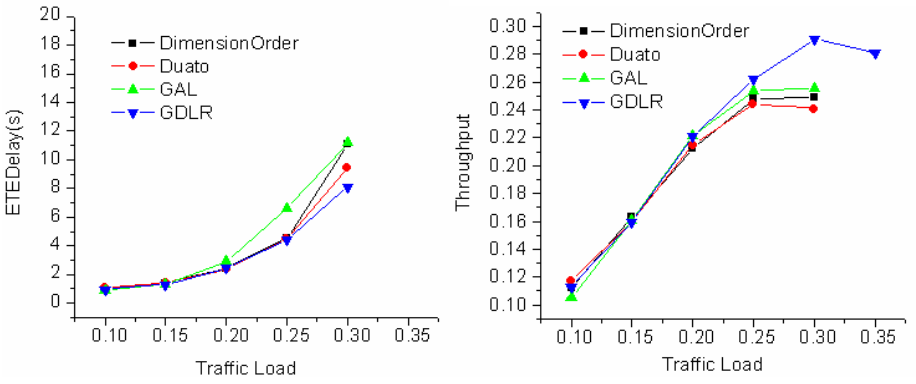


(a) ETE Delay (b) Throughput

Fig. 2. Performances of the four algorithms under hotspot traffic

What Fig.2 shows is the comparison of the four algorithms under hotspot traffic. As is shown, Dimension order performs poorly under the hotspot traffic, the first to reach saturation at traffic load 0.3. The reason is that Dimension order determines the same path between a source and destination node regardless of network conditions. Duato is the second with its saturation point of 0.4, due to its adaptive routing method. GAL has its saturation point 0.1 lower than Duato, because it can turn to non-minimal path when congestion is detected. GDLR turns out to be the best one, with the lowest latency and the highest throughput.

The performances of the four algorithms under tornado traffic are showed in Fig.3. As can be seen, Dimension order and Duato perform poorly under the tornado traffic for the reason of adopting only minimal routing. GAL achieves a good performance under high traffic load, due to the fact that it can adopt non-minimal path when congestion is found and thus can balance the load and relieve the congestion. GDLR, also a minimal algorithm, dramatically performs very well, with its saturation point of 0.3, due to its stochastic decisions for routing.



(a) ETE Delay (b) Throughput

Fig. 3. Performances of the four algorithms under tornado traffic

4 Conclusion

In this paper, we introduce a load balanced routing algorithm GDLR on k-ary n-cube networks. The algorithm makes use of local information, the number of potential deadlocked packets, to estimate the network congestion. Based on this information, GDLR determines the probability of ports a packet may use. In this way, GDLR leads packets to pass by congested area and achieves a good performance under different patterns. A comparison of GDLR with other popular algorithms such as Dimension Order, Duato, GAL, has been given in terms of throughput and latency under various environments. The simulation results show that GDLR can balance the network load and improve the overall performance.

Since Section 2 gives some extensions to GDLR, a further study on them will be carried out in the near future. What is more, GDLR algorithm with injection control is another research topic in the future.

Acknowledgement. This research was supported in part by the Zhongxing Telecommunication Equipment Corporation (ZTE) Research Fund under Grant No. ZXJS200609120159 and by the National Science Foundation of China under Grant No.60532060. The authors also want to thank Guochang Kang and Shaolei Chen for their generous help.

References

1. Scott, S., Thorson, G.: The cray t3e network: adaptive routing in a high performance 3d torus. In: Proceedings of Hot Interconnects Symposium IV (August 1996)
2. Pfister, G.: An Introduction to the InfiniBand Architecture. IEEE Press, Los Alamitos (2001), http://www.infinibadta.org
3. Dally, W.J., Carvey, P., Dennison, L.: Architecture of the Avici terabit switch/router. In: Proceedings of Hot Interconnects Symposium VI, August 1998, pp. 41–50 (1998)

4. Sullivan, H., Bashkow, T.R.: A large scale, homogeneous, fully distributed parallel machine, I. In: Proc. of the International Symposium on Computer Architecture, pp. 105–117 (1977)
5. Duato, J., Yalamanchili, S., Ni, L.: Interconnection Networks, an Engineering Approach. Morgan-Kaufmann Press, San Francisco (2003)
6. Dally, W., Towles, B.: Principles and Practices of Interconnection Networks. Morgan-Kaufmann Press, San Francisco (2004)
7. Duato, J.: A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. IEEE Trans. on Parallel and Distributed Systems 4, 1320–1331 (1993)
8. Valiant, L.G.: A scheme for fast parallel communication. SIAM Journal on Computing 11(2), 350–361 (1982)
9. Singh, A., Dally, W.J., Towles, B., Gupta, A.K.: Globally adaptive load-balanced routing on tori. Computer Architecture Letters 3 (2004)
10. Singh, A., Dally, W.J., Gupta, A.K., Towles, B.: Adaptive Channel Queue Routing on k-ary n-cubes. In: SPAA. ACM Symposium on Parallelism in Algorithms and Architectures, Barcelona, Spain (June 2004)
11. Anjan, K.V., Pinkston, T.M., Duato, J.: Generalized theory for deadlock-free adaptive routing and its application to Disha Concurrent. In: Rolim, J.D.P. (ed.) Parallel and Distributed Processing. LNCS, vol. 1586, Springer, Heidelberg (1999)
12. Kim, J., Liu, Z., Chien, A.: Compressionless Routing: A Framework for Adaptive and Fault-Tolerant Routing. IEEE Trans. Parallel and Distributed Systems 8(3), 229–244 (1997)
13. OPNET Modeler documentation. OPNET Technologies, Inc. (2004), http:// www.opnet.com/
14. Kermani, P., Kleinrock, L.: Virtual Cut through: A new computer communication switching technique. Computer Networks 3, 34 (1979)
15. Ni, L., M., P., McKinley, K.: A Survey of Wormhole Routing Techniques in Directed Networks. Computer 26, 62–76 (1993)
16. Duato, J., Yalamanchili, S., Ni, L.: Interconnection Networks, an Engineering Approach. Morgan-Kaufmann Press, San Francisco (2003)
17. Dally, W., Towles, B.: Principles and Practices of Interconnection Networks. Morgan-Kaufmann Press, San Francisco (2004)
18. Newman, Internet Core Router Test (March 6, 2001), On the Web at http://www.lightreading.com

Optimal Semi-online Scheduling Algorithms on a Small Number of Machines

Yong Wu, Zhiyi Tan*, and Qifan Yang

Department of Mathematics, State Key Lab of CAD & CG,
Zhejiang University, Hangzhou 310027, P.R. China
tanzy@zju.edu.cn

Abstract. This paper considers semi-online scheduling problems on parallel identical machines with combined partial information. For the objective to minimize makespan, and both the largest processing time of all jobs and the total processing time of all jobs are known in advance, we present an optimal algorithm with competitive ratio 4/3 on three machines. For the objective to maximize the minimum machine load, and both the largest processing time of all jobs and the optimal value are known in advance, we present algorithms which are optimal when the machine number is less than 5.

1 Introduction

In recent years, semi-online scheduling problems have received increasing attention from the scheduling community. Different from classical online setting, some partial information about future jobs is known in advance before we construct a schedule. We can obtain algorithms with better performance than online algorithms. Some kinds of partial information that have been studied before are as follows:

sum: The total processing time of all jobs is known in advance [8].
opt: The optimal value of the instance is known in advance [2].
max: The largest processing time of all jobs is known in advance [7].

Furthermore, it is interesting to investigate whether the combination of two kinds of partial information is more "powerful". In other words, whether the performance of a semi-online algorithm can be even better if another kind of partial information is available [9]. The answer is not always positive. For example, the combination of *sum* and *opt* seems of little use.

In this paper, we will consider problems arise from the combination of above three kinds of partial information. We are given a machine set $\mathcal{M}$ of m parallel identical machines $M_1, M_2, \cdots, M_m$ and a job set $\mathcal{J}$ of n independent jobs. We identify jobs by their processing times as $p_j, j = 1, \cdots, n$. Machines and jobs are available at time zero, and no preemption is allowed. Two different objectives are considered. The first is to minimize the *makespan*. The second is to maximize

* Corresponding author. Supported by Natural Science Foundation of China (10671177, 60021201).

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 504–515, 2007.

the minimum machine load, where the *load* of a machine is defined as the total processing time of jobs assigned to it. We denoted them by $Pm||C_{max}$ and $Pm||C_{min}$, respectively.

We use the *competitive ratio* to measure the performance of an online (semi-online) algorithm. For any instance I and an algorithm A, let $C^A(I)$ (or shortly C^A) and $C^*(I)$ (or shortly C^*) denote the objective value of the schedule produced by A and optimal offline schedule, respectively. For the objective to minimize makespan, the competitive ratio is defined as the smallest number c such that for any I, $C^A(I) \leq cC^*(I)$. For the objective to maximize the minimum machine load, the competitive ratio is defined as the smallest number c such that for any I, $C^*(I) \leq cC^A(I)$. An online (semi-online) scheduling problem has a *lower bound* ρ if no online (semi-online) algorithm has a competitive ratio smaller than ρ. An online (semi-online) algorithm A is called *optimal* if its competitive ratio matches the lower bound of the problem.

Many results of related semi-online scheduling problems have been got in recent years(see Table 1). For the objective to minimize makespan, it seems hard to obtain optimal algorithms for $m > 2$ machines, even for the special case of $m = 3$. In this paper, we will present an optimal algorithm for $P3|sum\ \&\ max|C_{max}$, and it remains optimal for $P3|opt\ \&\ max|C_{max}$. For the objective to maximize the minimum machine load, it is surprising that the problem with optimal value is known in advance is much harder than online and other semi-online problems. In this paper, we will present optimal algorithms for $Pm|opt\ \&\ max|C_{min}$ when $2 \leq m \leq 5$.

Let $T = \sum_{j=1}^{n} p_j$ be the total processing time of all the jobs, and $p_{max} = \max_{j=1,\cdots,n} p_j$ be the largest processing time of all the jobs. Since we are assured that at least one job with processing time p_{max} will arrive sooner or later, it is

Table 1. Results of related semi-online problems ($*$: given in this paper)

		C_{max}		C_{min}	
		LB	UB	LB	UB
opt	$m = 2$	4/3[2]	4/3[2]	$2 - 1/m$[1]	$2 - 1/m$[1], 11/6[5]
	$m = 3, 4$		$\frac{5m-1}{3m+1}$, 1.625[2]		
	$m \geq 5$			43/24($m \to \infty$)[5]	
sum	$m = 2$	4/3[8]	4/3[8]	3/2[6]	3/2[6]
	$m \geq 3$	1.5($m \geq 6$)[4]	1.6[4]	$m - 1$[10]	$m - 1$[10]
max	$m = 2$	4/3[7]	4/3[7]	3/2[6]	3/2[6]
	$m \geq 3$	open	open	$m - 1$[10]	$m - 1$[10]
$sum\ \&\ max$	$m = 2$	6/5[9]	6/5[9]	5/4[10]	5/4[10]
	$m = 3$	4/3*	4/3*	3/2[10]	3/2[10]
	$m \geq 4$	open	$\sqrt{10/3}$[3] 1.6[4]	$m - 2$[10]	$m - 2$[10]
$opt\ \&\ max$	$m = 2$	6/5[9]	6/5[9]	5/4*	5/4*
	$m = 3$	4/3*	4/3*	$2 - \frac{1}{m-1}$*	$2 - \frac{1}{m-1}$*, 11/6[5]
	$m = 4, 5$	open	$\frac{5m-1}{3m+1}$, 1.625[2]		
	$m \geq 6$	open		43/24($m \to \infty$)[5]	

convenient to assume that p_1 be the first such job. When analyzing a semi-online algorithm A, let l_i^t be the current load of M_i right before the assignment of p_t, and L_i be the load of M_i after all the jobs have been assigned, $i = 1, \cdots, m$.

The paper is organized as follows. In section 2, we consider problems $P3|sum\ \&\ max|C_{max}$ and $P3|opt\ \&\ max|C_{max}$. In section 3, the problem $Pm|opt\ \&\ max|C_{min}$ is studied.

2 Minimizing Makespan

Theorem 1. *Any semi-online algorithm A for the problem* $P3|sum\ \&\ max|C_{max}$ *or* $P3|opt\ \&\ max|C_{max}$ *has a competitive ratio of at least* $\frac{4}{3}$.

Proof. We prove the lower bound by adversary method. Let $p_{max} = 6$ be known in advance. Moreover, $T = 18$ or $C^* = 6$ is also known in advance for the respective problem. The first two jobs are $p_1 = p_2 = 2$. If they are assigned to the different machines, let the last three jobs be $p_3 = p_4 = 6$, $p_5 = 2$. Since at least one job with processing time 6 will be assigned to the same machine as p_1 or p_2, we have $C^A \geq 8$. If p_1 and p_2 are assigned to the same machine, let $p_3 = 6$. If p_3 is assigned to the same machine as p_1 and p_2, let the last two jobs be $p_4 = 6$ and $p_5 = 2$ and we have $C^A \geq 10$. Otherwise, let $p_4 = p_5 = 4$, we have $C^A \geq 8$. Hence, $C^A/C^* \geq 4/3$. □

Next, we will present an algorithm for $P3|sum\ \&\ max|C_{max}$ with a competitive ratio of exactly $\frac{4}{3}$. Since the total processing time of all jobs is known in advance, we can normalize the processing time of all jobs be $T = \sum_{j=1}^{n} p_j = 18$. As $C^* \geq T/3 = 6$, it is sufficient to prove $C^A = \max\{L_1, L_2, L_3\} \leq 8$ as well as $C^A/C^* \leq 4/3$. The cases of $p_{max} \leq 3$ and $p_{max} \geq 5$ are relatively easy to deal with, and is omitted here due to lack of space. We focus on the case of $3 < p_{max} < 5$ in the following. A machine is called *small, middle, big* and *huge* if its load lies in the interval $[0, 2)$, $[2, 8 - p_{max}]$, $(8 - p_{max}, 5)$ and $[5, 8]$, respectively.

Assign p_1 to M_1. The assignment of remaining jobs consists of two stages. In the first stage, jobs are assigned to three machines according to the following *Preliminary Rules* until there is at least one job on both M_2 and M_3. Then we go to the second stage. Jobs are assigned by different *Final Rules* depending on the loads of three machines at the end of the first stage.

Preliminary Rule
While there is no job assigned to M_2 or M_3, and suppose $p_j, j \geq 2$ is the current job,

1. if $l_1^j + p_j \leq 8$, then assign p_j to M_1.
2. if $l_1^j + p_j > 8$ and $l_2^j + p_j \leq 8 - p_{max}$, then assign p_j to M_2.
3. if $l_1^j + p_j > 8$, $l_2^j + p_j > 8 - p_{max}$ and $l_3^j + p_j \leq 8$, then assign p_j to M_3.
4. if $l_1^j + p_j > 8$, $l_2^j + p_j > 8 - p_{max}$ and $l_3^j + p_j > 8$, then assign p_j to M_2.

Obviously, if all jobs have been assigned in the first stage, then $C^A \leq 8$.

Lemma 1. *If at the end of first stage, M_2 is a big machine, then*

(i) there is a single job with processing time greater than $8 - p_{max}$ scheduled on M_2.

(ii) if M_3 is also a big machine, there is a single job with processing time greater than $8 - p_{max}$ scheduled on M_3.

(iii) if M_3 is a huge machine, there are two jobs with processing time both greater than $8 - p_{max}$ scheduled on M_3.

Proof. Note that in the first stage, the load of M_2 will not be greater than $8 - p_{max}$ unless some job is assigned by Preliminary Rule 4. In other words, assign current job p_j to M_3 will cause the load of M_3 be greater than 8. Since $p_j \leq p_{max} < 8$, there is at least one job have already assigned to M_3 when p_j arrives. It follows that M_2 does not process any job at that time, or the first stage ended before p_j arrives. Secondly, $p_j > 8 - p_{max}$. Otherwise, M_2 can not be a big machine after p_j is assigned. Finally, jobs assigned to M_3 before p_j arrived are all greater than $8 - p_{max}$. Otherwise, it will be scheduled on M_2 by Preliminary Rule 2. Since $8-p_{max} < 5 < 2(8-p_{max})$ and $3(8-p_{max}) > 8$, if M_3 is a big machine, there is a single job scheduled on it. If M_3 is a huge machine, there are two jobs scheduled on it. □

Let the loads of three machines at the beginning of the second stage be l_i^0, $i = 1, 2, 3$. Reindex M_2 and M_3 such that $l_2^0 \leq l_3^0$. Obviously, we have $l_1^0 > 8 - p_{max}$. Otherwise, jobs will not be assigned to M_2 or M_3. Therefore, M_1 must be a big or huge machine. On the other hand, M_2 and M_3 can not be both huge machines, since $p_{max} \leq 5$ and at least one of M_2 and M_3 process only one job in the first stage. Table 2 lists all possible situation of the three machines at the beginning of the second stage. The last column of the table shows how to assign jobs in the second stage, which will be given in the rest of the section case by case.

Lemma 2. *Suppose after the assignment of some jobs, one of the following status happens. Then we can assign the remaining jobs so that $C^A \leq 8$.*

(i) There exist $i_1, i_2 \in \{1, 2, 3\}$, such that M_{i_1}, M_{i_2} are huge machines. Denote such status by (M_{i_1}, M_{i_2}, I);

Table 2. Different Final Rules depending on the loads of three machines at the end of the first stage

	M_1	M_2	M_3	Final Rule		M_1	M_2	M_3	Final Rule
1	big	small	small	Final Rule 1	10	huge	small	small	Final Rule 1
2	big	small	middle	Final Rule 1	11	huge	small	middle	(M_1, M_3, II)
3	big	small	big	Final Rule 1	12	huge	small	big	Final Rule 1
4	big	small	huge	Final Rule 2	13	huge	small	huge	(M_1, M_3, I)
5	big	middle	middle	(M_3, M_2, III)	14	huge	middle	middle	(M_1, M_2, II)
6	big	middle	big	Final Rule 1	15	huge	middle	large	(M_1, M_2, II)
7	big	middle	huge	(M_3, M_2, II)	16	huge	middle	huge	(M_1, M_3, I)
8	big	big	big	Final Rule 3	17	huge	big	big	Final Rule 3
9	big	big	huge	Final Rule 2	18	huge	big	huge	(M_1, M_3, I)

(ii) There exist $i_1, i_2 \in \{1,2,3\}$*, such that* M_{i_1} *is huge and* M_{i_2} *is middle, while the load of* M_{i_3}*,* $i_3 = \{1,2,3\}\backslash\{i_1,i_2\}$ *is less than* 5*. Denote such status by* (M_{i_1}, M_{i_2}, II)*;*

(iii) There exist $i_1, i_2 \in \{1,2,3\}$*, such that* M_{i_1}*,* M_{i_2} *are middle machines, while the load of* M_{i_3}*,* $i_3 = \{1,2,3\}\backslash\{i_1,i_2\}$ *is less than* 5*. Denote such status by* (M_{i_1}, M_{i_2}, III)*.*

Proof. (i) Assign all the remaining jobs to M_{i_3}, $i_3 = \{1,2,3\}\backslash\{i_1,i_2\}$. As $T = 18$ and M_{i_1}, M_{i_2} are huge machines, $L_{i_3} = T - L_{i_1} - L_{i_2} \leq 18 - 5 - 5 = 8$, $C^A = \max\{L_{i_1}, L_{i_2}, L_{i_3}\} \leq 8$.

(ii) Assign successive jobs to M_{i_3} until the new load of M_{i_3} will be greater than 5 for the first time, i.e. there exists job p_r, such that $l^r_{i_3} < 5$ and $l^r_{i_3} + p_r \geq 5$. If $l^r_{i_3} + p_r \leq 8$, assign p_r to M_{i_3} makes the schedule agree with status (M_{i_1}, M_{i_3}, I), assign remaining jobs according (i) can finish the proof. Otherwise, assign p_r to M_{i_2}. Since $3 < p_r \leq p_{max}$ and $2 \leq l^r_{i_2} \leq 8 - p_{max}$, we have $5 \leq l^r_{i_2} + p_r \leq 8$, which implies that the schedule agrees with status (M_{i_1}, M_{i_2}, I).

(iii) Assign successive jobs to M_{i_3} until the new load of M_{i_3} will be greater than 5 for the first time, i.e. there exists job p_r, such that $l^r_{i_3} < 5$ and $l^r_{i_3} + p_r \geq 5$. If further $l^r_{i_3} + p_r \leq 8$, assign p_r to M_{i_3} makes the schedule agree with (M_{i_3}, M_{i_1}, II). Otherwise, assign p_r to M_{i_2}. As $3 < p_r \leq p_{max}$ and $2 \leq l^r_{i_2} \leq 8 - p_{max}$, we have $5 \leq l^r_{i_2} + p_r \leq 8$, which implies that the schedule agrees with (M_{i_2}, M_{i_1}, II). □

Final Rule 1

By the description of Preliminary Rule 1, $l^0_1 + l^0_2 > 8$, or jobs assigned to M_2 will be assigned to M_1. Assign the successive jobs to M_3 until there exists job p_u, such that $l^u_3 < 5$ and $l^u_3 + p_u \geq 5$.

If $l^u_3 + p_u \leq 8$, then assign p_u to M_3 and M_3 becomes a huge machine. If further M_2 is a middle machine at the end of the first stage, the schedule agrees with (M_3, M_2, II). Otherwise, M_2 is a small machine and $l^0_2 \leq 2$. Assign all the remaining jobs to M_2 with total processing time less than $T-(l^u_3+p_u)-(l^0_1+l^0_2) \leq 18-5-8=5$. Hence, $L_1 = l^0_1 \leq 8$, $L_2 \leq 2+5 < 8$ and $L_3 = l^u_3 + p_u \leq 8$.

If $l^u_3 + p_u > 8$, then $p_u > 3$. Assign p_u to M_2 and all the remaining jobs to M_3, we have $L_1 = l^0_1 \leq 8$, $L_2 = l^0_2 + p_u \leq (8 - p_{max}) + p_{max} = 8$, and

$$\begin{aligned} L_3 &= T - L_1 - L_2 = T - l^0_1 - (l^0_2 + p_u) \\ &= T - (l^0_1 + l^0_2) - p_u \leq 18 - 8 - 3 = 7. \end{aligned}$$

Final Rule 2

By Lemma 1(iii), let the two jobs scheduled on the huge machine M_3 be p_{v_1} and p_{v_2}. We have $l^0_1 + p_{v_1} > 8$ and $p_{v_2} > 8 - p_{max}$. Assign all the remaining jobs to M_2, we have $L_1 = l^0_1 \leq 8$, $L_3 = l^0_3 \leq 8$, and

$$\begin{aligned} L_2 &= T - L_1 - L_3 = T - l^0_1 - (p_{v_1} + p_{v_2}) \\ &= T - (l^0_1 + p_{v_1}) - p_{v_2} \leq 18 - 8 - (8 - p_{max}) \leq 8. \end{aligned}$$

Final Rule 3

By Lemma 1(i)(ii), let the jobs scheduled on M_2 and M_3 be p_y and p_x, respectively. By Preliminary Rule 3, $p_x + p_y > 8$. Assign successive jobs to M_1 until

there exists a job p_z, such that $l_1^z \leq 8$ and $l_1^z + p_z > 8$. Assign p_z to M_2 and all the remaining jobs to M_3. Because $l_1^z + p_z > 8$ and $p_x + p_y > 8$, the total processing time of the remaining jobs is less than 2, which implies that $L_3 \leq 5 + 2 < 8$. If $L_2 = p_y + p_z \leq 8$, we have $C^A \leq 8$. Otherwise, $C^A = p_y + p_z > 8$. Consider the assignment of p_1, p_x, p_y, p_z in any optimal schedule. Obviously, there exists a machine process at least two of them. By $p_{max} \geq p_j,\ j = x, y, z$ and $p_x \geq p_y$, we have $C^* \geq p_y + p_z$ or $C^* \geq p_x + p_y$. For the former case, our assignment is optimal. For the latter case, assume $C^A/C^* > 4/3$, we have

$$p_y + p_z = C^A > \frac{4}{3}C^* = \frac{4}{3}(p_x + p_y) > \frac{4}{3} \cdot 8 = \frac{32}{3}.$$

Therefore, $T \geq p_{max} + p_x + p_y + p_z \geq 2(p_y + p_z) > \frac{64}{3} > 18$, which is a contradiction.

Hence, we can reach our desired result.

Theorem 2. *There exists a semi-online algorithm for $P3|sum\ \&\ max|C_{min}$ with a competitive ratio of $\frac{4}{3}$.*

Note that if C^* is known in advance, then $T \leq 3C^*$. Similarly as Theorem 2, we have the following theorem.

Theorem 3. *There exists a semi-online algorithm for $P3|opt\ \&\ max|C_{min}$ with a competitive ratio of $\frac{4}{3}$.*

3 Maximizing the Minimum Machine Load

In this section, we always assume that $p_{max} \leq C^*$ since we may replace all jobs with processing time greater than C^* by jobs with processing time C^*, which would not influence the optimal algorithm nor our algorithm, as we will see it later.

Call p_j as a *small* job if $p_j \leq \frac{2}{5}C^*$, otherwise we say that it is a *big* job.

Algorithm $OM2$

1. If $0 < p_{max} \leq \frac{2}{5}C^*$, assign all jobs by LS rule, i.e., always assign job to the machine which it can start process the job earlier.
2. If $\frac{4}{5}C^* \leq p_{max} \leq C^*$, assign p_1 to M_2, and assign other jobs to M_1.
3. If $\frac{2}{5}C^* < p_{max} < \frac{4}{5}C^*$, assign p_1 to M_2. Suppose that $p_j, j \geq 2$ is the current job.

(3.1) In case p_j is a small job,
  (3.1.1) if $l_2^j < \frac{4}{5}C^*$, assign p_j to M_2.
  (3.1.2) if $l_2^j \geq \frac{4}{5}C^*$, assign p_j to M_1.

(3.2) In case p_j is a big job,
  (3.2.1) if $l_1^j < \frac{4}{5}C^*$, assign p_j to M_1.
  (3.2.2) if $l_1^j \geq \frac{4}{5}C^*$, assign p_j to M_2.

Theorem 4. *The competitive ratio of the algorithm $OM2$ for $P2|opt\ \&\ max|C_{min}$ is $\frac{5}{4}$, and it is an optimal algorithm for $P2|opt\ \&\ max|C_{min}$.*

Algorithm OMm

1. If $0 < p_{max} \leq \frac{m}{m-1}\frac{m-2}{2m-3}C^*$, assign all jobs by LS rule.
2. If $\frac{m-1}{2m-3}C^* \leq p_{max} \leq C^*$.
   (2.1) Assign p_1 to M_m.
   (2.2) Suppose $p_j, j \geq 2$ is the current job, let

$$Z^j = \{M_i | l_i^j = 0,\ 1 \leq i \leq m-1\},$$
$$W^j = \{M_i | 0 < l_i^j \leq \frac{m-1}{2m-3}C^*,\ 1 \leq i \leq m-1\}.$$

   (2.2.1) If $Z^j = \emptyset$, assign p_j to the machine with smallest load in $\mathcal{M}\backslash\{M_m\}$.
   (2.2.2) If $Z^j \neq \emptyset$ and $p_j \geq \frac{m-1}{2m-3}C^*$, assign p_j to an arbitrary machine in Z^j.
   (2.2.3) If $Z^j \neq \emptyset$, $p_j < \frac{m-1}{2m-3}C^*$ and $W^j \neq \emptyset$, assign p_j to an arbitrary machine in W^j.
   (2.2.4) If $Z^j \neq \emptyset$, $p_j < \frac{m-1}{2m-3}C^*$ and $W^j = \emptyset$, assign p_j to an arbitrary machine in Z^j.
3. If $\frac{m}{m-1}\frac{m-2}{2m-3}C^* < p_{max} < \frac{m-1}{2m-3}C^*$.
   (3.1) Assign p_1 to M_m.
   (3.2) Suppose $p_j, j \geq 2$ is the current job, let

$$U^j = \{M_i | l_i^j < \frac{m-1}{2m-3}C^*, l_i^j + p_j \leq C^*,\ 1 \leq i \leq m-1\},$$
$$V^j = \{M_i | l_i^j \leq p_{max},\ 1 \leq i \leq m-1\}.$$

   (3.2.1) If $U^j \neq \emptyset$, assign p_j to the machine in U^j with the smallest index.
   (3.2.2) If $U^j = \emptyset$, $V^j \neq \emptyset$, assign p_j to the machine in V^j with the largest index.
   (3.2.3) If $U^j = V^j = \emptyset$, assign p_j by LS rule.

Theorem 5. *The competitive ratio of the algorithm OMm for $Pm|opt\ \&\ max|C_{min}$, $m \geq 3$ is at most $2 - \frac{1}{m-1}$.*

Proof. For $0 < p_{max} \leq \frac{m}{m-1} \cdot \frac{m-2}{2m-3}C^*$, since all jobs are assigned by LS rule, we have

$$|L_i - L_k| \leq p_{max},\ 1 \leq i, k \leq m,$$
$$L_i \leq C^{OMm} + p_{max},\ 1 \leq i \leq m,$$
$$C^* \leq \frac{T}{m} = \frac{\sum_{i=1}^m L_i}{m} \leq \frac{(m-1)(C^{OMm} + p_{max}) + C^{OMm}}{m} = C^{OMm} + \frac{m-1}{m}p_{max},$$
$$C^{OMm} \geq C^* - \frac{m-1}{m}p_{max} \geq C^* - \frac{m-1}{m} \cdot \frac{m}{m-1} \cdot \frac{m-2}{2m-3}C^* = \frac{m-1}{2m-3}C^*.$$

For $\frac{m-1}{2m-3}C^* \leq p_{max} \leq C^*$, it is obviously that $L_m = p_1 \geq \frac{m-1}{2m-3}C^*$. Next we will prove that $L_i \geq \frac{m-1}{2m-3}C^*,\ 1 \leq i \leq m-1$. Suppose that there exists $i_0,\ 1 \leq i_0 \leq m-1$, such that $L_{i_0} < \frac{m-1}{2m-3}C^*$.

Lemma 3. *If* $p_g > \frac{m-1}{2m-3}C^*$, *then* $Z^g \neq \emptyset$.

Proof. According to the algorithm, $|W^j| \leq 1$ for all j, $1 \leq j \leq n$. Suppose that $p_g > \frac{m-1}{2m-3}C^*$ and $Z^g = \emptyset$, then $0 < l^g_{i_0} \leq L_{i_0} < \frac{m-1}{2m-3}C^*$ and M_{i_0} is the unique machine in W^g. Thus the load of each machine in $\mathcal{M}\backslash\{M_m\}$ is greater than $\frac{m-1}{2m-3}C^*$ except M_{i_0}. Hence p_g will be assigned to M_{i_0} which results $L_{i_0} \geq p_g > \frac{m-1}{2m-3}C^*$, a contradiction. □

By Lemma 3, all jobs with processing time greater than $\frac{m-1}{2m-3}C^*$ are assigned to empty machines. Since the loads of those machines are greater than L_{i_0}, no more jobs will be assigned to those machines. The final loads of those machines are all less than $p_{max} \leq C^* \leq \frac{2m-2}{2m-3}C^*$. For machines which process at least two jobs, no more jobs will be assigned to it after its load greater than $\frac{m-1}{2m-3}C^*$ considering the existence of M_{i_0}. Therefore, the final loads of these machines are also less than $\frac{m-1}{2m-3}C^* + \frac{m-1}{2m-3}C^* = \frac{2m-2}{2m-3}C^*$. Finally, we have

$$C^{OMm} = L_{i_0} \geq \sum_{i=1}^{m-1} L_i - (m-1-1)\frac{2m-2}{2m-3}C^* \geq T - L_m - (m-2)\frac{2m-2}{2m-3}C^*$$
$$\geq (m-1)C^* - (m-2)\cdot\frac{2m-2}{2m-3}C^* = \frac{m-1}{2m-3}C^*$$

which violates our assumption.

In the rest of the proof, we concentrate on the case of $\frac{m}{m-1}\cdot\frac{m-2}{2m-3}C^* < p_{max} < \frac{m-1}{2m-3}C^*$. From the description of Step (3.2), we conclude that if $M_i \notin U^j \cup V^j$ and p_j is still assigned to M_i, $i = 1, \cdots, m$, we must have

$$U^j = V^j = \emptyset,\ l^j_i = \min_{1\leq k\leq m} l^j_k. \tag{1}$$

Lemma 4. *If there exists* i_0, $1 \leq i_0 \leq m$, *such that* $L_{i_0} < \frac{m-1}{2m-3}C^*$, *then* $L_i > C^*$ *for all* $1 \leq i \leq m$ *and* $i \neq i_0$.

Proof. Suppose that there exists $k, 1 \leq k \leq m$ and $k \neq i_0$, such that $L_k < C^*$. Then we have

$$\sum_{i=1}^{m} L_i - L_k - L_{i_0} = T - L_k - L_{i_0} > mC^* - C^* - \frac{m-1}{2m-3}C^* = (m-2)\frac{2(m-1)}{2m-3}C^*.$$

Therefore, there exists at least one machine, say M_e, satisfying $L_e > \frac{2(m-1)}{2m-3}C^*$. Denoted by p_f the job which makes the load of M_e be greater than $\frac{2(m-1)}{2m-3}C^*$ for the first time, i.e., $l^f_e \leq \frac{2(m-1)}{2m-3}C^*$, $l^f_e + p_f > \frac{2(m-1)}{2m-3}C^*$. Since

$$l^f_e > \frac{2(m-1)}{2m-3}C^* - p_f > \frac{2(m-1)}{2m-3}C^* - p_{max} > \frac{m-1}{2m-3}C^* > p_{max},$$

we have $M_e \notin U^f \cup V^f$. Together with $l^f_{i_0} \leq L_{i_0} < \frac{m-1}{2m-3}C^* \leq l^f_e$, p_f can not be assigned to M_e, which contradicts to the definition of p_f. Hence, we have $L_i > C^*$ for all $1 \leq i \leq m$ and $i \neq i_0$. □

Lemma 5. *If there exists* $i_0,\ 1 \le i_0 \le m-1$, *such that* $L_{i_0} < \frac{m-1}{2m-3}C^*$, *then* $L_i > p_{max}$ *for all* $1 \le i \le m$.

Proof. By Lemma 4, we have

$$L_m > C^* > p_{max}. \tag{2}$$

Hence, there exists some job assigned to M_m together with p_1. Denote one of such jobs by p_d. By (1), p_d will not be assigned to M_m unless $U^d = V^d = \emptyset$, then we have $L_i \ge l_i^d > p_{max},\ 1 \le i \le m-1$. Together with (2), the lemma is thus proved. □

Now we are going to prove $C^{OMm} \ge \frac{m-1}{2m-3}C^*$ by contradiction. Assume $C^{OMm} < \frac{m-1}{2m-3}C^*$. Denote by p_{j_i} the last job assigned to $M_i,\ 1 \le i \le m-1$, respectively. We distinguish three cases according to which machine is the least loaded one.

Case 1. $C^{OMm} = L_m < \frac{m-1}{2m-3}C^*$
Obviously, $L_m \ge p_{max}$ since p_1 is always assigned to M_m. Moreover, by Lemma 4, we have $L_i > C^*,\ 1 \le i \le m-1$. We first show that

$$L_i - p_{j_i} = l_i^{j_i} \le L_m,\ 1 \le i \le m-1. \tag{3}$$

Since $l_i^{j_i} + p_{j_i} = L_i > C^*$, we have

$$M_i \notin U^{j_i},\ 1 \le i \le m-1. \tag{4}$$

If $M_i \in V^{j_i}$, by the definition of V^{j_i}, we have $l_i^{j_i} \le p_{max} \le L_m$. If $M_i \notin V^{j_i}$, together with (4), we have $l_i^{j_i} = \min\limits_{1 \le k \le m} l_k^{j_i} \le l_m^{j_i} \le L_m$ due to (1). (3) is thus proved.

Next, we will show that there is only one job assigned to M_i except $p_{j_i},\ 2 \le i \le m-1$. Suppose that for some $h,\ 2 \le h \le m-1$, there are at least two jobs assigned to M_h before the assignment of p_{j_h} with total processing time $l_h^{j_h} \le L_m < \frac{m-1}{2m-3}C^*$, then at least one job with processing time less than $\frac{m-1}{2(2m-3)}C^*$ exists. Let p_{q_h} be the first such job, i.e. $p_{q_h} < \frac{m-1}{2(2m-3)}C^*$. We distinguish two subcases to derive contradiction.

Subcase 1. p_{j_1} **comes before** p_{q_h}
By (4), $M_1 \notin U^{j_1}$. Since p_{j_1} is still assigned to M_1, we have $U^{j_1} = \emptyset$ and $V^{j_1} \subseteq \{M_1\}$. Thus $M_h \notin U^{j_1} \cup V^{j_1}$ and hence, $l_h^{j_1} > p_{max}$. As p_{j_h} is the last job assigned to M_h, p_{q_h} comes before p_{j_h}. Together with the fact that p_{j_1} comes before p_{q_h}, we have $l_h^{j_h} \ge l_h^{j_1} > p_{max}$, i.e., $M_h \notin V^{j_h}$. Note that $M_h \notin U^{j_h}$ by (4), we have $l_m^{j_h} \ge \min\limits_{1 \le k \le m} l_k^{j_h} = l_h^{j_h} > p_{max}$ by (1). Hence there exists some job, denoted by p_{j_m}, assigned to M_m before the assignment of p_{j_h}. But this also causes contradiction. In fact, as $p_{j_m} \le L_m - p_1 < \frac{m-1}{2m-3}C^* - p_{max} < \frac{1}{(m-1)(2m-3)}C^*$, together with (3) we have $l_h^{j_m} \le l_h^{j_h} \le L_m < \frac{m-1}{2m-3}C^*$ and $l_h^{j_m} + p_{j_m} < \frac{m-1}{2m-3}C^* + \frac{1}{(m-1)(2m-3)}C^* < C^*$. It follows that $M_h \in U^{j_m}$, which implies that p_{j_m} can not be assigned to M_m.

Subcase 2. p_{q_h} **comes before** p_{j_1}
In this case, we have $l_1^{q_h} \leq l_1^{j_1} \leq L_m < \frac{m-1}{2m-3}C^*$ and $l_1^{q_h} + p_{q_h} \leq \frac{m-1}{2m-3}C^* + \frac{m-1}{2(2m-3)}C^* \leq C^*$. It follows that $M_1 \in U^{q_h}$ and p_{q_h} should be assigned to M_1, a contradiction.

Combining with above discussions, we know that there are only two jobs on M_i, $2 \leq i \leq m-1$, one is p_{j_i} and denote the other one by p_{k_i}. Therefore,

$$L_i = p_{k_i} + p_{j_i} > C^*,\ 2 \leq i \leq m-1. \tag{5}$$

Consider the assignment of a total amount of $2m-2$ jobs of p_{j_i}, $1 \leq i \leq m-1$, p_{k_i}, $2 \leq i \leq m-1$ and p_1 in any optimal schedule, there must exist two of m machines in $\mathcal{M}$ on which totally process no more than two of them. Note that $m \geq 3$, together with (3) and (5), we have

$$\begin{aligned} C^* &\leq \frac{1}{m-(m-2)}\left(T - \left(\sum_{i=1}^{m-1} p_{j_i} + \sum_{i=2}^{m-1} p_{k_i} + p_1 - 2p_1\right)\right) \\ &= \frac{1}{2}\left((L_1 - p_{j_1}) + \sum_{i=2}^{m-1}(L_i - (p_{j_i} + p_{k_i})) + L_m + p_1\right) = \frac{1}{2}(l_1^{j_1} + L_m + p_1) \\ &< \frac{1}{2}\left(\frac{m-1}{2m-3}C^* + \frac{m-1}{2m-3}C^* + \frac{m-1}{2m-3}C^*\right) \leq C^*, \end{aligned}$$

which is a contradiction.

Case 2. $C^{OMm} = L_i < \frac{m-1}{2m-3}C^*$, for some i, $2 \leq i \leq m-1$
By Lemma 5, we have $L_i > p_{max}$. Therefore, there must be at least two jobs assigned to M_i with total processing time less than $\frac{m-1}{2m-3}C^*$. Similarly to Case 1, denoted by p_{q_i} the first job assigned to M_i with $p_{q_i} < \frac{m-1}{2(2m-3)}C^*$. On the other hand, by 4, we have $L_1 = l_1^{j_1} + p_{j_1} > C^*$ and thus $M_1 \notin U^{j_1}$. Since p_{j_1} is assigned to M_1, we have $U^{j_1} = \emptyset$ and $V^{j_1} \subseteq \{M_1\}$. If $V^{j_1} = \emptyset$, then $l_1^{j_1} \leq l_i^{j_1} \leq L_i < \frac{m-1}{2m-3}C^*$ by (1). If $V^{j_1} = \{M_1\}$, then $l_1^{j_1} \leq p_{max} \leq L_i < \frac{m-1}{2m-3}C^*$ by the definition of V^{j_1}. For both cases, $l_1^{j_1} < \frac{m-1}{2m-3}C^*$.

Subcase 1. p_{j_1} **comes before** p_{q_i}
Since $l_i^{j_1} < \frac{m-1}{2m-3}C^*$, we know that there are at least two jobs with total processing time less than $\frac{m-1}{2m-3}C^*$ assigned to M_i before the assignment of p_{j_1}. Therefore, at least one of them with processing time less than $\frac{m-1}{2(2m-3)}C^*$ exists, which contradicts to the fact that p_{q_i} is the first job assigned to M_i with processing time less than $\frac{m-1}{2m-3}C^*$.

Subcase 2. p_{q_i} **comes before** p_{j_1}
By $l_1^{q_i} \leq l_1^{j_1} < \frac{m-1}{2m-3}C^*$ and $l_1^{q_i} + p_{q_i} \leq l_1^{j_1} + p_{q_i} \leq \frac{m-1}{2m-3}C^* + \frac{m-1}{2(2m-3)}C^* < C^*$, we have $M_1 \in U^{q_i}$ and p_{q_i} must be assigned to M_1, which is a contradiction.

Case 3. $C^{OMm} = L_1 < \frac{m-1}{2m-3}C^*$
By Lemma 5, $L_i > p_{max}$, $1 \leq i \leq m$. Hence at least two jobs are assigned to each machine. Since the load of M_1 is always less than $\frac{m-1}{2m-3}C^*$. The processing

time of each job assigned to M_i, $2 \le i \le m-1$ and M_m are all greater than $\frac{m-2}{2m-3}C^*$, or they will be assigned to M_1 by Step (3.2.1). On the other hand, there are only two jobs p_{k_i} and p_{j_i} assigned to M_i. Suppose p_{t_i} is the third job assigned to M_i, then $l_i^{t_i} > 2 \cdot \frac{m-2}{2m-3}C^* > p_{max}$, $l_i^{t_i} + p_{t_i} > 3 \cdot \frac{m-2}{2m-3}C^* \ge C^*$ i.e. $M_i \notin U^{t_i} \cup V^{t_i}$. Since $l_i^{t_i} > \frac{m-1}{2m-3}C^* > L_1 \ge l_1^{t_i}$, p_{t_i} can not be assigned to M_i. We can also obtain that there is only one job, denoted by p_q, assigned to M_m besides p_1. In fact, if $p_{q'}$ is the third job assigned to M_m, then

$$\begin{aligned} l_m^{q'} &> \frac{m-2}{2m-3}C^* + p_1 > \frac{m-2}{2m-3}C^* + \frac{m(m-2)}{(m-1)(2m-3)}C^* \\ &= \frac{(2m-1)(m-2)}{(m-1)(2m-3)}C^* > \frac{(2m-1)(m-2)}{(m-1)^2}L_1 > L_1 \ge l_1^{q'}, \end{aligned}$$

which implies that $p_{q'}$ can not be assigned to M_m.

Similarly to the Case 1, we have

$$L_i = p_{k_i} + p_{j_i} > C^*,\ 2 \le i \le m-1, \tag{6}$$

$$L_m = p_q + p_1 > C^*. \tag{7}$$

Consider the assignment of $2m-2$ jobs of p_{j_i}, $2 \le i \le m-1$, p_{k_i}, $2 \le i \le m-1$, p_1 and p_q in any optimal schedule. Combining with (6) and (7), we have

$$\begin{aligned} C^* &\le \frac{1}{m-(m-2)}\left(T - \left(\sum_{i=2}^{m-1} p_{j_i} + \sum_{i=2}^{m-1} p_{k_i} + p_q + p_1 - 2p_1\right)\right) \\ &= \frac{1}{2}\left(L_1 + \sum_{i=2}^{m-1}(L_i - (p_{j_i} + p_{k_i})) + (L_m - (p_q + p_1)) + 2p_1\right) \\ &= \frac{1}{2}(L_1 + 2p_1) < \frac{1}{2}\left(\frac{m-1}{2m-3}C^* + \frac{2(m-1)}{2m-3}C^*\right) \le C^*, \end{aligned}$$

which is a contradiction.

Lemma 6. *Let r be the lower bound for the problem $Pm'|opt|C_{min}$ with $m' \ge 2$, then r is also a lower bound for the problem $Pm|opt$ & $max|C_{min}$, where $m = m' + 1$.*

Proof. Assume that r is a lower bound for the problem $Pm'|opt|C_{min}$ with a fixed $m' \ge 2$. We convert it into a lower bound for the problem $Pm|opt$ & $max|C_{min}$. Let $p_{max} = C^*$ be known in advance. We begin the job sequences with $p_1 = C^*$, and then continue with the original job sequences used for $Pm'|opt|C_{min}$. The optimal algorithm assign p_1 to one machine separately and the optimal value is unchanged. To make the objective value produced by semi-online algorithm A be larger, A also prefer to assign p_1 to one machine and do not use this machine later. The objective value of schedule produced by A will not decrease though one more machine is available. Thus the competitive ratio of any algorithm for the problem $Pm|opt$ & $max|C_{min}$ is at least r. □

By Lemma 6 and the lower bounds of $Pm|opt|C_{min}$ [1], any semi-online algorithm A for the problem $Pm|opt\ \&\ max|C_{min}$ has a competitive ratio of at least

$$\begin{cases} 3/2 & \text{if } m = 3, \\ 5/3 & \text{if } m = 4, \\ 7/4 & \text{if } m = 5. \end{cases}$$

Hence, OMm is an optimal algorithm for $Pm|opt\ \&\ max|C_{min}$ when $m = 3, 4, 5$.

References

1. Azar, Y., Epstein, L.: On-line machine covering. Journal of Scheduling 1, 67–77 (1998)
2. Azar, Y., Regev, O.: On-line bin-stretching. Theoretical Computer Science 168, 17–41 (2001)
3. Chang, S.Y., Hwang, H.C., Park, J.: Semi-on-line parallel machines scheduling under known total and largest processing times. Journal of the Operations Research Society of Japan 48, 1–8 (2005)
4. Cheng, T.C.E., Kellerer, H., Kotov, V.: Semi-on-line multiprocessor scheduling with given total processing time. Theoretical Computer Science 337, 134–146 (2005)
5. Ebenlendr, T., Noga, J., Sgall, J., Woeginger, G.: A note on semi-online machine covering. In: Erlebach, T., Persinao, G. (eds.) WAOA 2005. LNCS, vol. 3879, pp. 110–118. Springer, Heidelberg (2006)
6. He, Y.: Semi on-line scheduling problem for maximizing the minimum machine completion time. Acta Mathematica Applicatae Sinica 17, 107–113 (2001)
7. He, Y., Zhang, G.C.: Semi on-line scheduling on two identical machines. Computing 62, 179–187 (1999)
8. Kellerer, H., Kotov, V., Speranza, M.R., Tuza, Z.: Semi on-line algorithms for the partition problem. Operations Research Letters 21, 235–242 (1997)
9. Tan, Z.Y., He, Y.: Semi-on-line problems on two identical machines with combined partial information. Operations Research Letters 30, 408–414 (2002)
10. Tan, Z.Y., Wu, Y.: Optimal semi-online algorithms for machine covering. Theoretical Computer Science 372, 69–80 (2007)

Lower Bounds on Edge Searching

Brian Alspach[1], Danny Dyer[2], Denis Hanson[1], and Boting Yang[3]

[1] Department of Mathematics and Statistics, University of Regina
{alspach,dhanson}@math.uregina.ca
[2] Department of Mathematics and Statistics, Memorial University of Newfoundland
dyer@math.mun.ca
[3] Department of Computer Science, University of Regina
boting@cs.uregina.ca

Abstract. Searching a network for intruders is an interesting and difficult problem. Edge-searching is one such search model, in which intruders may exist anywhere along an edge. Since finding the minimum number of searchers necessary to search a graph is NP–complete, it is natural to look for bounds on the search number. We show lower bounds on the search number using minimum degree, girth, chromatic number, and colouring number.

1 Introduction

Clearing a graph (or network) of an intruder or intruders has a natural division into two classes of problem: those in which intruders may be located only at vertices and those in which intruders may be located at vertices or anywhere along edges. The latter situation is called *edge-searching*. Searching graphs serves as a model for important applied problems (see [3], [4] and [6]). A survey of results can be found in [1].

In this paper, we adopt the convention that multigraphs allow multiple edges, reflexive graphs allow loops, and that graphs allow neither loops nor multiple edges. A reflexive multigraph allows both loops and multiple edges. The specifics of searching a reflexive multigraph G are as follows. Initially, all edges of G are *contaminated*. To search G it is necessary to formulate and carry out a search strategy. A strategy is a sequence of actions performed as consecutive steps designed so that after the final step, all edges of G are *uncontaminated* (or *cleared*). Only three actions are allowed at each step.

1. Place a searcher on a vertex.
2. Move a searcher on a vertex u along an edge uv to v.
3. Remove a searcher from a vertex.

An edge uv in G can be *cleared* in one of two ways. Either at least two searchers are located on vertex u of edge uv, and one of them traverses the edge from u to v while the others remain at u, or at least one searcher is located on vertex u, where all edges incident with u, other than uv, are already cleared. Then the searcher moves from u to v. A cleared edge may become *recontaminated*. This

B. Chen, M. Paterson, and G. Zhang (Eds.): ESCAPE 2007, LNCS 4614, pp. 516–527, 2007.

happens if, at any time, there is a path from an endpoint of the cleared edge to an endpoint of a contaminated edge that does not contain a searcher. We say a vertex is cleared when all edges incident with it are cleared.

Knowing that our goal is a cleared graph, one in which all the edges are cleared, a basic question is: what is the fewest number of searchers for which a search strategy exists? We call this the *search number*, denoted $\mathrm{s}(G)$.

Let $E(i)$ be the set of cleared edges after action i has occurred. A search strategy for a graph G for which $E(i) \subseteq E(i+1)$ for all i is said to be *monotonic*. We may then define the *monotonic search number*, denoted $\mathrm{ms}(G)$. LaPaugh [7] and Bienstock and Seymour [2] proved that for any connected graph G, $\mathrm{s}(G) = \mathrm{ms}(G)$. We will only consider connected graphs throughout this paper.

In general, determining the search number of a graph G is NP-complete [8]. As any successful search strategy gives an upper bound, our goal becomes first to find the "right" way to clear the graph, using as few searchers as possible. Once this strategy is found, we must then prove that no fewer searchers will suffice. Here is where the true difficulty lies: most easily attainable lower bounds are quite poor. We will prove several lower bound results using the graph parameters of minimum degree, girth, and chromatic number.

The following three theorems (see [11]) give lower bounds for the search number of a graph G. The first uses the minimum degree $\delta(G)$, while the second uses the clique number $\omega(G)$ of G, the order of a maximum order complete subgraph.

Theorem 1. *If G is a connected graph then $\mathrm{s}(G) \geqq \delta(G)$. If $\delta(G) \geqq 3$, then $\mathrm{s}(G) \geqq \delta(G) + 1$.*

Theorem 2. *If G is a connected graph and $\omega(G) \geqq 4$, then $\omega(G) \leqq \mathrm{s}(G)$.*

We also recall a well-known theorem on searching. In this paper, if H is minor of G, we write $H \preceq G$, and if H is a subgraph of G, we write $H \subseteq G$.

Theorem 3. *If $H \preceq G$, then $\mathrm{s}(H) \leqq \mathrm{s}(G)$.*

2 Minimum Degree and Girth

Consider the graph $K_{3,3}$. By Theorem 1, four searchers are necessary. However, with four searchers it is impossible to clear more than two vertices! We introduce the idea of girth to expand our repertoire of lower bounds.

Since the search number of a connected graph is equal to its monotonic search number, we may investigate monotonic search strategies instead. Being able to assume a search is monotonic is very useful. Moreover, Theorem 4 tells us something about how such a search strategy may be formulated. However, we must first introduce the following lemma. A vertex in a graph G is said to be *exposed* if it has edges incident with it that are contaminated as well as edges incident with it that are cleared. Following a search strategy S on G, we define $\mathrm{ex}_S(G, i)$ to be the number of exposed vertices after the i-th step. We also define the *maximum number of exposed vertices* to be $\mathrm{mex}_S(G) = \max_i \mathrm{ex}_S(G, i)$.

Lemma 1. *If G is a connected reflexive multigraph, then for any monotonic search strategy S using* $\mathrm{ms}(G)$ *searchers,* $\mathrm{mex}_S(G) \leqq \mathrm{ms}(G) \leqq \mathrm{mex}_S(G) + 1$.

Proof. The first inequality is straightforward; every exposed vertex must contain a searcher, so there cannot be more exposed vertices than searchers.

For the second inequality, it suffices to show that if there is a monotonic search strategy S which clears G using k searchers, and if $\mathrm{mex}_S(G) \leqq k-2$, then we can formulate another monotonic search strategy T which clears G using only $k-1$ searchers, and $\mathrm{mex}_T(G) = \mathrm{mex}_S(G)$.

Let $\mathbb{S}$ denote the set of all search strategies for G that use k searchers. For a given search strategy $S \in \mathbb{S}$, label the searchers in the order that they are first placed in G. The first unlabelled searcher placed will be labelled γ_1, the second unlabelled searcher placed will be labelled γ_2, and so on. Certainly, if any searcher is not placed on the graph, then it may be removed from S without affecting the search strategy. Thus, we may assume that every searcher is labelled, and every searcher is at some point placed on the graph. We also shall index each action using successive positive integers starting with 1, 2, ...

Whenever a vertex v is exposed, then there must be at least one searcher on v in order to prevent recontamination of the edges incident with v. If there is more than one searcher located at v, then we arbitrarily designate one of the searchers as the *guard* for v. Of course, if there is only one searcher located at v, then that searcher automatically is designated as the guard. We shall call a searcher *important* if the searcher at some point either clears an edge or becomes the guard on an exposed vertex.

Now consider the searcher γ_k. We first want to show that there is a strategy in $\mathbb{S}$ for which γ_k is not important. To this end, we assume that $\mathbb{S}$ contains at least one strategy for which γ_k is important. For any strategy S for which γ_k is important, let $L(S)$ denote the index of the last action in S and let $k(S)$ denote the index of the action that results in γ_k becoming important. In other words, either action $k(S)$ is the first time γ_k clears an edge, or action $k(S)$ results in γ_k first being a guard at some vertex.

Over all strategies $S \in \mathbb{S}$ for which γ_k is important, let $L(S) - k(S)$ be minimum. Suppose $L(S) - k(S) > 0$.

Consider the case that γ_k becomes important because action $k(S)$ consists of γ_k clearing an edge uv by traversing the edge from u to v. Define a new strategy S' as follows. We let S' coincide with S for actions with indices $1, 2, \ldots, k(S)-1$. At this point, γ_k is located at vertex u and in strategy S should move along uv from u to v in order to clear it. Because $\mathrm{mex}_S(G) \leqq k-2$, there is at least one searcher $\gamma_i \neq \gamma_k$ who is not being used to protect an exposed vertex. If γ_i also is located on vertex u, then let γ_i traverse the edge uv from u to v as the action indexed with $k(S)$ in S'. From this point on, S' is the same as S except that the roles of γ_i and γ_k are interchanged. If γ_i is not located on vertex u, then remove γ_i from its current vertex as the action indexed by $k(S)$ in S'. Now place γ_i on u as the action of S' indexed $k(S)+1$. Remove γ_k from u as the next action of S' and replace γ_k on the vertex from which γ_i just came. These four actions have interchanged the locations of γ_i and γ_k. From this point on, the action indexed

t in S' is the same as the action indexed $t-4$ in S with the roles of γ_i and γ_k interchanged.

In both subcases for which γ_k became important, it is easy to see that $L(S')-k(S') < L(S)-k(S)$. Now consider the remaining case that γ_k becomes important because following action $k(S)$, γ_k is a guard on an exposed vertex u. If there are two or more searchers on u following action $k(S)$, designate a searcher γ_i other than γ_k to be the guard. Then interchange the roles of γ_i and γ_k from that point on. The resulting strategy S' certainly satisfies $L(S') - k(S') < L(S) - k(S)$. Hence, just before the action $k(S)$ is carried out, there is just one searcher on u, in addition to γ_k, and this searcher leaves u on action $k(S)$. Because $\mathrm{mex}_S(G) \leqq k-2$, there is another non-guard searcher γ_j located at some vertex $v \neq u$. Take four steps to interchange γ_k and γ_j and then define S' to be the initial part of S, the four actions just concluded, and the completion of S with the roles of γ_j and γ_k interchanged. Again it is easy to see that $L(S')-k(S') < L(S)-k(S)$.

Therefore, if there exists a search strategy S with γ_k being important and $L(S) - k(S) > 0$, there must be such a search strategy with $L(S) = k(S)$. Suppose S is such a strategy with $L(S) = k(S)$. This means that γ_k becomes important only on the last action of S, and this action must complete clearing the graph. Hence, on the last action $L(S)$, γ_k traverses an edge uv from u to v clearing the last contaminated edge of G. Because γ_k is not important and u is incident with a contaminated edge, there must be another searcher on u acting as the guard. Have this searcher clear the edge uv instead of γ_k. This results in a strategy for which γ_k is not important.

From this strategy, form a new strategy T which is exactly the same, but with all of γ_k's actions are removed. Since γ_k is unimportant, every edge is still cleared, and the maximum number of exposed vertices is the same, but only $k-1$ searchers are used.

Theorem 4. *If G is a connected reflexive graph with no vertices of degree 2, then there exists a monotonic search S with $\mathrm{ms}(G)$ searchers such that $\mathrm{ms}(G) = \mathrm{mex}_S(G) + 1$.*

Proof. Let G be a connected reflexive graph G with no vertices of degree 2. Assume that for every monotonic search strategy S on G, $\mathrm{mex}_S(G) = \mathrm{ms}(G) = k$. Since S is a search strategy, there is a moment when the number of exposed vertices becomes $\mathrm{mex}_S(G)$ for the last time. Let S' be a monotonic search strategy which has the minimum number of instances where the number of exposed vertices goes from being less than k to being k and has the minimum number of edge clearings after the last time the number of exposed vertices becomes k. The only action which can increase the number of exposed vertices is clearing an edge, which can expose at most two additional vertices. Let xy be the last edge cleared before the number of exposed vertices becomes $\mathrm{mex}_S(G)$ for the last time. We consider four cases as to how xy can be cleared.

Case 1: The edge xy is a loop, with $x = y$. Since clearing xy can expose at most one additional vertex, the number of exposed vertices must be $k-1$. If x was already exposed, clearing xy would not increase the number of exposed

vertices. Thus, x must not have been an exposed vertex. But since there must be a searcher on each of the $k-1$ exposed vertices, this leaves only one searcher to clear the loop xy. But a single searcher cannot clear a loop, a contradiction. Thus, xy cannot be a loop.

Case 2: The number of exposed vertices just before xy is cleared is $k-2$, and at this time neither x nor y is exposed. Label the $k-2$ exposed vertices as $v_1, v_2, \ldots v_{k-2}$, and assume that searcher γ_i rests on vertex v_i, $1 \leqq i \leqq k-2$. The edge xy must be such that neither x nor y is some v_i. Assume that after xy is cleared, searcher γ_{k-1} is on x and γ_k is on y.

If there are any pendant edges or loops attached to some v_i that are not cleared, we can use searcher γ_k to clear these edges first. If this reduces the number of exposed vertices, then at some later action k vertices must be exposed because the number of exposed vertices increasing to k occurs a minimum number of times in S'. This later point must have more cleared edges, contradicting the minimality of S'. Thus, clearing such an edge cannot reduce the number of exposed vertices. But then, clearing xy next would produce a search strategy with fewer edges to be cleared after the number of exposed vertices becomes k for the last time, again contradicting the minimality of S'. Similarly, if there are any contaminated edges between v_i and v_j, γ_k may clear these edges first, and then xy, again contradicting the minimality of S'. So we may assume that all edges between the v_i have already been cleared, as have all pendant edges and loops incident with them.

If some vertex v_i is incident with only one contaminated edge, then γ_i may clear that edge first, and then γ_k may clear xy, again contradicting the minimality of S'. Thus, each v_i must have at least two contaminated edges incident with it, and the γ_i, $1 \leqq i \leqq k-2$, must remain where they are as blockers.

Neither x nor y are exposed before xy is cleared so that all edges incident with x and y are contaminated. After xy is cleared, both x and y are exposed. Thus, each of them is incident with a contaminated edge. Since G has no vertices of degree 2, both x and y must have at least two contaminated edges incident with them, and thus neither γ_{k-1} nor γ_k may move, contradicting that S' is a search strategy.

Case 3a: The number of exposed vertices just before xy is cleared is $k-1$ and one of the vertices of xy already is an exposed vertex. Label the exposed vertices v_1, v_2, …, v_{k-1}, and assume that they have searchers on them, with searcher γ_i on vertex v_i, $1 \leqq i \leqq k-1$. Without loss of generality, assume that $x = v_{k-1}$. Since the vertex v_{k-1} is still exposed, we may assume that γ_{k-1} stays on v_{k-1}, that the remaining searcher γ_k clears $v_{k-1}y$ by traversing the edge from v_{k-1} to y, and that there is another contaminated edge $v_{k-1}z$ incident with v_{k-1}.

If there are any pendant edges or loops attached to some v_i that are not cleared, we use the remaining searcher γ_k to clear these edges first, and then $v_{k-1}y$, contradicting the minimality of S'. In particular, $v_{k-1}z$ is not pendant so that z must have degree at least 3. Similarly, if there are any contaminated edges between v_i and v_j, γ_k may clear these edges first, then $v_{k-1}y$, again contradicting

the minimality of S'. So we may assume that all edges between the v_i already have been cleared, as have all pendant edges and loops incident with them.

If some vertex v_i is incident with only one contaminated edge, then γ_i may clear that edge first, then γ_k may clear $v_{k-1}y$, again contradicting the minimality of S'. Thus, each v_i must have at least two contaminated edges incident with it, and all the γ_i, $1 \leqq i \leqq k-2$, must remain where they are as blockers. Note that $\deg(y) > 1$, as otherwise searching $v_{k-1}y$ does not expose a new vertex. Since $\deg(y) \geqq 3$, we know that once γ_k clears $v_{k-1}y$, γ_k must remain on y. After $v_{k-1}y$ is cleared, if v_{k-1} has two or more contaminated edges incident with it, then γ_{k-1} must remain at v_{k-1}. Then no searchers may move, contradicting that S' is a search strategy. Thus, the only contaminated edge remaining incident with v_{k-1} must be $v_{k-1}z$. Thus, the next action in S' must be that γ_{k-1} clears $v_{k-1}z$. Since $\deg(z) \geqq 3$, z must have at least two contaminated edges incident with it, and thus γ_{k-1} also cannot move, contradicting that S' is a search strategy.

Case 3b: The number of exposed vertices is $k-1$, and neither of the vertices of xy is already exposed. Since the number of exposed vertices increases by 1 after xy is cleared, exactly one of x and y must have degree 1. (If both were degree 1, the graph would be disconnected.) Without loss of generality, assume that $\deg(x) = 1$. Then $\deg(y) \geqq 3$. Assume that the $k-1$ exposed vertices are labelled v_i and that the searcher γ_i is on v_i, $1 \leqq i \leqq k-1$. Then the searcher γ_k must clear xy.

As in the previous case, all edges between v_i must be cleared, as must all pendant edges and loops incident with them. Also, each v_i must have at least two contaminated edges incident with it. Thus, none of the γ_i, $1 \leqq i \leqq k-1$, may move. Similarly, since $\deg(y) \geqq 3$, y must have at least two contaminated edges incident with it, meaning that γ_k cannot move. This contradicts that S' is a search strategy.

This theorem tells us that there exist search strategies for some graphs that "save" searchers, in the sense we may keep a searcher in reserve, to never be stationed at an exposed vertex, but instead to clear edges between stationed searchers. If we consider the analogy of a graph filled with gas, we may always keep a searcher from being exposed, or by rotating searchers reduce the amount of "exposure" to a toxic substance.

The use of graph instead of multigraph in Theorem 4 is intentional. While it is possible that the result may be extended to some multigraphs, this proof does not suffice.

We first introduce a lemma from [10] to be used in the proof of Theorem 5.

Lemma 2. *If G is a graph and $\delta(G) \geqq 3$, then the number of cycles with pairwise distinct vertex sets is greater than $2^{\frac{\delta}{2}}$.*

Theorem 5. *If G is a connected graph with $\delta(G) \geqq 3$, then $\mathrm{s}(G) \geqq \delta(G) + g(G) - 2$.*

Proof. From Lemma 2, we know that the girth of G is finite, that $g = g(G) \geqq 3$, and that G has at least 3 cycles. Since $\delta = \delta(G) \geqq 3$, it follows from Theorem 4

that there exists a monotonic search S with $\text{ms}(G) = \text{s}(G)$ searchers such that $\text{mex}_S(G) = \text{ms}(G) - 1$. Let $E_0, E_1, ..., E_m$ be the sequence of cleared edge sets corresponding to S. Let G_i be the graph induced by the cleared edges in E_i.

Case 1. $\delta \geqq g = 3$. Consider the smallest i such that G has one cleared vertex u at step i. Since $\deg(u) \geqq \delta$, G must have at least δ exposed vertices adjacent to u. Since S exposes at most $\text{ms}(G) - 1$ vertices, $\delta \leqq \text{s}(G) - 1$, and thus $\text{s}(G) \geqq \delta + 1 = \delta + g - 2$.

Case 2. $\delta \geqq g = 4$. Let i be the least number such that G has at least two cleared vertices u and v at step i. If u and v are adjacent, they can have no common neighbours, and since $\deg(u) \geqq \delta$ and $\deg(v) \geqq \delta$, they must both be adjacent to at least $\delta - 1$ exposed vertices each. This accounts for $2\delta - 2$ searchers, and $2\delta - 2 \geqq \delta + g - 2$, as required. If u and v are not adjacent, then they may share common neighbours. At worst, all their neighbours are common. Consider the graph G_{i-1}. Since u and v are not adjacent, only one of them can become cleared by the next move. Assume that v is already cleared at step $i - 1$, and u becomes clear at step i. Then v has at least δ exposed vertices adjacent to it, and certainly u itself is exposed at this point. Thus G must have at least $\delta + 1$ different exposed vertices at step $i - 1$. Since S exposes at most $\text{ms}(G) - 1$ vertices, $\delta + 1 \leqq \text{ms}(G) - 1$, and thus $\text{ms}(G) \geqq \delta + 2 = \delta + g - 2$.

Case 3. $\delta \geqq g \geqq 5$. Let i be the least number such that G has at least two cleared vertices u and v at step i. If these two vertices are adjacent, then one must have $\delta - 1$ exposed vertices adjacent to it, and the other must have at least $\delta - 2$ exposed vertices adjacent to it (it may be adjacent to a third cleared vertex). Thus $2\delta - 3 \leqq \text{ms}(G) - 1$, and $\text{ms}(G) \geqq 2\delta - 2 \geqq \delta + g - 2$. If u and v are not adjacent, they have at most one neighbour in common, and hence again must have at least $2\delta - 3$ exposed vertices between them. Thus, as above, $\text{ms}(G) \geqq \delta + g - 2$.

Case 4. $g > \delta = 3$. Consider the smallest i such that G_i contains exactly one cycle C. Then each vertex of this cycle is either exposed or cleared. (Since only one edge was cleared, if G_i contained more than one cycle, then G_{i-1} must have contained a cycle.) Let u be a cleared vertex in C. Consider the graph H obtained when the edges of C are removed from G_i. Certainly, H is a forest, as G_i contained exactly one cycle. Then u is certainly in one of the non-trivial component trees that make up H. Since there are no vertices of degree 1 in G, any vertices of degree 1 in H must be exposed. Thus, there is an exposed vertex in the tree containing u. Further, this exposed vertex cannot be an exposed vertex in C, as this would mean that G_i contains two cycles. Thus, for every cleared vertex in C, there is an exposed vertex in G. Certainly, for every exposed vertex in C there is a corresponding exposed vertex (itself), and the number of exposed vertices is at least g. Since the monotonic search strategy S exposes at most $\text{ms}(G) - 1$ vertices, $g \leqq \text{ms}(G) - 1$, and thus $\text{ms}(G) \geqq g + 1 \geqq \delta + g - 2$.

Case 5. $g > \delta \geqq 4$. Let i_1 be the smallest i such that G_{i_1} has two or more cycles. Accordingly, we know G_i has at most one cycle for any $i < i_1$. If C_1 and C_2 are two of the cycles formed and are vertex-disjoint, then as before, there is

an exposed vertex that corresponds to each vertex in each cycle. But at most one exposed vertex may correspond to a vertex in both cycles. Thus the number of exposed vertices is at least $2g-1$, and so $\text{ms}(G) \geqq 2g \geqq \delta + g - 2$. If C_1 and C_2 share exactly one common vertex, then there are at least $2g-2$ exposed vertices at step i_2. Again, $\text{ms}(G) \geqq 2g-1 \geqq \delta+g-2$. If C_1 and C_2 share more than one vertex, then G_{i_2} contains exactly three cycles. In this case, we consider step i_2, the first moment that the graph G_i contains four or more cycles.

Let C be the subgraph of G formed by $V(C) = V(C_1) \cup V(C_2)$ and $E(C) = E(C_1) \cup E(C_2)$, as shown in Figure 1(i). Let one of the new cycles formed be C_3. If C_3 is vertex-disjoint from C, then G_{i_2} contains two vertex-disjoint cycles, and as before, the number of exposed vertices is at least $2g-1$. Thus, $\text{ms}(G) \geqq 2g \geqq \delta + g - 2$. If C_3 and C share exactly one vertex, then there are at least $2g-2$ exposed vertices at step i_2. Again, $\text{ms}(G) \geqq 2g-1 \geqq \delta+g-2$. Otherwise, C and C_3 share two or more vertices. We consider some subcases (see Figure 1).

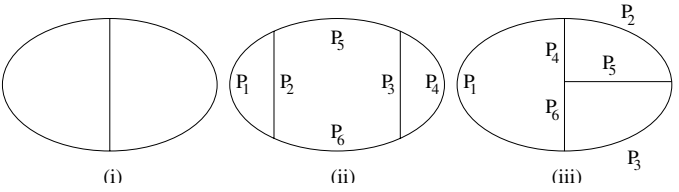


Fig. 1. (i) The graph C; (ii) Case 5(a); (iii) Case 5(b)

Case 5(a). In this case, we consider four cycles: the cycle induced by the paths P_1 and P_2; the cycle induced by P_2, P_3, P_5, and P_6; the cycle induced by P_3 and P_4; and finally the cycle induced by P_1, P_4, P_5, and P_6. These cycles all have length at least g. We note that either or both of P_5 and P_6 may be paths of length zero. Summing the lengths of the cycles, we see that we count each path, and hence each edge, exactly twice. Thus, in this subgraph G', $E' = E(G') \geqq 2g$. We next consider how many vertices are in $V' = V(G')$. If neither P_5 nor P_6 are paths of length zero, then summing vertex degrees over V' shows that $2(|V'|-4)+3\cdot 4 = 2|E|$, or that $|V'| = |E'| - 2 \geqq 2g-2$. In this case, every vertex corresponds to an exposed vertex, and so $\text{ms}(G) \geqq 2g-1 \geqq \delta+g-2$. If exactly one of P_5 or P_6 is a path of length zero, then summing vertex degrees over V' shows that $2(|V'|-3)+2\cdot 3+4 = 2|E'|$, or that $|V'| = |E'|-2 \geqq 2g-2$. All but one of these vertices must correspond to an exposed vertex, so $\text{ms}(G) \geqq 2g-2 \geqq \delta+g-2$. Finally, if both P_5 and P_6 are paths of length zero, then summing vertex degrees over V' shows that $2(|V'|-2)+2\cdot 4 = 2|E'|$, or that $|V'| = |E'|-2$. In this case, however, all but two vertices must correspond to an exposed vertex, so the number of exposed vertices is at least $|E'|-4 \geqq 2g-4 \geqq \delta+g-3$, since $g \geqq \delta+1$. Thus, $\text{ms}(G) \geqq \delta+g-2$.

Case 5(b). In this case, we again consider four cycles: the cycle induced by the paths P_1, P_4, and P_6; the cycle induced by the paths P_2, P_4, and P_5; the cycle induced by the paths P_3, P_5, and P_6; and the cycle induced by the paths P_1, P_2,

and P_3. Each cycle has length at least g. Consider the sum of the lengths of the cycles. Each path is counted twice, as is each edge. Thus, in this subgraph G', the total number of edges $|E'| \geqq 2g$. We sum the degrees of the vertices, and find that $2(|V'|-4)+4\cdot 3 = 2|E'|$, or that $|V'| = |E'|-2 \geqq 2g-2$. Since each vertex in G' corresponds to an exposed vertex, we see that $\mathrm{ms}(G) \geqq 2g-1 \geqq \delta+g-2$.

In fact, this result is best possible. Recall that the *complete bipartite graph* $K_{a,b}$ on $a+b$ distinct vertices, where $1 \leqq a \leqq b$, is the graph with vertex set $V(K_{a,b}) = \{v_1, v_2, \ldots, v_a\} \cup \{u_1, u_2, \ldots, u_b\}$ and edge set $E(K_{a,b}) = \{u_i v_j | 1 \leqq i \leqq b, 1 \leqq j \leqq a\}$. We now have sufficient tools to calculate the search number of the complete bipartite graph for all possible values of a and b.

Corollary 1. *Let* $1 \leqq a \leqq b$.

1. *If* $a = 1$ *and* $1 \leqq b \leqq 2$, *then* $\mathrm{s}(K_{a,b}) = 1$.
2. *If* $a = 1$ *and* $b \geqq 3$, *then* $\mathrm{s}(K_{a,b}) = 2$.
3. *If* $a = b = 2$, *then* $\mathrm{s}(K_{a,b}) = 2$.
4. *If* $a = 2$ *and* $b \geqq 3$, *then* $\mathrm{s}(K_{a,b}) = 3$.
5. *If* $3 \leqq a \leqq b$, *then* $\mathrm{s}(K_{a,b}) = a+2$.

A similar result can be shown for a complete multipartite graph.

Theorem 6. *For a complete multipartite graph* $K_{m_1,\ldots,m_k}$, *where* $m_1 \leqq \ldots \leqq m_k$, *if* $m_k \geqq 3$ *and* $k \geqq 3$, *then* $\mathrm{s}(G) = \sum_{i=1}^{k-1} m_i + 2$.

Proof. Let $\sum_{i=1}^{k-1} m_i = x$. It is easy to see that $\mathrm{s}(K_{m_1,\ldots,m_k}) \leqq x+2$. Suppose $K_{m_1,\ldots,m_k}$ can be cleared by $x+1$ searchers. By Theorem 4, there exists a monotonic search strategy S with $\mathrm{ms}(G)$ searchers such that $\mathrm{mex}_S(G) = \mathrm{ms}(G)-1 \leqq x$. Let $V_1, \ldots, V_k$ be the k parts of the vertex set with $|V_j| = m_j$, and $v \in V_i$ be the first cleared vertex using strategy S. Thus, all neighbours of v must be exposed vertices. If $m_i < m_k$, v has at least $x+1$ neighbours. This contradicts that $\mathrm{mex}_S(G) \leqq x$. If $m_i = m_k$, the x neighbours of v must be exposed vertices. Since $\mathrm{mex}_S(G) \leqq x$, each of other vertices in V_i must be contaminated. Since $m_i = m_k \geqq 3$, each of these exposed vertices has at least 2 contaminated edges incident on it. When we use the only free searcher to clear any edge incident on a vertex in $V_i - \{v\}$, we have $x+1$ exposed vertices, each of which is incident with at least two contaminated edges. Thus, no searcher can move, a contradiction.

The Petersen graph P is a cubic graph with girth 5. Thus, $\mathrm{s}(P) \geqq 6$. In fact, 6 searchers are sufficient. To see this, place a searcher on each of the vertices of a 5-cycle in P. Use a sixth searcher to clear the 5-cycle induced by these vertices. Move each searcher from the vertex it is on along the single remaining contaminated edge incident with it. This leaves searchers on every remaining uncleared vertex, and the sixth searcher can then clear the 5-cycle induced by these vertices, clearing the graph. In the same fashion, Theorem 5 implies that the Heawood graph and the McGee graph, which have girths 6 and 7, respectively, must have search numbers at least 7 and 8. In fact, it can be shown that these numbers are also sufficient to clear these graphs. The search strategies are similar to those for the Petersen graph.

3 Chromatic Number

If a graph G has a clique of order k, then at least k colours are required for a proper colouring. Thus, for any graph G, $\omega(G) \leqq \chi(G)$. Since we know that the clique number is a lower bound on the search number, it is reasonable to wonder whether Theorem 2 can be extended to the chromatic number.

Recall that the least number k such that the vertices of G can be ordered from v_1 to v_n in which each vertex is preceded by less than k of its neighbours is called the *colouring number* $\mathrm{col}(G)$ of G. Theorem 7 comes from [5]. Corollary 2 then follows directly from Theorems 7, 1, and 3.

Theorem 7. *For every connected graph G, $\chi(G) \leqq \mathrm{col}(G) \leqq \max\{\delta(H) | H \subseteqq G\} + 1$.*

Corollary 2. *For every connected graph G, $\chi(G) - 1 \leqq \mathrm{s}(G)$.*

We also offer a constructive proof for Corollary 2 which gives a proper colouring of G using at most $\mathrm{s}(G) + 1$ colours.

We begin by introducing the homeomorphic reduction of a reflexive multigraph X. Let $V' = \{u \in V(X) : \deg(u) \neq 2\}$. A *suspended path* in X is a path of length at least 2 joining two vertices of V' such that all internal vertices of the path have degree 2. A *suspended cycle* in X is a cycle of length at least 2 such that exactly one vertex of the cycle is in V' and all other vertices have degree 2. Let $V' = \{u \in V(X) : \deg(u) \neq 2\}$. The *homeomorphic reduction* of X is the reflexive multigraph X' obtained from X with vertex set V' and the following edges. Any loop of X incident with a vertex of V' is a loop of X' incident with the same vertex. Any edge of X joining two vertices of V' is an edge of X' joining the same two vertices. Any suspended path of X joining two vertices of V' is replaced by a single edge in X' joining the same two vertices. Any suspended cycle of X containing a vertex u of V' is replaced by a loop in X' incident with u. In the special case that X has connected components that are cycles, these cycles are replaced by loops on a single vertex.

Lemma 3. *If X is a connected reflexive multigraph and Y is its homeomorphic reduction, then $\mathrm{s}(X) = \mathrm{s}(Y)$.*

To obtain a bound on the search number involving chromatic number, we return to the idea of the maximum number of exposed vertices in a search.

Theorem 8. *If G is a connected reflexive multigraph with homeomorphic reduction G' and a monotonic search strategy S for G' such that $\mathrm{mex}_S(G') \geqq 3$, then $\chi(G) \leqq \mathrm{mex}_S(G') + 1$.*

Proof. Let $\mathrm{mex}_S(G') = k$. We will show that G is $(k+1)$-colourable. We first show that G' is $(k+1)$-colourable. Following the monotonic search strategy S that exposes at most k vertices in G', we can design a colouring such that it can colour G' using at most $k+1$ colours.

Initially, searchers are placed on G'. When a vertex first becomes exposed (or in the case of vertices of degree 1, becomes cleared), the vertex is coloured. This colour cannot be changed or erased in the following searching process. We now consider how to colour a vertex v in the moment it becomes exposed (or cleared, in the case of vertices of degree 1). Before this moment, v cannot be adjacent to any cleared vertex. Thus, each coloured vertex that is adjacent to v must be an exposed vertex. Since the number of exposed vertices is less then or equal to k, we can always assign v a colour that is different from the colours of the adjacent vertices of v. Thus, while S clears G', we can assign a colour to each vertex of G' such that any pair of adjacent vertices has different colours. Thus, G' is $(k+1)$-colourable.

We now show that G is $(k+1)$-colourable. For each vertex u in G', assign the colour of u in G' to the corresponding vertex u in G. Any uncoloured vertex in G must be on a suspended path or a suspended cycle. If it is on a suspended cycle, one vertex in this cycle has already been coloured. At most two more colours are needed to colour the remaining vertices of this cycle, but since $k \geqq 3$, we have a sufficient number of colours to do so. Similarly, if the vertex is in a suspended path, the ends of the suspended path have already been coloured. Now at most one more colour is needed to colour the remaining vertices of this path, but again, we have sufficient colours to do so. Hence, G is $(k+1)$-colourable. Therefore, $\chi(G) \leqq k+1$.

Combining Theorem 8 with Lemma 1, we obtain the following corollary, an improvement on Corollary 2.

Corollary 3. *If G is a connected reflexive multigraph and $\mathrm{s}(G) \geqq 3$, then $\chi(G) - 1 \leqq \mathrm{s}(G)$.*

Of course, we can do better. As demonstrated in Theorem 4, there are graphs where the maximum number of exposed vertices is one less than the search number.

Corollary 4. *If G is a connected reflexive graph with $\mathrm{s}(G) \geqq 3$ and the property that its homeomorphic reduction is not a multigraph, then $\chi(G) \leqq \mathrm{s}(G)$.*

Proof. Since G is not a multigraph, the homeomorphic reduction can only have multiple edges if two or more suspended paths have the same end points. Forbidding this, the homeomorphic reduction must be a graph with no vertices of degree 2, as required by Theorem 4. The result follows.

We now demonstrate an infinite family of graphs for which Corollary 2 provides a better bound than any of the others demonstrated here. Let P be the graph with vertex set $V(P) = \{v_i\}_{i=1}^{p+1}$, and edge set $E(P) = \{v_i v_j | 1 \leqq i < j \leqq p\} \cup \{v_1 v_{p+1}\}$. Thus, the graph P is a complete graph on p vertices with an extra edge incident with a vertex of degree 1.

We will employ the Mycielski construction [9]. Given a graph G, we form the graph $M(G)$, with vertex set $V(M(G)) = V(G) \cup V'(G) \cup \{u\}$, where $V'(G)$ contains the "twins" of $V(G)$. That is, $V'(G) = \{x' | x \in V(G)\}$. The edge set

$E(V(M)) = E(G) \cup \{x'y | xy \in E(G)\} \cup \{x'u | x' \in V'(G)\}$. That is, for each vertex $v \in V$, we introduce a new vertex v' adjacent to the neighbours of v. Finally, we add a new "super vertex" u which is adjacent to each new vertex v'. Similarly, we may define an infinite family of graphs by repeatedly applying a Mycielski construction. Define $M^0(G) = G$, and $M^t(G) = M(M^{t-1}(G))$ for $t \geqq 1$.

The Mycielski construction based on the 5-cycle C_5 was introduced in [9] to create an infinite family of triangle-free graphs with arbitrarily large chromatic number. In fact, $\chi(M^t(C_5)) = t + 3$ for $t \geqq 0$. More generally, for any graph G, it can be shown that $\omega(M^t(G)) = \omega(G)$, $\delta(M^t(G)) = \delta(G) + t$, and $\chi(M^t(G)) = \chi(G) + t$ for $t \geqq 0$.

Taking the graph P as defined above, it is clear that $\delta(P) = 1$, $\omega(P) = p$, and $\chi(P) = p$. Applying the Mycielski construction, we see that $\delta(M^t(P)) = 1 + t$, $\omega(M^t(P)) = p$, and $\chi(M^t(P)) = p + t$. As well, since P is a subgraph of $M^t(P)$, we know that $g(M^t(P)) = 3$ so long as $p \geqq 3$. So for large p and t, Theorem 5 tells us that $\delta(M^t(P)) + 1 \leqq t + 2 \leqq \mathrm{s}(M^t(P))$. Similarly, Theorem 2 tells us that $\omega(M^t(P)) = p \leqq \mathrm{s}(M^t(P))$. But Corollary 2 tells us that $\chi(M^t(P)) - 1 = p + t - 1 \leqq \mathrm{s}(M^t(P))$, a clear improvement.

References

1. Alspach, B.: Searching and sweeping graphs: A brief survey. Combinatorics 04 (Catania, 2004) Matematiche (Catania) 59 (2004), Fasc. I–II, pp. 5–37 (2004)
2. Bienstock, D., Seymour, P.: Monotonicity in graph searching. Journal of Algorithms 12, 239–245 (1991)
3. Fellows, M., Langston, M.: On search, decision and the efficiency of polynomial time algorithm. In: 21st ACM Symp. on Theory of Computing, pp. 501–512 (1989)
4. Frankling, M., Galil, Z., Yung, M.: Eavesdropping games: A graph-theoretic approach to privacy in distributed systems. Journal of ACM 47, 225–243 (2000)
5. Halin, R.: Unterteilungen vollständiger Graphen in Graphen mit unendlicher chromatischer Zahl. Abh. Math. Sem. Univ. Hamburg 31, 156–165 (1967)
6. Kirousis, L.M., Papadimitriou, C.H.: Searching and pebbling. Theoret. Comput. Sci. 47(2), 205–218 (1986)
7. LaPaugh, A.S.: Recontamination does not help to search a graph. Journal of ACM 40, 224–245 (1993)
8. Megiddo, N., Hakimi, S.L., Garey, M., Johnson, D., Papadimitriou, C.H.: The complexity of searching a graph. Journal of ACM 35, 18–44 (1988)
9. Mycielski, J.: Sur le coloriage des graphes. Coll. Math. 3, 161–162 (1955)
10. Tusa, Z.: Exponentially many distinguishable cycles in graphs. Graphs, designs and combinatorial geometries (Catania, 1989). J. Combin. Inform. System Sci. 15, 281–285 (1990)
11. Yang, B., Dyer, D., Alspach, B.: Sweeping graphs with large clique number (extended abstract). In: Fleischer, R., Trippen, G. (eds.) ISAAC 2004. LNCS, vol. 3341, pp. 880–892. Springer, Heidelberg (2004)

Author Index