10th Cologne-Twente Workshop on Graphs and Combinatorial Optimization

CTW 2011

Villa Mondragone, Frascati, June 14-16, 2011

Proceedings of the Conference

Ludovica Adacher, Marta Flamini, Gianmaria Leo,
Gaia Nicosia, Andrea Pacifici, Veronica Piccialli (Eds.)

10th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2011)

Villa Mondragone,

Frascati, June 14-16, 2011

The Cologne-Twente Workshop (CTW) on Graphs and Combinatorial Optimization started off as a series of workshops on theory and applications of discrete algorithms, graphs and combinatorial structures in the wide sense, organized by either Köln University or Twente University.

The first nine editions of CTW have been held in Cologne (2001, 2005, and 2010); Enschede (2003 and 2007); Menaggio (2004); Lambrecht (2006); Gargnano (2008) and Paris (2009). The 10th edition of the workshop will be hosted by Università di Roma "Tor Vergata", in Villa Mondragone, an historical villa located in Frascati (Rome), Italy, from June 14 to 16, 2011.

As in previous editions, a special issue of Discrete Applied Mathematics (DAM) will be devoted to CTW 2011, containing full-length versions of selected presentations given at the workshop and possibly other contributions related to the workshop topics.

CTW 2011 will honor the memory of Prof. Bruno Simeone (Università degli Studi di Roma "La Sapienza") who recently passed away. We are grateful to Bruno's friends and colleagues who are commemorating his fundamental contributions in the field of combinatorial optimization and graph theory of the last four decades.

We wish to thank the members of the Program Committee: U. Faigle (Universität zu Köln), J. Hurink (Universiteit Twente), L. Liberti (École Polytechnique), F. Maffioli (Politecnico di Milano), G. Righini (Università degli Studi di Milano), R. Schrader (Universität zu Köln), R. Schultz (Universität Duisburg-Essen), for setting up such an attractive program. Special thanks also go to some volunteering referees.

LUDOVICA ADACHER, MARTA FLAMINI, GIANMARIA LEO, GAIA NICOSIA, ANDREA PACIFICI, VERONICA PICCIALLI

Università di Roma "Tor Vergata" e "Roma Tre",
Roma, Italy, June 6, 2011

Organization

The CTW 2011 conference is co-organized by the Università di Roma "Tor Vergata" and Università "Roma Tre". The conference is hosted by Villa Mondragone in Frascati, thanks to a sponsorship of Università di Roma "Tor Vergata". We are also grateful to Microsoft for their partial support the workshop.

Scientific Committee

- Ulrich Faigle (U. Cologne),
- Johann L. Hurink (U. Twente),
- Leo Liberti (École Polytechnique, Paris)
- Francesco Maffioli (Politecnico, Milano),
- Gaia Nicosia (U. Roma Tre),
- Andrea Pacifici (U. Roma Tor Vergata),
- Giovanni Righini (U. Milano),
- Rainer Schrader (U. Cologne),
- Rudiger Schultz (U. Duisburg-Essen)

Organizing Committee

- Ludovica Adacher (U. Roma Tre),
- Marta Flamini (U. Uninettuno),
- Gianmaria Leo (U. Roma Tor Vergata),
- Gaia Nicosia (U. Roma Tre),
- Andrea Pacifici (U. Roma Tor Vergata),
- Veronica Piccialli (U. Roma Tor Vergata)

Table of contents

Memorial Session Talks

Contributed Abstracts

Memorial Session Talks

Research Work with Bruno Simeone

Ronald I. Becker

AIMS (African Institute for Mathematical Sciences),6-8 Melrose Road, Muizenberg 7945, South Africa
Department of Mathematics and Applied Mathematics
University of Cape Town, Rondebosch 7701, South Africa
Ronald.Becker@uct.ac.za

Key words: Graph Partitioning, Continuous Partitioning, Tree Partitioning, Grid Graph partitioning, Optimization, Graph Colouring, Gerrymandering

---

1 Introductory Remarks

I collaborated with Bruno Simeone and his research group for a period of more than 20 years. It was one of the best and most fruitful enterprises of my professional life. The continued striving for scientific excellence, the wonderfully pleasant atmosphere that Bruno generated and the continued friendship once the day's work was over are things that I will always hold dear.

I was a visitor to "La Sapienza" in most years during the period, for varying lengths of time. Bruno and his family also visited me in Cape Town on one occasion for an extended period, and I would have liked them to come again but this never happened.

I will reserve further comments for the talk.

The lecture will describe briefly three pieces of joint work with Bruno and other colleagues, the first near the beginning of our collaboration, the second around the middle and the third at the end. The problems are stated below and graphical illustrations will be given in the talk. The problems all involve graph partitioning, which was a topic very close to Bruno's heart.

2 Max-min Partitioning of Grid Graphs and Ladders

We consider an undirected $M \times N$ grid graph G which we think of as embedded in $\mathbb{R}^2$ with vertices a set of integer pairs with each vertex joined by an edge to all its nearest neighbours. Each vertex v has a weight $w(v) \in \mathbb{R}$. A p-partition is a decomposition of G into p (connected) non-empty components.

Max-min p-Partition of G
To find a p-partition π^* which maximizes

$$\min_{1 \le i \le p} w(C_i \in \pi)$$

over all p-partitions π.

This is a natural generalisation to a fairly minimal set of graphs with cycles of a much-studied problem for trees to which members of the group have made a number of contributions. In particular, shifting algorithms have been investigated. These start with one component (the whole graph) and progressively add boundaries separating the graph into additional components, one at a time, until there are p components, and then the partitions are improved until an optimum is reached. Generally changes are made by a form of steepest descent principle. The two papers [2],[3] give additional references.

These papers show that for grid graphs with 3 or more rows the problem is NP-complete. For graphs with 2 rows (ladders) there is a dynamic programming algorithm which solves the problem in polynomial time. Further, there are shifting algorithms which give a good approximation to the optimum.

3 A Shifting Algorithm for Continuous Tree Partitioning

Shifting Algorithms for Tree Partition have been extensively studied (see [1] for references). These give elegant algorithms for max-min p-partitioning a tree with weighted vertices, or weighted edges or both. Consider the case of edge-weighted trees. The algorithms only allow components which contain complete edges. One might want to consider partitions which contain only connected portions of edges, thus cutting off edges at some point in the middle. This problem is referred to as Continuous Partitioning. For practical problems we may consider only edges with rational lengths and this leads equivalently to considering only problems with integer lengths. We then place additional vertices along the edges so that all new edges have length 1. We can then use a shifting algorithm to find an optimal partition. However, the complexity would depend on the sizes of the edges. This is clearly undesirable.

The paper [1] uses the shifting algorithm framework but modifies it to allow several moves along an edge at a time in such a way that the optimum is reached in polynomial time.

4 Bicoloured Graph Partitioning

Consider once more a rectangular grid graph. Minimal sub-squares are considered as the units of our problem. A (connected) component is a collection of units each of which has an edge in common with another unit in the component. A unit is thought of as a district all of whose voters vote for a single one of two possible parties (blue and red). We suppose that it is possible to find a partition all of whose components have the same (odd) number $s > 0$ of units. We think of each component as a seat and the party that wins the most units in the component takes the seat.

The question arises in the context of gerrymandering of constituencies of how badly can one gerrymander. In other words, can we find a colouring of the units so that one partition has blue winning as many seats as is possible with its total number of units, while another partition has the reverse property that blue wins as many seats as is possible with its total number of units - further, blue takes the election in the first case and red takes the election in the second case. We will suppose further that the number of blue units is equal to the number of red units + 1 since this will make gerrymandering a more grievous problem.

For graphs with an even number of units, the grid graph is Hamiltonian. This allows us to reduce the problem of finding two partitions with diametrically opposite properties to colouring the vertices of a circle graph in two extreme ways. It can be shown that there are colourings which transform from one extreme to the other by a rotation of all the partition boundaries, so this is always possible. The case for graphs with an odd number of units can then be reduced to graphs with an even number and an additional single boundary shift. See reference [4] for details.

References

[1] R. Becker, B. Simeone, Yen-I Chang, A shifting algorithm for continuous tree partitioning,Theoretical Computer Science 282 (2002) 353-380.

[2] Ronald Becker, Isabella Lari, Mario Lucertini, Bruno Simeone. Max-min Partitioning of Grid Graphs into Connected Components. Networks 32 (1998) 115-125.

[3] R. Becker, I. Lari, M. Lucertini, B. Simeone, A Polynomial-Time Algorithm for Max-min Partition of Ladders. Theory Comput. Systems 34 (2001) 353-374.

[4] N. Apollonio, R.I. Becker, I. Lari, F. Ricca, B. Simeone, Bicoloured graph partitioning, or: gerrymandering at its worst. Discrete Applied Mathematics 157 (2009) 3601-3614.

Incompatibility Graphs and Data Mining
In Memory of Bruno Simeone (1945-2010)

E. Boros, [a] V. Spinelli, [b] F. Ricca [b]

[a] *Rutgers Center for Operations Research, RUTCOR, 640 Bartholomew Road, Piscataway, NJ 08854-8003.*
boros@rutcor.rutgers.edu

[b] *ISTAT - Istituto Nazionale di Statistica, Via Tuscolana, 1788, 00173, Rome, Italy.*
vispinel@istat.it

[c] *Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università 'La Sapienza', Piazzale Aldo Moro 5, 00185, Rome, Italy.*
federica.ricca@uniroma1.it

Key words: Box clustering, incompatibility graphs, embeddings.

1 Introduction

In this paper [1], we introduce Incompatibility Graphs (IGs), a class of graphs that arises in the *Box Clustering* (*BC*) approach to the supervised classification of data. These graphs turn out to have an autonomous interest from a theoretical viewpoint. Box Clustering was introduced in [9] and it can be viewed as an offspring of a more general methodology, called *Logical Analysis of Data* (*LAD*) (see, for example, [8,5,3]). Unlike *LAD*, *BC* has the ability to deal directly, not only with binary data, but also with numerical and ordinal ones. The input of a *BC* problem is a *training* data set, consisting of a finite set of points in a d-dimensional space, which are classified either as *positive* or *negative*. A box (i.e., a d-dimensional closed interval) is called *positive* (or *negative*) if it includes some positive (resp. negative) observations, but does not include any negative (resp. positive) one. Positive and negative boxes will also be called *homogeneous*. The output of a *BC* model is a set of homogeneous boxes, which are used to predict the class of an unclassified observation belonging to a "*testing*" data set. In this paper we address two key *BC* problems, namely, the Maximum Box (*MB*) and the Minimum Covering by Boxes (*MCB*), see [13,14].

In order to solve these problems, we introduce Incompatibility Graphs that, for a given set of observations in the d-dimensional space, represent the structural rela-

[1] This is a joint work we started with Bruno and we intend to publish the full version with him as a co-author.

tions of homogeneity between pairs of points. These graphs have shown to be of help in the solution of both the two above mentioned *BC* problems.

We shall regard an observation, w.l.o.g., as a point in the real d-dimensional space $\mathbb{R}^d$. For two vectors $l, u \in \mathbb{R}^d$, such that $l_i \leq u_i, i = 1, \ldots, d$, we denote by $I(l, u) = \{x \in \mathbb{R}^d : l_i \leq x_i \leq u_i, i = 1, \ldots, d\}$ the spanned *box* (or a *hyper-rectangle*). For a finite set $S \subset \mathbb{R}^d$ the *box-closure* of S is the smallest box containing all points in S. Let $L_i = \min_{x \in S}\{x_i\}$ and $U_i = \max_{x \in S}\{x_i\}$. The box-closure of S is given by $[S] = I(L, U)$. Notice that L and U are two *bounding points* in $\mathbb{R}^d$ that determine the box-closure $[S]$, though themselves may not belong to S. Note that for any finite set S of points, the box-closure of S can be also seen as the intersection of all boxes containing S, see e.g., [9].

Suppose that $S \subset \mathbb{R}^d$ is the set of the n points in $\mathbb{R}^d$ representing a *BC* data set. Let P and N denote the two finite non empty subsets of S corresponding to the *positive* and *negative* examples, respectively, so that one has $P \cap N = \emptyset$ and $S = P \cup N$. We say that a box $B \subseteq \mathbb{R}^d$ is *homogenous* for P if $N \cap B = \emptyset$ and $P \cap B \neq \emptyset$. Finally a set of boxes $\mathcal{B}_m = \{B_1, \ldots, B_m\}$ is a homogenous cover of P if each box in $\mathcal{B}_m$ is a homogenous box for P, and they cover all points in P. Similar definitions hold for the set of negative points N.

On the basis of the above definitions, different *BC* models were provided in the literature in order to study different data analysis problems [11,12]. One of the main *BC* problems is *Minimum Covering by Boxes*, that is finding a homogenous box cover $\mathcal{B}_m$ for a given data set (P, N). with the minimum number of boxes. Another basic *BC* problem is the *Maximum Box* problem, that is finding a box B, which is homogenous for P, and $|B \cap P|$ is as large as possible.

2 Incompatibility Graphs

Given two finite and disjoint sets of points $P, N \subseteq \mathbb{R}^d$, we associate to them a graph $G = G_{P,N}$ on vertex set $V(G) = P$ such that two vertices $u, v \in P$ are connected in G by an edge if $[u, v] \cap N \neq \emptyset$. We call $G_{P,N}$ the d-incompatibility graph (d-IG) of P and N. We say that a graph is a d-IG, if there are subsets $P, N \subseteq \mathbb{R}^d$ such that $G = G_{P,N}$. The pair (P, N) will be called a d-embedding of G.

In this paper we show some few interesting properties of d-IG-s, provide a partial characterization of 2-IG-s, and demonstrate how to use these properties for solving the above problems arising in BC. We also pose some open conjectures.

The following properties follow by the definitions.

Property 1 (IG hereditary property) *G is a d-IG, then an induced subgraph of G is also a d-IG.*

Property 2 (IG monotonic dimension property) *If G is d-IG, then G is h-IG for every $h \geq d$.*

Theorem 2.1 (Universality of IGs) *If $G = (V, E)$ is an arbitrary graph then there always exist $d \geq 1$ and $P, N \subseteq \mathbb{R}^d$ such that $G_{P,N}$ is isomorphic to G.*

Theorem 2.2 (Embeddings and general position) *If G is a d-IG, then it is isomorphic to some $G_{P,N}$, such that $P \cap N = \emptyset$ and the points of $P \cup N$ are in general position.*

Theorem 2.3 *A 2-IG cannot have $3K_2$ as an induced subgraph.*

Conjecture 1 *A d-IG cannot have an mK_2 as an induced subgraph for $m > 2^{d-1}$.*

Theorem 2.4 *A 2-IGcannot have C_m, $m \geq 7$ and P_n, $n \geq 8$ as induced subgraphs.*

Let us note that in a planar embedding of a 2-IGG all edges are tilting either (north-west, south-east), or (north-east, south-west) directions. We call G polarized, if it has an embedding with only one type of edge tilting.

Theorem 2.5 *Any polarized 2-IG is a comparability graph.*

Theorem 2.6 *Let G be a connected 2-IG that is not polarized. Then G has radius at most 3.*

Theorem 2.7 *If a 2-IG is nontrivially disconnected, then it has exactly two non trivial connected components and they are both weakly chordal.*

Due to space limitations we cannot list here more results. Let us however add that the problem of realizing a given graph as a d-IG, or in particular as a 2-IG is far from trivial. For instance, 2-IG-s seem to have an infinite set of minimal forbidden subgraphs.

3 IG-s and Box Clustering

To conclude this short abstract, let us point out that a maximum-box for a data set (P, N) corresponds to a maximum stable set in $G = G_{P,N}$, and a minimum box-covering corresponds to a coloring of G with the minimum number of colors.

Due to a result of [1], we know that graphs with no mK_2 as an induced subgraph have at most $O(2^{poly(m)})$ maximal stable sets, all of which can be generated by a results of [7] in $O(2^{poly(m)})$ time. Thus, by Theorem 2.3 the maximum stable set problem can be solved in a 2-IG in linear time, and hence the maximum box problem can also be solved in linear time. We hope to be able to extend this result to any constant dimension d. In general these problems are well-known to be NP-hard. Let us finally note that due to Theorem 2.7, the optimal coloring problem, and hence the minimum box-cover problem can also be solved in polynomial time for some special 2-IG-s.

References

[1] E. Balas and C. S. Yu, On graphs with polynomially solvable maximal-weight clique problem, *Networks* **19** (1989) 247–253.

[2] T.BONATES, P.L. HAMMER. *Logical Analysis of Data: from combinatorial optimization to medical applications*. Annals of Operations Research, 148: 203-225, 2006.

[3] E. BOROS, T. IBARAKI, K. MAKINO. *Boolean Analysis of Incomplete Examples*. RRR 7-1996.

[4] A. BRANDSTÄDT, V.B. LE, J.P. SPINRAD. *Graph Classes: A Survey*. SIAM Monographs on Discrete Mathematics and Applications, ISBN 978-0-89871-432-6, 1999.

[5] Y. CRAMA, P.L. HAMMER, T. IBARAKI. *Cause-effect relationship and partially defined Boolean functions*. Annals of Operational Research, 16: 299-325, 1988.

[6] G. FELICI, K. TRUEMPER. *A Minsat Approach for Learning in Logic Domains*. INFORMS Journal on Computing, 13 (3), 2001, 1-17.

[7] M.C. GOLUMBIC. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[8] P.L. HAMMER. *Partially defined Boolean functions and cause-effect relationship*. Lecture at the International Conference on Multi-Attribute Decision Making Via Or-Based Expert Systems, University of Passau Germany, April 1986.

[9] P.L. HAMMER, Y. LIU, S. SZEDMÁK, B. SIMEONE. *Saturated systems of homogeneous boxes and the logical analysis of numerical data*. Discrete Applied Mathematics, Volume 144, 1-2: 103-109, 2004.

[10] D. S. Johnson, M. Yannakakis and C. H. Papadimitriou (1988). On generating all maximal independent sets. *Information Processing Letters*, **27**, 119–123.

[11] B. SIMEONE, M. MARAVALLE, F. RICCA, V. SPINELLI. *Logic Mining of non-logic data: some extensions of box clustering*. Euro XXI, 21st European Conference on Operational Research. Reykjavik, Iceland, July 2-5, 2006.

[12] B. SIMEONE, V. SPINELLI. *The optimization problem framework for box clustering approach in logic mining*. Euro XXII - 22nd European Conference on Operational Research, Prague, page 193. The Association of European Operational Research Societies, July 2007.

[13] *Maximum Box Problem*. http://www.xxxx.it/ig.

[14] *Minimum Covering by Boxes*. http://www.xxxx.it/ig.

Control and voting power in complex shareholding networks

Y. Crama

HEC – Management School of the University of Liège
Boulevard du Rectorat 7 (B31), 4000 Liège, Belgium
Yves.Crama@ulg.ac.be

In this talk, we discuss how game-theoretic power indices can be used to model the amount of control held by individual shareholders in complex financial networks.

In its simplest form, when the shares of a firm are held by a number of independent players, the mechanism by which shareholders form a decision can be modeled as a majority voting game. Hence, game-theoretical measures of power, like the Banzhaf index [1] or the Shapley-Shubik index [4], can be used to define the "control" of shareholders. The resulting distribution of control is usually different from the nominal distribution of voting weights, mainly because the voting function is, in essence, nonlinear. A trivial example arises when there is a dominant voter whose voting weight exceeds the half of the total weight: This voter has absolute control, although he might only possess 51% of the shares.

Earlier researchers have proposed many methods for the computation of power indices for the weighted majority game played by the shareholders of a single firm [2]. The model can be rather easily generalized in the presence of "pyramidal", or multilayered levels of ownership: Here, the value of the power indices is determined by a compound voting game viewed as the composition of several weighted majority games.

However, little has been done regarding more complex situations, where the graph defined by shareholding relations may be *incomplete*, in the sense that some firms have many small, unidentified shareholders (so-called *float*), as well as *cyclic*, reflecting the intricacies of real-world financial networks.

In this talk, we describe an integrated algorithmic approach that allows us to deal efficiently with the complexity of computing power indices in large intricate networks. This approach has been successfully applied to the analysis of real–world financial networks [3].

We also discuss more recent attempts to provide a rigorous framework for the above model in the presence of float and of cyclic shareholding relationships. In this framework, we identify conditions under which Banzhaf power indices can be naturally generalized.

The presentation is based on joint work with Luc Leruth and Su Wang.

CTW2011, Villa Mondragone, Frascati, Italy. June 14–16, 2011

References

[1] J.F. Banzhaf III, Weighted voting doesn't work: A mathematical analysis, *Rutgers Law Review* 19 (1965) 317-343.

[2] Y. Crama and P.L. Hammer, *Boolean Functions: Theory, Algorithms, and Applications*, Cambridge University Press, New York, N.Y., 2011.

[3] Y. Crama and L. Leruth, Control and voting power in corporate networks: Concepts and computational aspects, *European Journal of Operational Research* 178 (2007) 879-893.

[4] L.S. Shapley and M. Shubik, A method for evaluating the distribution of power in a committee system, *American Political Science Review* 48 (1954) 787-792.

Graph Sandwich Problems

M. C. Golumbic

Caesarea Rothschild Institute and Department of Computer Science,
University of Haifa, Israel.
golumbic@cs.haifa.ac.il

Key words: graph sandwich problems, chain graphs, probe graph problems

---

1 Abstract

A sandwich problem for a graph with respect to a graph property Π is a partially specified graph, i.e., only some of the edges and non-edges are given, and the question to be answered is, can this graph be completed to a graph which has the property Π? The graph sandwich problem was investigated for a large number of families of graphs in a 1995 paper by Golumbic, Kaplan and Shamir [1], and much subsequent work has taken place since. In some cases, the problem is NP-complete such as for interval graphs, comparability graphs, chordal graphs and others. In other cases, the sandwich problem can be solved in polynomial time such as for threshold graphs, cographs, and split graphs, an important graph class studied by Stephan Foldes, Peter Hammer and Bruno Simeone.

There are also interesting special cases of the sandwich problem, most notably the *probe graph* problem where the unspecified edges are confined to be within a subset of the vertices. Similar sandwich problems can also be defined for hypergraphs, matrices and Boolean functions, namely, completing partially specified structures such that the result satisfies a desirable property.

In this talk, we will present a survey of results that we and others have obtained in this area during the past several years.

References

[1] M.C. Golumbic, H. Kaplan, R. Shamir. Graph sandwich problems. *J. Algorithms* 19 (1995) 449–473.

Bruno Simeone's Work in Clustering

P. Hansen

GERAD, HEC Montreal and LIX, Ecole Polytechnique, Palaiseau

1 Introduction

Given a set of entities, Cluster analysis, or Clustering, aims at finding subsets, called clusters, which are homogeneous and/ or well separated. Homogeneity means that entities in the same cluster shoud be similar and separation that entities in different clusters should be dissimilar.Often the clusters are requested to form a partition; sometimes they must satisfy additional constraints on connectivity, minimal separation or maximum weight.

Clustering interested Bruno Simeone all along his career, and he made a large number of diverse and innovative contributions: new concepts such as the espalier, many complexity results and polynomial algorihms for connectivity-constrained clustering, development of heuristics and exact algorithms, applications to image processing and political districting. We will review this work, stressing its influence and open problems.

CTW2011, Villa Mondragone, Frascati, Italy. June 14–16, 2011

Separating negative and positive points with the minimum number of boxes

Paolo Serafini

Dipartimento di Matematica e Informatica, Università di Udine, Italy
paolo.serafini@uniud.it
http://users.dimi.uniud.it/~paolo.serafini/

Key words: Data analysis, boxes, patterns

Identifying new data as either positive or negative on the basis of quantitative feature information is fundamental in data analysis. If a linear separation between positive and negative points is not possible, then a simple way to carry out this task is via boxes. Boxes can be easily described in mathematical terms and their properties allow to design good algorithms for the separation task.

In this paper we build upon the ideas developed in [1], where the problem of finding a single homogeneous box of maximum weight is fully investigated. It turns out that this problem can be used in pricing the column generation model we propose in this paper for computing the minimum number of boxes covering the positive points.

There are given p *positive* points $X^i \in R^n$ $(i = 1 \dots, p)$ and q *negative* points $Y^i \in R^n$, $(i = 1 \dots, q)$. A *box* $B(\ell, u)$ is the set

$$\{X \in R^n : \ell_i \leq X_i \leq u_i,\ i = 1 \dots, n\}\,.$$

If $X^i \in B(\ell, u)$ we say that the box $B(\ell, u)$ *covers* X^i, and similarly for Y^i. A box is said *positive* (*negative*) if it covers only positive (negative) points. A box which is either positive or negative is also called *homogeneous*. A family of boxes $B(\ell^j, u^j)$, $j = 1 \dots, s$, is *positive* if

$$X^i \in \bigcup_{j=1}^{s} B(\ell^j, u^j), \quad i = 1 \dots, p, \qquad Y^i \notin \bigcup_{j=1}^{s} B(\ell^j, u^j), \quad i = 1 \dots, q$$

In other words all boxes of a positive family are positive and in addition they jointly cover all positive points. Similarly a family of boxes $B(\ell^j, u^j)$, $j = 1 \dots, t$, is *negative* if

$$X^i \notin \bigcup_{j=1}^{t} B(\ell^j, u^j), \quad i = 1 \dots, p, \qquad Y^i \in \bigcup_{j=1}^{t} B(\ell^j, u^j), \quad i = 1 \dots, q$$

Positive and negative families are called homogeneous.

The problem we want to solve consists in finding a positive family of minimum cardinality. We model the problem as the following set covering problem. Let J be the set of all positive boxes. Then let

$$x_j = \begin{cases} 1 & \text{if thej-th box is in the family} \\ 0 & otherwise \end{cases}$$

and

$$a_j^i = \begin{cases} 1 & \text{if the } j\text{-th box covers } X^i \\ 0 & \text{otherwise} \end{cases}$$

Then the minimum cardinality problem can be formulated as

$$\begin{aligned} \min \ & \sum_{j \in J} x_j \\ & \sum_{j \in J} a_j^i \, x_j \geq 1, \qquad i = 1, \ldots, p \\ & x_j \in \{0, 1\} \end{aligned} \tag{1}$$

The relaxation of (1) can be solved via column generation. The dual pricing constraints are

$$\sum_{i=1}^{p} a_j^i \, y_i \leq 1 \qquad j \in J$$

So columns can be generated by solving a max weighted box problem. Though the pricing problem is NP-hard, it can be solved quickly via a specialized combinatorial algorithm described in [1]. However, we rely in this paper to the the ILP model also presented in [1]. We briefly recall this ILP model for the max weighted problem: let

$$V^k := \bigcup_{i=1}^{p} \{X_k\}, \qquad k = 1, \ldots, n$$

The k-th coordinate of a positive box can be assumed to take values in V^k. So in order to define a box we need assigning ℓ_k to exactly one value in V^k and similarly for u_k. Therefore let

$$\zeta_{kv} = \begin{cases} 1 & \text{if } \ell_k = v, \ v \in V^k \\ 0 & \text{otherwise,} \end{cases} \qquad \xi_{kv} = \begin{cases} 1 & \text{if } u_k = v, \ v \in V^k \\ 0 & \text{otherwise} \end{cases}$$

and the assignment is realized through the constraints

$$\sum_{v \in V^k} \zeta_{kv} = 1, \qquad \sum_{v \in V^k} \xi_{kv} = 1, \qquad k = 1, \ldots, n \tag{2}$$

We need counting the points covered by the box. To this aim let

$$\eta_i = \begin{cases} 1 & \text{if } X^i \text{ is covered} \\ 0 & \text{otherwise} \end{cases}$$

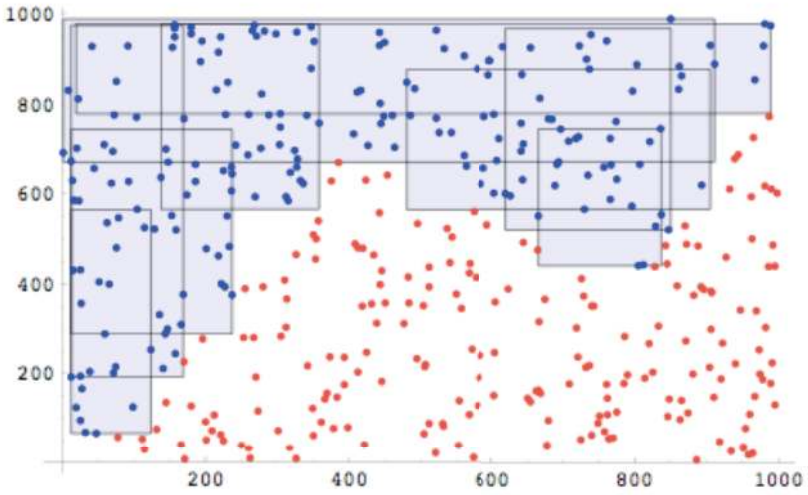


Fig. 1. An optimal family of boxes

and define the constraints

$$\eta_i \le \sum_{v \le X_k^i} \zeta_{kv}, \qquad \eta_i \le \sum_{v \ge X_k^i} \xi_{kv}, \qquad k = 1 \ldots, n, \quad i = 1 \ldots, p \tag{3}$$

For a box not to cover negative points the following constraints must be introduced

$$\sum_{k=1}^{n} \Big(\sum_{v > Y_k^i} \zeta_{kv} + \sum_{v < Y_k^i} \xi_{kv} \Big) > 1, \qquad i = 1, \ldots, q \tag{4}$$

Then the pricing problem consists in maximizing

$$\sum_{i=1}^{p} y_i \, \eta_i$$

subject to (2), (3) and (4).

Since we want to solve (1) via branch-and-bound, we need a branching scheme which does not conflict with the pricing procedure. Suppose there is a fractional variable x_j corresponding to the box $B(\ell^j, u^j)$. As usual, we branch by imposing either $x^j = 0$ or $x^j = 1$. These choices can be implemented as follows. Imposing $x^j = 1$ is equivalent to cover all positive points in $B(\ell^j, u^j)$. Therefore the master problem can be reformulated by dropping these points. Imposing $x^j = 0$ means that in the pricing problem we don't want the box $B(\ell^j, u^j)$ as a possible solution. To this aim it is enough that at least one of the $2\,n$ assignment variables $\zeta_{k\ell_k^j}$ and $\xi_{ku_k^j}$ identifying the box be different from 1, which can be accomplished by

$$\sum_{k=1}^{n} \left(\zeta_{k\ell_k^j} + \xi_{ku_k^j} \right) \le 2\,n - 1$$

In Figure 1 we show an instance with two features. There are 200 positive points

and as many negative ones. An optimal solution covering the positive points is displayed in the figure. The number of generated columns to reach optimality is 284. The initial solution consisted of 200 boxes (i.e., one for each point). The optimal boxes are the ones generated at iterations 1, 2, 4, 5, 6, 10, 45, 49 and 216. The overall CPU time has been 62 minutes, which is a rather large amount of time. This has to be ascribed to the fact that the max weighted box problems have been solved via the ILP model and not the special combinatorial algorithm developed in [1] which, according to the authors, exhibits a much higher computing performance. However, an integral solution with 11 boxes emerged already at the 7-th box generation and another one with 10 boxes at the 10-th generation. An optimal solution with 9 boxes appeared at the 102-nd iteration consisting of the boxes numbered 1, 2, 4, 5, 6, 10, 50, 86 and 102, after 20 minutes CPU time. Apparently the rest of the iterations and computing time is spent just to prove optimality.

References

[1] J. Eckstein, P.L. Hammer, Y. Liu, M. Nediak, and B. Simeone: The maximum box problem and its application to data analysis, *Computational Optimization and Applications*, **23**, 285-298 (2002).

Contributed Abstracts

Resources and transmission formats allocation in OFDMA networks

A. Abrardo, M. Belleschi, P. Detti

Dipartimento di Ingegneria dell'Informazione Università di Siena, via Roma 56 - 53100 Siena, Italy
abrardo@ing.unisi.it, {belleschi, detti}@dii.unisi.it

Key words: resource allocation, graph based models, approximation and heuristic algorithms

1 Introduction

In this paper, we address an optimization problem arising in the management of wireless network systems using the Orthogonal Frequency-Division Multiple Access (OFDMA) scheme. Such a radio technology has been proposed for the implementation of WiMax networks in [1], and is one of the most promising techniques for next generation of wireless systems. The OFDMA technique provides a subchannelization structure in which the overall frequency bandwidth of a block of transmission, i.e., a radio frame, is divided into a given set of orthogonal radio resources (each resource defined by a pair frequency/ time), called subcarriers, the basic units of resource allocation. Assigning the available subcarriers to the active users, with suitable transmission formats, in an adaptive manner is a viable method to achieve multi-user diversity: the propagation channels are independent for each user and thus the subcarriers that are in a deep fade for one user may be good ones for another. Many resource allocation algorithms in wireless communication systems have been designed for taking advantage of both the frequency selective nature of the channel and the multi-user diversity, but all of them exhibit a tradeoff between complexity and performance: low complexity algorithms, as the one presented in [2,5], tend to be outperformed by those requiring high computational loads, e.g. see [4,6].

In this work, a complexity and an approximation analysis is presented for the addressed allocation problem. Moreover, a simple approximation algorithm based on finding a minimum weighted b-matching on a suitable graph and two heuristic approaches are proposed. The heuristics are tested on random instances simulating a real world scenario.

2 System model and problem definition

The multi-format radio resource allocation problem (MF-RAP) that we address is a constrained minimization problem in which subcarriers and transmission formats must be assigned to users, in such a way that a given bit-rate is provided to each user and that the total transmission power is minimized. Let $N = \{1, \ldots, n\}$ be the set of the users. Given a user $i \in N$, we denote by R_i the transmission rate of user i. The frequency bandwidth of a radio frame is divided into orthogonal (i.e., not interfering) subcarriers: Let $M = \{1, \ldots, m\}$ be the set of the available subcarriers (i.e., resources). Let $Q = \{a_1, \ldots, a_p\}$ be the set of the possible *transmission formats* that can be used on a subcarrier, where $a_1, \ldots, a_p$ are positive integers with $a_1 < a_2 \ldots < a_p$. The selection of transmission format $a_q \in Q$ corresponds to the usage of a certain error correction code and symbol modulation that involves a spectral efficiency $\eta(q)$ expressed in bit/s/Hz. In other words, a user employing the format $a_q \in Q$ on a subcarrier of bandwidth B transmits with rate $R = B\eta(q)$. In practical transmission schemes, the spectral efficiency on the q-th format, $\eta(q)$, is an integer multiple of a given $\tilde{\eta}$, i.e, $\eta(q) = a_q\tilde{\eta}$. In the same way, the required transmission rate of user i, R_i, is expressed as an integer multiple of a certain fixed rate, i.e., $R_i = r_i B\tilde{\eta}$, with r_i an integer number. In this context, due to intra-cell interference, each subcarrier can be used by at most one user, and any given user can employ only a single format q to transmit on a subcarrier. Hence, by adopting the Shannon capacity as a measure of the achievable spectral efficiency on a given subcarrier, we obtain $\eta = \log_2(1 + \gamma)$ where γ is the signal to noise ratio (SNR). We assume that each link offers a spectral efficiency $\eta(q)$ that determines a fixed SNR target, i.e. $\gamma^{\mathrm{tgt}}(q) = 2^{\eta(q)} - 1$. The power $P_{i,j}(q)$ required to user i to transmit on subcarrier j using format q is computed as

$$P_{i,j}(q) = \gamma^{\mathrm{tgt}}(q)\frac{BN_0}{G_{i,j}} = \frac{BN_0}{G_{i,j}}(2^{\eta(q)} - 1) \tag{1}$$

where $G_{i,j}$ is the squared module of the channel gain between user i and the BS when using subcarrier j (this quantity includes the effects of distance-based attenuation, fading, beamforming etc.), and N_0 is the noise spectral density of the zero-mean thermal noise. A *feasible radio resource allocation* for MF-RAP consists in assigning resources (i.e., subcarriers) to users and, for each subcarrier-user pair, in choosing a transmission format, in such a way that (a) a bit-rate $R_i = r_i B\tilde{\eta}$ is provided to each user i, (b) at most one user is assigned to the same radio resource. MF-RAP consists in finding a feasible radio resource allocation minimizing the overall transmission power.

Let x_{ijq} be a binary variable equal to 1 if user i is assigned to radio resource j with format q (and 0 otherwise). An Integer Linear Programming formulation for MF-RAP is as follows.

$$\min \sum_{i\in N, j\in M, q\in Q} P_{i,j}(q)x_{ijq} \tag{2}$$

$$\sum_{i\in N, q\in Q} x_{ijq} \leq 1 \quad \forall j \in M \tag{3}$$

$$\sum_{j\in M, a_q\in Q} B\tilde{\eta}a_q x_{ijq} \geq B\tilde{\eta}r_i \quad \forall i \in N \tag{4}$$

$$x_{ijq} \in \{0,1\} \quad \forall i \in N, j \in M, a_q \in Q \tag{5}$$

3 Approximation results and heuristic algorithms

We prove that MF-RAP is strongly NP-hard even when only two transmission formats exists, i.e., $Q = \{a_1, a_2\}$, and $a_1 = 1$. Moreover, for this case, we show that the problem does not admit a polynomial-time approximation algorithm with approximation ratio bounded by a constant. An approximation tight ratio for a simple heuristic working on a particular graph is also established. Given an instance I of MF-RAP, the heuristic, that we call b-MATCH, basically consists in finding a *perfect weighted b-matching* on a complete bipartite graph $G = (V_1, V_2, E)$. In V_1 (in V_2) a node exists for each user in N (each resource in M). An extra dummy node, say z, exists in V_1. For each pair i, j, with $i \in V_1$ and $j \in V_2$, an arc (i, j) exists in E with weight $P_{i,j}(1)$ if $i \neq z$, and 0 otherwise. A value $b_i = \left\lceil \frac{r_i}{a_p} \right\rceil$ is assigned to each node i in $V_1 \setminus \{z\}$, a value $b_j = 1$ to each node j in V_2, and a value $b_z = m - \sum_{i\in N} \left\lceil \frac{r_i}{a_p} \right\rceil$ to the extra node z. Note that, since $a_p \geq a_q$, for $q = 1, \ldots, p$, $\left\lceil \frac{r_i}{a_p} \right\rceil$ is the minimum number of resources that have to allocate to user i, in any feasible solution of MF-RAP. Moreover, it must be $m - \sum_{i\in N} \left\lceil \frac{r_i}{a_p} \right\rceil \geq 0$, otherwise no feasible solution exists, too. Algorithm b-MATCH has two steps and works as follows. In the first step, a perfect b-matching with minimum weight is found on G, defining the set $S(i)$ of the resources to assign to each user. In the second step, the transmission formats are assigned as follows: For each user i, the resources in $S(i)$ are ordered in not decreasing order of powers $P_{i,j}(1)$; according to this order, the maximum transmission format a_p is assigned to the first $\left\lceil \frac{r_i}{a_p} \right\rceil - 1$ resources, and a transmission format of $a = \min\{a_q, a_q \in Q : a_q \geq r_i - (\left\lceil \frac{r_i}{a_p} \right\rceil - 1)a_p\}$ is assigned to the last resource in $S(i)$. In words, the minimum transmission format to achieve the required transmission rate is assigned to the last resource in $S(i)$. The first step of b-MATCH can be solved as a transportation problem, where nodes in V_1 are supply nodes and nodes in V_2 are demand nodes in $O(m \log m(mn + n \log n))$ (see [3]), the second step requires $O(m)$ steps in total.

For b-MATCH, we prove that (i) the following approximation ratio holds, and (ii) that the ratio is tight:

$$\frac{Z_H}{Z^*} \leq \frac{2^{\tilde{\eta}a_p} - 1}{2^{\tilde{\eta}a_1} - 1} \frac{a_1 \max\limits_{i\in N, j\in M} \{P_{i,j}(1)\}}{a_p \min\limits_{i\in N, j\in M} \{P_{i,j}(1)\}} \tag{6}$$

where Z_H is the value of solution produced by b-MATCH and Z^* is the value of an optimal solution.

For MF-RAP, we also propose and test two heuristic algorithms. Both algorithms leverage the information of the LP relaxation of the ILP formulation (2)–(5). Specifically, the first algorithm, called LP-ROUND, is based on rounding the optimal solution of the Linear Programming relaxation solution of (2)–(5). The second algorithm, denoted as b-MATCH-MOD, is an improvement of algorithm b-MATCH presented above. Both algorithms have two phases: In Phase 1, radio resources are allocated to users; in Phase 2, transmission formats are assigned to the allocated resources, in such a way that the rate required by the users are satisfied. In both the algorithms, Phase 2 consists in suitably applying an optimal dynamic programming procedure.

Computational experiences on random instances simulating a real world scenario are performed. They show that, in a comparison with a commercial state-of-the-art optimization solver, the proposed algorithms are effective in terms of solution quality and CPU times. Moreover, comparisons with alternatives proposed in the literature definitely assess the validity of the proposed approaches.

References

[1] Long Term Evolution of the 3GPP radio technology. `http://www.3gpp.org/Highlights/LTE/LTE.htm`.

[2] D. Kivanc, G. Li, and H. Liu, "Computationally efficient bandwidth allocation and power control for OFDMA," *IEEE Trans. Wireless Commun.,* vol. 2, no. 6, pp. 1150-1158, November 2003.

[3] Kleinschmidt, P., Schannath, H., *A strongly polynomial algorithm for the transportation problem*, Mathematical Programming 68 (1-3), pp. 1-13, 1995.

[4] I. Kim, I. Park, and Y. Lee, "Use of Linear Programming for Dynamic Subcarrier and Bit Allocation in MultiUser OFDM," *IEEE Trans. Vehic. Technol.,* Vol. 55, 1195-1207, 2006.

[5] W. Rhee and J. Cioffi, "Increase in capacity of multiuser OFDM system using dynamic subchannel allocation," *Proc. IEEE VTC 2000 Spring,* Tokio, Japan, pp. 1085-1089, May 2000.

[6] C. Y. Wong, R. S. Cheng, K. B. Letaief, and R. D. Murch, "Multiuser OFDM with adaptive subcarrier, bit, and power allocation," *IEEE J. Select. Areas Commun.,* Vol. 17, No. 10, pp. 1747-1758, Oct. 1999.

Modeling and solving Aircrafts Scheduling Problem in Ground Control

Ludovica Adacher, [a] Marta Flamini, [a,b]

[a] *Dip. di Informatica e Automazione, Università degli studi Roma Tre, Italy.*

[b] *Università Telematica Internazionale UNINETTUNO, Roma, Italy.*

Key words: Aircraft scheduling problem, job-shop scheduling, alternative graph

Abstract

In this paper we consider a real time aircraft scheduling problem in ground control, with fixed routes and the objectives of minimizing the number of tardy aircrafts or maximizing the number of on time aircrafts. Constraints of the problem are given by the safety rules. We propose an alternative graph model for the problem and heuristic solution procedures.

1 Introduction

Air Traffic Control tasks concern with (*i*) routing decisions, assigning a route to each aircraft from its current position to its destination and (*ii*) scheduling decisions, determining feasible aircrafts schedules with fixed routes, such that safety rules are satisfied. The increase of air traffic asks for the optimization of the usage of the existing resources at different levels. Moreover, the development of decision support systems could help human dispatchers, currently performing control operations, in optimizing real time traffic operations especially in case of congestion. In this paper we deal with a real time Ground control Aircraft Scheduling Problem with fixed routes ($GASP$, from now on). Ground control is, in particular, responsible for directing all ground traffic. This problem is hard to perform in terms of both designing adequate models and projecting efficient and effective solution techniques. Studies on the $GASP$ ([2], [3], [4]) usually suffer for a substantial lack of information due to the usage of very simplified models. Recent models ([1]) introduce an increased level of realism and incorporate a larger variety of constraints and possibilities, such as no-wait constraints, and earliness/tardiness penalty costs. We consider the objectives of minimizing the number of tardy aircrafts and of maximizing the number of on time aircrafts. The second objective function is motivated by the real nature of the problem in which also early departures or arrivals

could cause coordination discomforts in the Terminal Manoeuvering Area (TMA) management. We propose an alternative graph model [5] of the problem and some heuristic procedures producing fast solutions able to face the real time nature of the problem.

2 Problem Description

The layout of a TMA is composed by one or more runways, a taxiway network, parking bays and yards. Yards host aircrafts queue before the take-off. Parking bays are areas in which aircrafts are parked once landed. Runways are divided in segments (each between two consecutive intersections with the taxiway). When necessary, a stop bar regulates the transit from a taxiway to another. We consider both arriving and departing aircrafts, in a given time interval. We are given a nominal landing time and an arrival time for the arriving aircrafts. The nominal landing time is the minimum time instant in which the aircraft could be able to land and is considered as a problem variable since the landing instant could be postponed for optimization scopes. The arrival time represents the time instant in which the aircraft should reach the parking bay. For each departing aircraft we are given a departing time that is the time instant in which the aircrafts should take-off. The arrival times and departing times are considered as due dates for the aircrafts. The ground routes are fixed a priori for all the aircrafts. $GASP$ consists in scheduling aircrafts movings in the TMA. Problem constraints are given by the safety rules regulating the movings of the aircrafts in the TMA. Among the most important we can mention: the runway (all the segments) can be occupied only by one aircraft a time; each taxiway can be can be occupied only by one aircraft a time; a parking bay can host only one aircraft; when an aircraft waits at a stop bar, no other aircrafts can occupy the taxiway the stop bar lays on. We refer to two objective functions which are actually representative of the criteria driving human dispatchers decisions, namely the minimization of the number of tardy aircrafts and the maximization of the number of on time aircrafts, with respect to their due dates.

3 The job shop scheduling model with additional constraints

We model the $GASP$ as a job-shop scheduling problem with additional constraints, in which jobs are represented by aircrafts and resources by parts of the TMA. Resources we consider are runway segments, taxiway, parking bays, stop bars, yards. All such resources, but the yards, have unitary capacity and can not be occupied by two or more aircrafts at the same time. Yards are assumed to have infinitive capacity. An operation is the occupation of a resource by an aircraft. Operation processing time represents the time an aircraft need to run/occupy a resource. Standard job-shop constraints hold: precedence constraints between two consecutive operations, resources capacity constraints, no-preemption constraints. Additional constraints are due to the safety rules regulating spatial or temporal security dis-

tance between each pair of aircrafts moving in the TMA. For instance no-wait constraints regulate the usage of runway segments. Blocking constraints regulate the occupation of the taxiway preceding a stop bar. Two resources are *incompatible* if they can not be occupied at the same time. A conflict arises when two aircrafts have to occupy the same resource or two incompatible resources at the same time. A conflict resolution consists in fixing the precedence of an aircraft over the other in the usage of the requested resources. In a feasible solution all conflicts are resolved so that deadlocks are avoided and safety rules are respected.

4 The Alternative Graph model

We model the $GASP$ with the *Alternative Graph* formulation ([5]). The alternative graph is an effective model for representing and studying conflicts arising in the competitive request of a set of resources by several users. In the alternative graph model, the route of an aircraft is represented by a chain of nodes, each one representing an operation, that is the occupation of a resource. Fix arcs model the precedence constraints between two consecutive nodes of the same chain. All arcs are weighted with the processing time of the operation represented by the arc starting node. Capacity constraints are modeled by disjunctive arcs while conflicts on pairs of incompatible resources are modeled by pairs of alternative arcs. Disjunctive and alternative arcs model the two possible precedences between the aircrafts. In our problem we have several pairs of incompatible resources, each requiring a specific incompatibility model. A feasible solution for $GASP$ is represented by an acyclic graph in which for each alternative pair exactly one arc has been selected. A simple forward visit of the graph allows to compute nodes heads. The head of a node is the minimum time in which the operation represented by the node can start. The number of tardy (on time) aircrafts can be deduced by comparing the due dates with the heads of nodes representing *(i)* the arrival to the parking bays for the arriving aircrafts, and *(ii)* the arrival to the first segment of the runway for departing aircrafts.

5 Solution approach

Due to the real time nature of the problem we first compute different fast initial solutions, then we model them with the alternative graph to improve their quality by applying local search heuristics based on the alternative graph representation. Some of the initial solutions we consider, implement the following dispatching rules: (*i*) the precedence between two consecutive landing (departing) airplanes follows the FIFO rule respect to the landing (departing) time; (*ii*) a landing aircraft always precedes a departing aircraft if a conflict occurs on the runway; (*iii*) a departing aircraft precedes a landing one in taxiway, stop bars, parking bays. Such solutions represent a good estimation of the solutions currently produced by controllers in terms of both decisions and quality. We use them as reference values. Once an

initial solution has been constructed, it is modeled by the alternative graph and local search procedures are applied. The alternative graph, in fact, allows to rapidly estimate the effect produced by modifying the structure of a solution. The basic idea is to try and iteratively invert precedences between two aircrafts. After each inversion a forward visit of the graph detects the acyclicity of the graph and updates the nodes heads. An inversion is accepted only if the new solution is feasible and improves. Local search procedures differ in the criteria for choosing pairs of aircrafts at each iteration. We develop two heuristic procedures. mML heuristic inverts precedence between the aircraft with the minimum lateness preceding the aircraft with the maximum lateness, to reduce the number of tardy aircrafts. ET heuristic inverts precedence between the aircraft with minimum earliness preceding the aircraft with minimum tardiness, to improve the number of on time aircrafts. Very preliminary results show that the two heuristics produce an effective improvement with respect to the initial solutions. For those instances in which the initial solution does not improve in terms of objective function, that is the number of tardy aircrafts or the number of on time aircrafts, heuristics always produce a performance improvement in terms of mean tardiness or mean lateness.

References

[1] Adacher L., Pacciarelli D., Paluzzi D., Pranzo M., 2004. *Scheduling arrivals and departures in a busy airport*. Preprints of the 5th Triennial Symposium on Transportation Analysis, Le Gosier, Guadaloupe.

[2] Anagmostakis I., Clarke J., Bohme D., Volckers U., 2001. *Runway Operations Planning and control: Sequencing and Scheduling*. Journal of Aircraft, 38(6), 988–996.

[3] Beasley J.E., M. Krishnamoorthy, Y.M. Sharaiha, D. Abramson, 2000, *Scheduling aircraft landings - the static case*, Transportation Science, 34, 180–197.

[4] Bianco L., P. Dell'Olmo, S. Giordani, 1997. *Scheduling models and algorithms for TMA traffic management*, Modelling and simulation in air traffic management, eds. Bianco L. et al., Springer Verlag, 139–168.

[5] Mascis A., D. Pacciarelli, 2002. *Job shop scheduling with blocking and no-wait constraints* European Journal of Operational Research, 143 (3), 498–517.

Network design with SRG based protection

B. Addis, [a] G. Carello, [b] F. Malucelli [c]

[a] *Dipartimento di Informatica Università degli Studi di Torino*
C.So Svizzera, 185 10149 Torino
addis@di.unito.it

[b] *Dipartimento di Elettronica ed Informazione Politecnico di Milano*
Via Ponzio 34 - 20133 Milano
carello@elet.polimi.it

[c] *Dipartimento di Elettronica ed Informazione Politecnico di Milano*
Via Ponzio 34 - 20133 Milano
malucell@elet.polimi.it

Key words: Shared Risk Group, network design, shared protection, ILP, greedy

1 Introduction and problem statement

The design of resilient networks is a crucial problem in telecommunications and has obtained much attention by the optimization community. The most traditional setting considers link failures. The link capacity must be allocated at minimum cost so that in case of single or multiple link failures the demands can still be routed. When a single failure is accounted for, for each demand the capacity on a pair of disjoint paths (primary and spare) from the origin to the destination nodes must be allocated. If the protection is of dedicated type, the capacity allocated on each link must consider the sum of flows on the primary and spare paths using that link. While in the *shared protection* context , the capacity to be allocated on each link is given by the sum of the flows of the primary paths using that link plus the maximum of the flows on the spare paths using that link provided that the corresponding primary paths are disjoint., thus allowing a remarkable saving in comparison with the dedicated protection.

Multilayer networks require that protection considers more complex features than single link failures. The links of the logical layer network correspond to complex objects in lower layers, thus in case of a failure in the lower layers more than one link is affected in the upper one. This complexity is captured by the so called *Shared Risk Groups* (SRG). We can define a Shared Risk Group as the set of links of the logical layer that simultaneously fail in the case of a failure of one link of the lower layer. One link of the logical layer may belong to more than one SRG. This additional complexity usually spoils the network structure of the problem as one

SRG can include apparently uncorrelated links, just looking at the logical layer network.

Problems arising in the SRG network design context have been considered in recent works as for example [1] and [2] where some path and cut problems on colored graphs are analyzed from the computational complexity and approximability point of view. In [3] a network design of resilient networks with SRG is considered and a very involved mathematical model is proposed that cannot be solved by commercial optimization software.

We consider the network design problem in the presence of SRG. We propose mathematical models for the dedicated and shared protection. and also a simple constructive heuristic reporting some comparative computational results.

The network is represented by a directed graph $G = (N, A)$. The set of arcs corresponds to the set of links where capacity must be allocated. A set of traffic demands K is given, each defined by a triple (o_k, t_k, d_k) where o_k and t_k are the origin and the destination of d_k units of flow. The set $\mathcal{S}$ of shared risk groups is also given. Each SRG $s \in \mathcal{S}$ is a subset of A. To guarantee resilience to single arc failure, each arc represents a SRG, as well. We assume that the arc capacity is provided by installing transportation channels, each providing a capacity of λ flow units. The channel installation on arc (i, j) is c_{ij}.

The reliable network design problem consists in finding, for each demand k, two paths from the origin o_k to the destination d_k disjoint on the SRGs and allocate the capacity on the arcs at minimum cost. The capacity allocation depends on the type of applied protection. In the case of dedicated protection, the capacity to be allocated on a link (i, j) amounts to the sum of flows (primary or spare) traversing the link. In the case of shared protection, the capacity allocated on link (i, j) is given by the sum of flows of the primary paths traversing the link plus the maximum of the spare paths flows of demands that do not share SRG on their primary paths.

2 Mathematical models

The dedicated protection problem can be modeled using two sets of binary variables, describing the nominal and backup paths of each demand k, namely x_{ij}^k and y_{ij}^k that equal 1 if and only if the nominal or the back up paths, respectively, are routed on arc (i, j). Beside, integer variables z_{ij} give the number of channels installed on each arc (i, j). The model is the following:

$$\min \sum_{(i,j)\in A} c_{ij} z_{ij} \tag{1}$$

$$\sum_{(j,i)\in A} x^k_{ji} - \sum_{(i,j)\in A} x^k_{ij} = b^k_i \qquad \forall i \in N, k \in K \tag{2}$$

$$\sum_{(j,i)\in A} y^k_{ji} - \sum_{(i,j)\in A} y^k_{ij} = b^k_i \qquad \forall i \in N, k \in K \tag{3}$$

$$\sum_{k\in K} d_k \left(x^k_{ij} + y^k_{ij}\right) \leq \lambda z_{ij} \qquad \forall (i,j) \in A \tag{4}$$

$$x^k_{ij} + y^k_{hl} \leq 1 \, \forall k \in K, s \in \mathcal{S}, (i,j), (h,l) \in s \tag{5}$$

$$x^k_{ij}, y^k_{ij} \in \{0,1\}, z_{ij} \in \mathbb{Z}_+ \qquad \forall (i,j) \in A \tag{6}$$

where $b^k_i = -1$ if $i = o_k$, $b^k_i = 1$ if $i = t_k$ and is equal to 0 otherwise. Objective function (1) guarantees the minimum installation costs. Constraints (2) (and (3)) guarantee that there exists one and only one primary (and backup, respectively) path for each traffic demand. Constraints (5) force the two paths to be SRG disjoint for each demand. Equations (4) are arc dimensioning constraints.

In the shared protection model we use x^k_{ij} and y^k_{ij} binary variables, and integer variables z_{ij} with the same meaning of the dedicated case. In addition we consider binary variables v_{ks} which is equal to 1 if demand k is affected by SRG s, and continuous variables r^{ij}_{ks}, representing the amount of traffic of demand k which must be rerouted on arc (i,j) if SRG s occurs. The objective function (1) is kept, as well as constraints (2), (3) and (5). The following constraints are added to the basic model

$$\sum_{(i,j)\in s} x^k_{ij} \leq |s| v_{ks} \qquad \forall s \in \mathcal{S}, k \in K \tag{7}$$

$$r^{ij}_{ks} \geq d_k \left(v_{ks} + y^k_{ij} - 1\right) \quad \forall s \in \mathcal{S}, k \in K, (i,j) \in A \tag{8}$$

$$\sum_{k\in K} r^{ij}_{ks} + \sum_{k\in K} d_k x^k_{ij} \leq \lambda z_{ij} \, \forall (i,j) \in A, s \in \mathcal{S}. \tag{9}$$

We propose a greedy algorithm integrated in a Multistart framework. The greedy approach sequentially routes one demand at a time and dimensions the network, following a certain ordered which is randomly changed to provide in the iterations of the multistart approach. The key issue of the algorithm is the evaluation of the incremental costs, i.e. the increase in the capacity installation cost which the demand will cause if routed on the arc: for each arc the capacity installation cost is computed assuming that the considered demand is routed on the arc. The incremental cost are given by the difference between such cost and the current one. For each demand the algorithm finds the minimum incremental cost pair of primary and backup paths solving suitable ILP models. and routes the demand on graph and dimensioning the link capacity.

Table 1. Results on instances with dedicated protection

	CPLEX				Heuristic			
Instance	LB	UB	gap	CPU time	UB	gap UB	gap LB	CPU time
i_17_5	7506	7506	0.00%	0.33	7700	2.58%	2.58%	3.74
i_17_40	25350.47	25353	0.01%	263.23	28332	11.75%	11.76%	21.59
i_45-0_5	3348	3348	0.00%	8.19	3693	10.30%	10.30%	5.67
i_45-0_40	9342.85	10265	9.87%	t.l.	12569	22.45%	34.53%	45.16
i_45-1_5	3975	3975	0.00%	2.02	4435	11.57%	11.57%	5.63
i_45-1_40	12815.21	14102	10.04%	t.l.	17223	22.13%	34.39%	45.17
i_45-2_5	4311	4311	0.00%	4.73	4941	14.61%	14.61%	5.62
i_45-2_40	13931.12	15514	11.36%	t.l.	19013	22.55%	36.48%	47.90
i_45-3_5	2939	2939	0.00%	1.35	2940	0.03%	0.03%	5.67
i_45-3_40	9928.96	10765	8.42%	t.l.	12836	19.24%	29.28%	44.91
i_45-4_5	4238	4238	0.00%	1.94	4843	14.28%	14.28%	5.64
i_45-4_40	14474.29	15827	9.35%	t.l.	18966	19.83%	31.03%	44.89
average			4.09%	40.26		14.28%	19.24%	23.47
maximum			11.36%	263.23		22.55%	36.48%	47.90

Table 2. Results on instances with shared protection

	CPLEX first model		CPLEX second model				Heuristic	
	LB	1 hour	LB	1 hour	LB	First model	Second model	
instance	impr	gap	impr	gap	gap	UB gap	UB gap	CPU time
i_17-4_5	82.34%	0.00%	82.58%	0.00%	0.00%	12.50%	12.50%	8.90
i_17-4_40	39.53%	22.42%	39.66%	22.79%	2.44%	7.86%	10.22%	57.26
i_45-0_5	75.53%	67.76%	73.59%	60.69%	7.35%	2.56%	15.56%	25.55
i_45-0_40	17.45%	421.21%	12.72%	169.00%	5.60%	-36.99%	29.32%	295.47
i_45-1_5	67.12%	75.24%	66.30%	73.13%	2.43%	13.81%	18.06%	26.48
i_45-1_40	21.57%	318.82%	17.20%	135.75%	5.29%	-27.51%	35.96%	305.48
i_45-2_5	66.46%	93.08%	65.82%	81.20%	1.87%	4.78%	13.77%	31.47
i_45-2_40	15.29%	269.14%	11.99%	144.03%	3.79%	-20.47%	25.04%	291.01
i_45-3_5	75.54%	57.30%	71.65%	83.10%	13.71%	61.93%	61.21%	25.97
i_45-3_40	20.68%	345.47%	17.63%	171.94%	3.70%	-28.02%	22.44%	285.00
i_45-4_5	63.84%	94.90%	62.73%	94.99%	2.98%	11.16%	14.52%	31.06
i_45-4_40	-	-	14.83%	131.16%		-	36.39%	284.08
average	49.58%	160.49%	44.73%	97.31%		0.14%	24.58%	138.98
maximum	82.34%	421.21%	73.59%	171.94%		61.93%	61.21%	305.48

3 Computational results

The models and the proposed heuristic have been tested on a set of a randomly generated instances with 10 nodes and 4 SRGs. Models have been solved with CPLEX 11.0.1 with a time limit of 3600 seconds. Both CPLEX and heuristic, with 30 multistart iterations, have been run on a Xeon (2.0GHz, 4Gb RAM).

Table 1 reports the results of the dedicated protection case. CPLEX solves to optimality 7 instances, those with five demands over 12, in reasonable CPU time. The gap is limited for the five instances for which optimality is not proved. The heuristic algorithm provides gaps with respect to the integer solution of about 14% inaverage. Although the computational time increases on the 40 demands instances.

The shared protection case is reported in Table 2. The exact approach proves optimality in only one instance. The number of demands affects significantly the final gap. The continuous relaxation is quite poor. The heuristic computational time are

reasonable, never rising above 6 minutes. Computational experiments show that our models cannot solve the instances of both problems even with a small number of nodes. The multistart approach, although fast, is affected by sensible gaps.

References

[1] D. Coudert, P. Datta, H. Rivano, M.-E. Voge, "Minimum color problems and shared risk resource group in multilayer networks", Rapport de Recherche ISRN I3S/RR-2005-37-FR.

[2] P. Datta, A. K. Somani "Diverse routing for shared risk resource groups (SRRG) failures in WDM Optical networks", BROADNETS'04.

[3] L. Shen, X. Yang, B. Ramamurthy, "Shared risk link group (SRLG)-diverse path provisioning under hybrid service level agreements in wavelength-routed optical mesh networks", IEEE/ACM Trans. on Networking 13(4), 2005, 918-931

Scheduling Problems with Unreliable Jobs and Machines

A. Agnetis, [a] P. Detti, [a] P. Martineau, [b] M. Pranzo, [a]

[a] *Università di Siena*
via Roma 56 - 53100 Siena (ITALY) , Italy
{agnetis, detti, pranzo}@dii.unisi.it

[b] *Ecole Polytechnique de l'Université de Tours, France*
patrick.martineau@univ-tours.fr

Key words: Scheduling, Approximation, Parallel Machines

1 Introduction

A set of jobs $J = \{J_1, \ldots, J_n\}$ must be performed on a system consisting of m parallel, identical machines. Each job must be assigned to a single machine and a machine can process one job at a time. Jobs and/or machines are *unreliable*, i.e., while a job is being processed by a machine, a *failure* can occur, which implies losing all the work which was scheduled but not yet executed by the machine. Here we investigate the case in which failures depend on job or machine characteristics, but not both.

When failures are job-related, each job J_i is characterized by a certain *success probability* π_i (independent from other jobs) and a *reward* r_i, which is obtained if the job is successfully completed. The problem is to find a schedule of jobs on the m machines that maximizes total expected reward. In the following, we refer to this problem as *Unreliable Job Scheduling Problem*, denoted by UJP(m).

A different scenario is when the failure process depends on the machines. In this case, jobs are characterized by a certain length (duration) L_i, and a reward r_i. Machines are characterized by a function $P(t)$ that represents the probability of the machine still being on at time t. The problem is again to maximize expected reward, and we call this problem *Unreliable Machine Scheduling Problem*, denoted by UMP(m).

2 Job-related failures

Let σ_k be a sequence of jobs assigned to machine k, and let $\sigma_k(j)$ be the job in j-th position in σ_k. A feasible solution $\zeta = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ for UJP(m) is an

assignment and sequencing of the n jobs on the m machines. Considering that a job can be processed only if no failure occurred on the machine till then, if h jobs are sequenced on machine M_k by σ_k, the expected reward $ER[\sigma_k]$ is given by

$$ER[\sigma_k] = \pi_{\sigma_k(1)} r_{\sigma_k(1)} + \pi_{\sigma_i(1)} \pi_{\sigma_i(2)} r_{\sigma_k(2)} + \ldots + \pi_{\sigma_k(1)} \ldots \pi_{\sigma_i(h-1)} \pi_{\sigma_k(h)} r_{\sigma_k(h)} \quad (1)$$

and $ER[\zeta] = ER[\sigma_1] + ER[\sigma_2] + \ldots + ER[\sigma_m]$.UJP($m$) consists in finding a solution ζ^* that maximizes the total expected reward. We will denote by z^* the value of the optimal solution.

It is easy to show that the single-machine case $UJP(1)$ can be efficiently solved by sequencing the jobs in nonincreasing order of the following Z*-ratio*:

$$Z_i = \frac{\pi_i r_i}{1 - \pi_i} \quad (2)$$

In [1], a reduction from PRODUCT PARTITION to $UJP(2)$ is presented. A recent complexity result on PRODUCT PARTITION by Ng et al. [3] implies that $UJP(m)$ is strongly NP-hard for any fixed $m \geq 2$. Note that UJP(m) consists in deciding how to partition the n jobs among the m machines, since on each machine the sequencing is then dictated by the Z-ordering.

A simple heuristic list-scheduling algorithm (LSA) for UJP(m) is the following. Sort all jobs by nonincreasing Z-ratios, and then assign them to the machines, always assigning the next job in the list to the machine having the current largest cumulative probability. Let z^H be the value of the solution produced by LSA.

In this paper we analyze the worst-case behavior of LSA when $m = 2$. In particular, we first address the special case in which all jobs have the same Z-ratio, and then extend the result to general values of Z_i. This is the same approach used by Kawaguchi and Kyan [2] to prove that any list scheduling algorithm is $\frac{1+\sqrt{2}}{2} = 1.207$-approximate for $Pm||\sum w_i C_i$, when the jobs are ordered by nonincreasing ratios p_i/w_i. However, our argument is actually simpler than the one used by Kawaguchi and Kyan.

First observe that if all Z_i are equal, with no loss of generality we can assume $Z_i = 1$, so that $r_i = (1 - \pi_i)/\pi_i$ for all jobs J_i. The contribution of machine M_k to total expected revenue simplifies to

$$ER[\sigma_k] = (1 - \pi_{\sigma_k(1)}) + (1 - \pi_{\sigma_k(2)}) \pi_{\sigma_k(1)} + \ldots + (1 - \pi_{\sigma_k(h)}) \pi_{\sigma_k(1)} \ldots \pi_{\sigma_k(h-1)} \quad (3)$$

$$= 1 - \pi_{\sigma_k(1)} \pi_{\sigma_k(2)} \ldots \pi_{\sigma_k(h-1)} \pi_{\sigma_k(h)} \quad (4)$$

Notice that (4) does not depend on the actual sequencing, but only on the set of jobs assigned to M_k. Hence, there being only two machines, M_1 and M_2, we can rewrite the objective function as:

$$ER = 2 - \prod_{i \in M_1} \pi_i - \prod_{i \in M_2} \pi_i \quad (5)$$

Now, (5) is largest when the two product terms are as balanced as possible. In order to derive a lower bound for z^H, we must therefore consider the most unbalanced solution which can be produced by LSA.

Denote by P be the product of all jobs probabilities, i.e., $P = \prod_{i \in J} \pi_i$, and by π_1 the smallest probability. Note that since all jobs have the same Z-ratio, they are randomly sequenced by LSA. Since LSA always selects the machine with the current largest cumulative probability, it can never happen that the ratio between the two current cumulative probabilities is smaller than π_1. So, the most unbalanced situation that may occur is when the last job being scheduled is J_1, and before that the two machines were perfectly balanced. In this case one machine's expected revenue is $1 - \pi_1\sqrt{P/\pi_1}$ and the other is $1 - \sqrt{P/\pi_1}$, so that:

Theorem 2.1 *When $Z_i = 1$ for all i, $z^H \geq 2 - \sqrt{P\pi_1} - \sqrt{P/\pi_1}$.*

Now let us turn to upper bounds. For sure, an upper bound is obtained assuming that the two machines are perfectly balanced, i.e. the final cumulative probability is $\sqrt{P}$ for both machines.

Theorem 2.2 *When $Z_i = 1$ for all i, $z^* \leq 2 - 2\sqrt{P}$*

A stronger (tighter) upper bound can be obtained if π_1 is indeed very small. If π_1 is smaller than the product of all the other probabilities, the best solution would be to put J_1 on M_1 and all the other jobs on M_2:

Theorem 2.3 *When $Z_i = 1$ for all i, if $\pi_1^2 \leq P$, then $z^* = 2 - \pi_1 - P/\pi_1$.*

From these results, it is possible to prove the following theorem.

Theorem 2.4 *When $Z_i = 1$ for all i, the value z^H of the solution produced by LSA is such that $\frac{z^H}{z^*} \geq \frac{2+\sqrt{2}}{4} \simeq 0.8535$.*

Since the result can be extended to the general case, Theorem 2.4 indeed holds for any instance of $UJP(2)$. Moreover, the bound can be shown to be tight.

3 Machine-related failures

Turning to problem $UMP(m)$, we investigate the case in which machine failures are exponentially distributed, i.e., the probability of the machine being up at time t is given by $P(t) = e^{-\lambda t}$, where λ is the average number of failures per time unit. For the well-known properties of the exponential, this means that if a machine is up when job i is started, the probability of being successfully carried out is $\pi_i = e^{-\lambda L_i}$.

In this scenario, we can think of two sensible performance measures: (i) the expected *number* of jobs, and (ii) the expected *total work* (time) done. In the former case, we can simply set the reward $r_i = 1$ for each i. In the latter case, $r_i = L_i$ for each i.

Let us order all jobs in SPT order, i.e., in nondecreasing order of job length, and let us consider again a list scheduling algorithm, which assigns the next job in the list to the machine that has currently received the smallest workload. The following result can be proved:

Theorem 3.1 *If either*

- $r_i = 1$ *for all* i
- $r_i = L_i$, *and* $L_i \geq 1/\lambda$ *for all* i

then LSA finds an optimal solution to $UMP(m)$

Note that the quantity $1/\lambda$ represents the mean time between failures. If the job lengths can be smaller than $1/\lambda$, the problem is indeed difficult:

Theorem 3.2 *If* $r_i = L_i$, $UMP(2)$ *is NP-hard.*

References

[1] Agnetis, A., Detti, P., Pranzo, M. and Sodhi M.S., Sequencing unreliable jobs on parallel machines, *Journal of Scheduling*, 2009, 12, 45–54.

[2] Kawaguchi, T., Kyan, S., Worst case bound of an LRF schedule for the mean weighted flow-time problem, *SIAM J. Computing*, vol. 15, 4, 1986.

[3] Ng, C.T., Barketau, M.S., Cheng, T.C.E., Kovalyov, M.Y., "Product partition" and related problems of scheduling and systems reliability: Computational complexity and approximation, *European Journal of Operational Research*, 207, 601–604, 2010.

Models and Algorithms for the Bin Packing Problem with Fragile Objects

M.A. Alba Martinez, [a] F. Clautiaux, [b] M. Dell'Amico, [a] M. Iori, [a]

[a] *University of Modena and Reggio Emilia,*
Via Amendola 2, 42122 Reggio Emilia, Italy
{manuel.alba, mauro.dellamico, manuel.iori}@unimore.it

[b] *Université des Sciences et Technologies de Lille,*
INRIA Lille Nord Europe, Parc de la Haute Borne, 59655 Villeneuve d'Ascq, France
francois.clautiaux@univ-lille1.fr

Key words: Bin Packing Problem, Fragile Objects, Column Generation, Mathematical Models

1 Introduction

In the *Bin Packing Problem with Fragile Objects* (BPPFO), we are given n objects, each having weight w_j and fragility f_j ($j = 1, \ldots, n$), and a large number of uncapacitated bins. The aim is to pack all objects in the minimum number of bins, in such a way that in each bin the sum of the object weights is less than or equal to the smallest fragility of an object. More formally, let $J(i)$ denote the set of objects assigned to a bin i in a given solution. The solution is feasible if for any bin i:

$$\sum_{j \in J(i)} w_j \leq \min_{j \in J(i)} \{f_j\}. \tag{1}$$

The BPPFO is clearly NP-complete, because it generalizes the classical *Bin Packing Problem* (BPP). In the BPP we are given n objects of weight w_j ($j = 1, \ldots, n$) and a large number of bins of capacity C, with the aim of packing the objects in the minimum number of bins without exceeding the bin capacity. The BPP is a particular BPPFO where all object fragilities are set to C.

The BPPFO is important in telecommunications, because it models the allocation of cellular calls to frequency channels (see Chan et al. [2] and Bansal et al. [1]). In *Code Division Multiple Access* (CDMA) systems, a limited number of frequency channels is given. It is possible to assign many calls to the same channel, because each channel has a capacity much larger than the bandwidth requirement of a single call. Such assignment may, however, produce interferences among the calls and may result in a loss of quality of the communication. Indeed each call is characterized by a certain noise that it produces, and by a certain tolerance with respect to

the total noise in the channel. It is required that the total noise in a frequency channel does not exceed the tolerance of any call assigned to the channel. To model this telecommunication problem as a BPPFO, it is enough to associate each frequency channel to a bin, and each call to an object having weight equal to the call noise and fragility equal to the call tolerance. The solution of the BPPFO gives the minimum number of frequency channels required to allocate all calls.

The literature on the BPPFO is still small. Chan et al. [2] presented approximation schemes for the on-line version of the problem. Bansal et al. [1] proposed approximation schemes and a so-called *fractional lower bound*, which runs in $O(n)$ and is quite effective in practice. Clautiaux et al. [3] developed a large number of lower and upper bounds, built a benchmark set of challenging instances and presented the first computational results for the problem.

This contribution extends the results by Clautiaux et al. [3], and is devoted to the presentation of mathematical models (see Section 2), exact algorithms and related computational performances (see Section 3).

2 Mathematical Models for the BPPFO

We suppose that w_j and f_j are positive integers ($j = 1, \ldots, n$), and that objects are sorted according to non-decreasing values of fragility, breaking ties by non-increasing values of weight. The first model that we propose is made by a compact number of variables and constraints, and is obtained as follows. We define y_i as a binary variable taking value 1 if object i is the object with smallest fragility in the bin in which it is packed, 0 otherwise ($i = 1, \ldots, n$). We also define x_{ji} as a binary variable taking value 1 if object j is assigned to the bin having object i as object with smallest fragility, 0 otherwise ($i = 1, \ldots, n, j = i + 1, \ldots, n$). The BPPFO can be modeled as:

$$\min \sum_{i=1}^{n} y_i \tag{2}$$

$$y_j + \sum_{i=1}^{j-1} x_{ji} = 1 \qquad j = 1, \ldots, n \tag{3}$$

$$\sum_{j=i+1}^{n} w_j x_{ji} \leq (f_i - w_i) y_i \qquad i = 1, \ldots, n \tag{4}$$

$$x_{ji} \leq y_i \qquad i = 1, \ldots, n, j = i + 1, \ldots, n \tag{5}$$

$$y_i \in \{0, 1\} \qquad i = 1, \ldots, n \tag{6}$$

$$x_{ji} \in \{0, 1\} \qquad i = 1, \ldots, n, j = i + 1, \ldots, n. \tag{7}$$

Constraints (3) require that either an object is the one of smallest fragility in its bin, or it is assigned to a bin containing an object with smaller fragility. Constraints (4) impose that the sum of the weights of the objects packed in a bin does not exceed the smallest fragility in the bin. Constraints (5) are used to tighten the model linear relaxation.

The second model we propose builds upon the classical set covering formulation. We define a *pattern* as a feasible combination of objects. We describe the pattern, say p, by a column $(a_{1p}, \ldots, a_{jp}, \ldots, a_{np})^T$, where a_{jp} takes value 1 if object j is in pattern p, 0 otherwise. Let P be the set of all valid patterns, i.e., the set of patterns p for which:

$$\sum_{j=1}^{n} w_j a_{jp} \leq \min_{j=1,\ldots,n} \{f_j a_{jp}\}. \tag{8}$$

Let z_p be a binary variable taking value 1 if pattern p is used, 0 otherwise ($p \in P$). The BPPFO can be modeled as:

$$\min \sum_{p \in P} z_p \tag{9}$$

$$\sum_{p \in P} a_{jp} z_p \geq 1 \qquad j = 1, \ldots, n \tag{10}$$

$$z_p \in \{0, 1\} \qquad \forall p \in P. \tag{11}$$

Constraints (10) impose that each object j is packed in at least one bin. As the number of possible patterns may be very large, even solving the linear relaxation of model (9)–(11) is difficult. We approach this problem by means of a column generation method. We initialize the model with a subset of patterns and associate dual variables π_j ($j = 1, \ldots, n$) to constraints (10). In an iterative way, we look for a feasible pattern with negative reduced cost. If we find it, we add it to the model and reiterate, otherwise we terminate.

The existence of a pattern that is feasible, with respect to (8), and has negative reduced cost can be determined by solving a particular *0–1 Knapsack Problem with Fragile Objects* (KP01FO). In the KP01FO, we are given n objects with profit π_j, weight w_j and fragility f_j ($j = 1, \ldots, n$) and a single uncapacitated bin, with the aim of determining a subset of objects whose total weight does not exceed the fragility of any object in the bin, according to (1), and whose total profit is largest.

3 Algorithms and Preliminary Computational Considerations

We made use of the above models to build exact algorithms for the BPPFO. We also implemented polynomial lower and upper bounds (whose explanation is not included in this extended abstract for sake of conciseness) to speed up the performance of the exact algorithms. We tested the algorithms on the benchmark set proposed by Clautiaux et al. [3], that consists of 675 instances with 50, 100 and 200 objects and is available at `http://www.or.unimore.it/resources.htm`. Only preliminary computational results are available.

We first solved the compact model (2)–(7) by using Cplex 12. This proved to be very effective on instances with 50 objects, but very poor on larger instances.

Then, we implemented a classical branch-and-bound in which an enumeration scheme attempts at each level the assignment of an object to an open bin. At each node we invoke an implementation of the fractional lower bound by Bansal et al. [1]

that we modified so as to take into consideration the branching decisions adopted by the enumeration scheme. This algorithm improved the results obtained by the compact model, but still was unable to solve all instances with just 50 objects, and left unsolved many larger instances.

We finally implemented a branch-and-price based on the formulation (9)–(11). We solved the KP01FO by means of an integer linear mathematical model and a dynamic programming algorithm, and used a classical branching scheme to try to produce an integer solution whenever the outcome of the column generation was fractional. This algorithm turned out to be quite slow, compared to the combinatorial branch-and-bound, on the small instances, but allowed to close to optimality many larger instances that were left open by the two previous algorithms.

As future work we intend to improve the compact model by making use of cutting planes, in a branch-and-cut fashion. We also intend to improve the performance of the branch-and-price by attempting more enhanced strategies to branch, to fathom nodes and to find negative reduced cost patterns.

References

[1] N. Bansal, Z. Liu, and A. Sankar. Bin-packing with fragile objects and frequency allocation in cellular networks. *Wireless Networks*, 15:821–830, 2009.

[2] W.T. Chan, F.Y.-L. Chin, D. Ye, G. Zhang, and Y. Zhang. Online bin packing of fragile objects with application in cellular networks. *Journal of Combinatorial Optimization*, 14:427–435, 2007.

[3] F. Clautiaux, M. Dell'Amico, M. Iori, and A. Khanafer. Lower and upper bounds for the bin packing problem with fragile objects. Technical report, DISMI, University of Modena and Reggio Emilia, Italy, 2010.

New upper bound for the number of maximal bicliques of a bipartite graph

A. Albano, [1] Do Lago A.P.

DCC - Instituto de Matemática e Estatística, Universidade de São Paulo
{albano,alair}@ime.usp.br

Abstract

Given a bipartite graph, we present an upper bound for its number of maximal bicliques as the product of the numbers of maximal bicliques of two appropriate subgraphs. We show an infinite family of graphs for which our bound is sharper than the only non-trivial known bound for general bipartite graphs. In an infinite subfamily, the previous bound is exponential while ours is polynomial. We also discuss complexity aspects.

Key words: maximal bicliques, convex bipartite graph, consecutive ones property

1 Introduction

Let $G = (U, W, E)$ be a undirected bipartite graph with vertex set $U \cup W$. A *biclique* is a pair (A, B) with $A \subseteq U$ and $B \subseteq W$ such that $uw \in E$ for every $u \in A$, $w \in B$. The sets A and B are called, respectively, *left* and *right biclique coordinates*. We call (A, B) *maximal* if for every biclique (A', B') the following holds: $A' \supseteq A$ and $B' \supseteq B \Rightarrow A' = A$ and $B' = B$. If $S \subseteq U \cup W$, we denote by $G[S]$ the subgraph of G induced by S. Let $W = \{w_1, \ldots, w_r\}$ and π be a permutation of $\{1, 2, \ldots, r\}$ acting on W. We say that a vertex $u \in U$ *has the convex neighborhood property (CNP) with respect to* π if for every $w_{\pi(i)}$ and $w_{\pi(j)}$ adjacent to u with $i < j$, u is adjacent to every $w_{\pi(k)}$ such that $i \leq k \leq j$. The *convexity of* G *(over* W*) with respect to* π is the number of vertices $u \in U$ that have the CNP with respect to π, and is denoted $conv(G, \pi)$. The convexity (over W) of G is defined to be $conv(G) = \max_{\pi \in \mathfrak{S}_r} \{conv(G, \pi)\}$, where $\mathfrak{S}_r$ is the set of all permutations of $\{1, 2, \ldots, r\}$. This definition of convexity generalizes the convex property: a bipartite graph (U, W, E) is *convex* (over W) if $conv(G) = |U|$. For any vertex v, we denote by $N(v)$ the set of its neighbors. We denote by $^{\triangleright} : \mathcal{P}(U) \to \mathcal{P}(W)$ and $^{\triangleleft} : \mathcal{P}(W) \to \mathcal{P}(U)$ the *consensus functions*: $^{\triangleright}(A) = \cap_{u \in A} N(u)$, for

[1] Financial support by CNPq (Proc. number 132590/2009-3)

$A \subseteq U$, and ${}^{\triangleleft}(B) = \cap_{w \in B} N(w)$ for $B \subseteq W$. Instead of writing ${}^{\triangleright}(A)$ and ${}^{\triangleleft}(B)$, we write $A^{\triangleright}$ and $B^{\triangleleft}$.

Classes of bipartite graphs with a polynomial number of maximal bicliques admit a polynomial-time solution to the NP-hard maximum-edge biclique problem. Another application of this work is in association rule discovery, which is a main topic in data mining. Sets T, I of transactions and items, and a relation $R \subseteq T \times I$ are given, and the determination of *frequent itemsets* and their *support* plays a vital role in rule discovery. The so-called *frequent closed itemsets*, which are maximal biclique coordinates satisfying a size constraint, can be used to derive the support of all frequent itemsets [7].

The following proposition first appeared in [8], was also mentioned in [3] and has been more recently presented in [5], which is a very clear presentation of a few known definitions and results of [4] in terms of graph theory.

Proposition 1 *Let (U, W, E) be a bipartite graph. Then, (A, B) is a maximal biclique if and only if $A^{\triangleright} = B$ and $B^{\triangleleft} = A$. Furthermore, the set of left coordinates of maximal bicliques is precisely the image set of ${}^{\triangleleft}$. Dually, the set of right coordinates of maximal bicliques is precisely the image set of ${}^{\triangleright}$.*

One consequence of the above proposition is the following.

Corollary 1 *Every maximal biclique of a bipartite graph (U, W, E) is uniquely determined by one of its coordinates, and therefore the number of maximal bicliques of a bipartite graph is at most $2^{\min\{|U|,|W|\}}$, in general, and at most $\frac{|W|(|W|+1)}{2} + 1 = O(|W|^2)$ if (U, W, E) is convex over W.*

The graph obtained by removing a perfect matching from the complete bipartite graph $K_{n,n}$ is called *crown graph*. For $n \geq 3$, such graphs are not convex over W, nor U, and the bound $2^{\min\{|U|,|W|\}}$ is tight.

2 Decomposition result, new upper bound and comparisons

The following theorem is an application of a stronger result by Ganter and Wille [4, p.77] to the particular case we are interested here.

Theorem 1 *Let $G = (U, W, E)$ be a bipartite graph and let $U = U_1 \cup U_2$. The number of maximal bicliques of G is at most the product of the number of maximal bicliques of $G_1 = G[U_1 \cup W]$ and $G_2 = G[U_2 \cup W]$. Furthermore, this bound is tight.*

Proof: Let $\mathfrak{B}$ be the set of all maximal bicliques of G and let $\mathfrak{B}_1$ ($\mathfrak{B}_2$) be the set of maximal bicliques of G_1 (G_2). Denote by ${}^{\triangleright}$ and ${}^{\triangleleft}$ the consensus functions of G and by ${}^{\triangleright}_i$ and ${}^{\triangleleft}_i$ the consensus functions of G_i for $i \in \{1, 2\}$. Define $\varphi : \mathfrak{B} \to \mathfrak{B}_1 \times \mathfrak{B}_2$ by $\varphi((A, B)) = ((A \cap U_1, (A \cap U_1)^{\triangleright}), (A \cap U_2, (A \cap U_2)^{\triangleright}))$. Indeed, φ is well defined. Suppose $(A, B) \in \mathfrak{B}$. Then, $A \cap U_1$ is the set of vertices in G_1 adjacent to each $b \in B$. Therefore, $A \cap U_1$ is a G_1 maximal biclique left coordinate. By Proposition 1, it follows that $(A \cap U_1, (A \cap U_1)^{\triangleright})$ is a maximal biclique of G_1. Analogously, $(A \cap U_2, (A \cap U_2)^{\triangleright})$ is a maximal biclique of G_2. To show that φ

is injective, suppose $(A_1, B_1), (A_2, B_2)$ are such that $\varphi((A_1, B_1)) = \varphi((A_2, B_2))$. Then $A_1 \cap U_1 = A_2 \cap U_1$ and $A_1 \cap U_2 = A_2 \cap U_2$. Since $U_1 \cup U_2 = U$, it follows that $A_1 = A_2$. Proposition 1 implies that $B_1 = B_2$. For graphs attaining this upper bound, consider crown graphs. $\square$

Choosing π that maximizes $conv(G, \pi)$, and defining $U_1 = \{u \in U \mid u$ has the CNP with respect to $\pi\}$, and $U_2 = U \setminus U_1$, a straight application of Theorem 1 generalizes the final statement in Corollary 1 to the following:

Corollary 2 *Let $G = (U, W, E)$ be a bipartite graph of convexity $\mathfrak{C}$ over W. Then, the number of maximal bicliques of G does not exceed* $2^{|U|-\mathfrak{C}}(\frac{|W|(|W|+1)}{2} + 1) = O(2^{|U|-\mathfrak{C}} \cdot |W|^2)$.

Notice that the bound presented above can be improved, since $2^{|U|-\mathfrak{C}}$ bounds the number of maximal bicliques of a subgraph that can have low arboricity (see below), and therefore a better bound could be used with our decomposition argument. Also, $|W|$ could be replaced by $|\cup_{u \in U_1} N(u)|$.

Now, we exhibit a quite known upper bound for the number of maximal bicliques. Before, we need the following definitions. A graph $G = (V, E)$ is said to be $k-$*forest-decomposable* if there exists $E_1, \ldots, E_k \subseteq E$ such that each subgraph induced by E_i is acyclic and $\bigcup_{i=1}^{k} E_i = E$. The *arboricity* of G is the least integer k such that G is k-forest-decomposable. The arboricity of a graph can be given by Nash-Williams [6] formula: $a(G) = \max_H \lceil \frac{m(H)}{n(H)-1} \rceil$, where H ranges over all subgraphs and $m(H)$ and $n(H)$ denote the number of edges and vertices of H, respectively. The following theorem is directly implied by work of Eppstein [2].

Theorem 2 *The number of maximal bicliques of an n-vertex bipartite graph of arboricity a is at most $2^{2a} \cdot n$.*

We now construct a bipartite graph $G = (U, W, E)$ of convexity $> \frac{2|U|}{3}$ such that the bound given in Corollary 2 is sharper than the bound above. Let $U = \{u_1, \ldots, u_k\}$ and $W = \{w_1, \ldots, w_k\}$ be vertex sets, let $n = 2k$ be the number of vertices, and let $0 \leq c \leq 1$ be a parameter (to be determined later). For $1 \leq i \leq ck$, u_i is adjacent to every vertex in $\{w_1, \ldots, w_i\}$ (these vertices u_i correspond to a convex subgraph). For $ck < i \leq k$, u_i is adjacent to $\{w_{ck+1}, \ldots, w_r\} \setminus \{w_i\}$. No other adjacencies exist. Let $\mathfrak{C}$ be the convexity of this graph. Clearly, $\mathfrak{C} \geq ck$. Let a be its arboricity. By Nash-Williams formula, setting $U_1 = \{u_1, \ldots, u_{ck}\}, W_1 = \{w_1, \ldots, w_{ck}\}$ and choosing $H = G[U_1 \cup W_1]$, one has $a \geq \lceil \frac{(1+\frac{cn}{2})\frac{cn}{4}}{cn-1} \rceil > \frac{1+\frac{cn}{2}}{4} \Rightarrow 2a > \frac{cn}{4}$. Now, $|U| - \mathfrak{C} \leq k - ck = \frac{n}{2}(1-c)$. Setting $c = \frac{2}{3} + \epsilon$ ($\epsilon > 0$), we have

$$|U| - \mathfrak{C} \leq \frac{n}{2}(1-c) = (\frac{2}{3} - 2\epsilon)\frac{n}{4} < (\frac{2}{3} + \epsilon)\frac{n}{4} < 2a.$$

Next table shows the upper bounds by [2] and by Corollary 2, for some values of c. Our bound is cubic for $c = 1 - \frac{\log_2 |U|}{|U|}$, but previous bound is exponential.

$\lvert U_2\rvert = \lvert U \setminus U_1\rvert$	c	Upper bound by [2]	Upper bound by Corollary 2
$k(1/3-\epsilon)$	$2/3+\epsilon$	$\Omega(2^{(2/3+\epsilon)\frac{n}{4}}n)$	$O(2^{(2/3-2\epsilon)\frac{n}{4}}n^2)$
$k/6$	$5/6$	$\Omega(32^{n/24}n)$	$O(4^{n/24}n^2)$
$\log_2 k$	$1-\frac{\log_2 k}{k}$	$\Omega(2^{(1-\log_2 k/k)\frac{n}{4}}n)$	$O(n^3)$

3 Complexity and open problem

Deciding if a bipartite graph is convex can be solved in polynomial time [1]. However, determining the convexity of a bipartite graph is hard, since we obtained a polynomial-time reduction from a variant of the TSP.

Theorem 3 *The problem of determining whether a bipartite graph has convexity at least c is NP-complete.*

This naturally leads to the following problem.

Problem 1 *To obtain approximation results for determining convexity.*

References

[1] K. S. Booth and G S. Lueker. Linear algorithms to recognize interval graphs and test for the consecutive ones property. In *Proc. of 7th annual ACM symposium on Theory of computing*, STOC '75, pages 255-265, NY, USA, 1975. ACM.

[2] D. Eppstein. Arboricity and bipartite subgraph listing algorithms. *Inf. Process. Lett.*, 51(4):207-211, 1994.

[3] M. Farach-Colton and Y. Huang. A linear delay algorithm for building concept lattices. In *Proceedings of the 19th annual symposium on Combinatorial Pattern Matching*, CPM '08, pages 204-216, Berlin, Heidelberg, 2008. Springer-Verlag.

[4] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin/Heidelberg, 1999.

[5] E. Kayaaslan. On enumerating all maximal bicliques of bipartite graphs. In U. Faigle, R. Schrader, and D. Herrmann, editors, CTW, pages 105-108, 2010.

[6] C. S. J. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of London Mathemathical Society*, 36, 1961.

[7] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Closed set based discovery of small covers for association rules. In *Proc. 15èmes Journées Bases de Données Avanceés*, BDA, pages 361-381, 1999.

[8] B.Schmidt. Ein Zusammenhang zwischen Graphentheorie und Verbandstheorie. *Diplomarbeit, TH Darmstadt*, 1982.

Binary Betting Strategies with Optimal Logarithmic Growth

K. Albrecht, U. Faigle

Mathematisches Institut der Universität zu Köln
50931 Köln, Germany
{albrecht,faigle}@zpr.uni-koeln.de

Key words: binary betting, Bernoulli criterion, logarithmic growth, strategy

1 Introduction

We consider a bettor who has a capital of size $C > 0$ and is to bet on the outcome of a random source X with values in $B = \{0, 1\}$. In order to assess the quality of a strategy, we assume that the source X gives rise to a (discrete) stochastic process (X_t), where X_t is the outcome of X at time t. Letting C_t be the capital accrued at time t, the *Bernoulli criterion* is its logarithm $\log C_t$.

This criterion was introduced by Daniel Bernoulli [2] in order to resolve the *St. Petersburg paradox* that had been noted by his cousin Nikolaus Bernoulli. This criterion is commensurate with *Fechner's law* [4], which postulates that subjective intensities are generally perceived on a logarithmic scale. Interestingly, it is highly disputed by economists in the betting/investment context. The theorists (and nobel prize laureates) Merton and Samuelson [6,8], for example, argue that strategies based on this criterion are too rigid to take the economic reality of dynamics in time into account. The practitioner Thorp [7,9], on the other hand, reports very good practical success with it. [1]

In this presentation, we will investigate binary bets and identify optimal strategies with respect to the (Bernoulli) criterion of maximizing the expected value of the stochastic variable $\gamma_t = \log C_t$. This investigation generalizes the seminal work of Kelly [5] and its extensions by Belmann and Kalaba [1]. In contrast to their models, we do not make any assumption on the nature of the stochastic process (X_t). In particular, no independence or Markov chain assumptions are made.

[1] Merton's own investment fund LTCM (= Long Term Capital Management) failed in the year 2000 after 6 years of operation

2 The betting model

A *(binary) betting instance* is a triple $\mathcal{X} = (X, \alpha_0, \alpha_1)$, where X is a discrete binary source and the α_b are sequences of positive real numbers $\alpha_b^{(t)} > 0$ with the following interpretation:

- At any time $t - 1 = 0, 1, \ldots$, the bettor may place bets on the possible outcomes $(X_t = 0)$ and/or $(X_t = 1)$.
- If $(X_t = b)$ occurs, the bettor receives the payoff $\alpha_b^{(t)}$ for each unit wagered on this event. Otherwise, he looses his investment.

Our bettor does *not* have to invest all of his available capital at time t. However, for the evaluation of a strategy, we assume that the betting is *self-financing* (*i.e.*, the bettor does not replenish his financial means from outside sources).

Let C_t be the bettor's capital at time t and set $\gamma_t := \log C_t$. Then γ_t is a random variable that the bettor would like to be as large as possible by choosing an appropriate strategy. We say that a strategy is *(Bernoulli) optimal* if it maximizes the expected value

$$\hat{\gamma}_t := E(\gamma_t) = E(\log C_t) \quad \text{(assuming } C_0 = 1.\text{)}$$

2.1 Strategies

Denote by B^* the collection of all finite $(0, 1)$-strings v and write $|v| = t$ if $v \in B^t$. A *strategy* is a function $a : B \times B^* \to \mathbb{R}_+$ (denoted $(v, b) \to a(b|v)$) such that

$$a(0|v) + a(1|v) \leq 1. \tag{1}$$

We interpret $a(b|v_1 \ldots v_{t-1})$ as the fraction of the current capital C_{t-1} the investor is betting on $(X_t = b)$ having observed $(X_1, \ldots, X_{t-1}) = (v_1, \ldots, v_{t-1})$.

2.2 Expected logarithmic growth

For any $v = v_1 \ldots v_k \in B^*$, define the indicator variable

$$I(v) := \begin{cases} 1 \text{ if } (X_1, \ldots, X_k) = (v_1, \ldots, v_k) \\ 0 \text{ otherwise.} \end{cases}$$

Set $B_t := \{v \in B^* \mid |v| \leq t\}$. After t bets, the initial capital $C_0 = 1$ has grown to

$$C_t = \prod_{v \in B_{t-1}} \prod_{b \in B} (\alpha_b a(b|v) + r_v)^{I(vb)} C_0, \tag{2}$$

where $r_v := 1 - a(0|v) - a(1|v)$ is the portion of the capital not invested upon having observed v. We thus obtain the logarithmic growth as

$$\gamma_t = \log(C_t) = \sum_{v \in B_{t-1}} \sum_{b \in B} I(vb) \log(\alpha_b a(b|v) + r_v) \tag{3}$$

In view of $E(I(v_1 \dots v_k)) = p(v_1 \dots v_k) := \Pr(X_1 = v_1, \dots, X_k = v_k)$, we find

$$\hat{\gamma}_t := E(\gamma_t) = \sum_{v \in B_{t-1}} p(v) \sum_{b \in B} p(b|v) \log(\alpha_b a(b|v) + r_v), \tag{4}$$

where $p(b|v) := \Pr(X_k = b \mid X_1 \dots X_{k-1} = v)$ denotes the associated conditional probability.

2.3 Optimal expected logarithmic growth

Observing that the expected logarithmic growth is optimized if each summand in (4) is optimized individually, a detailed analysis reveals optimal strategies. We distinguish two cases.

Theorem 2.1 *Assume* $1/\alpha_0^{(t)} + 1/\alpha_1^{(t)} \leq 1$. *Then an optimal strategy* a^* *is given by the choice*

$$a^*(b|v) = p(b|v) \quad \textit{for all } v \in B^{t-1}.$$

◇

Theorem 2.2 *Assume* $1/\alpha_0^{(t)} + 1/\alpha_1^{(t)} > 1$. *Then an optimal strategy* a^* *is given as follows:*

(1) If $\alpha_0^{(t)} p(0|v) \leq 1$ *and* $\alpha_1^{(t)} p(1|v) \leq 1$, *then*
$a^*(0|v) = 0$ *and* $a^*(1|v) = 0$.

(2) If $\alpha_0^{(t)} p(0|v) > 1$ *and* $\alpha_1^{(t)} p(1|v) \leq 1$, *then*
$a^*(0|v) = \dfrac{p(0|v)\alpha_0^{(t)} - 1}{\alpha_0^{(t)} - 1}$ *and* $a^*(1|v) = 0$.

(3) If $\alpha_0^{(t)} p(0|v) \leq 1$ *and* $\alpha_1^{(t)} p(1|v) > 1$, *then*
$a^*(0|v) = 0$ *and* $a^*(1|v) = \dfrac{p(1|v)\alpha_1^{(t)} - 1}{\alpha_1^{(t)} - 1}$.

No further cases need to be considered.

◇

2.4 Remarks

The practical implementation of an optimal betting strategy obviously depends on the statistical estimates for the relevant betting parameters $p(b|v)$. The presentation will address this issue and provide furthermore a discussion of limiting cases under the assumption that the stochastic process (X_t) has a finite *evolution dimension* in the sense of Faigle and Schönhuth [3] (which properly includes all (hidden)

Markov processes). Moreover, connections with the notion of entropy in the sense of Shannon's information theory as well as physics will be outlined.

References

[1] R. Bellman and R. Kalaba: *On the role of dynamic programming in statistical communication theory.* IEEE Transactions on Information Theory 3 (1957), 197-203.

[2] D. Bernoulli: *Specimen theoriae novae de mensura sortis.* Comentarii Academiae Scientiarum Imperialis Petropolitanae, 5 (1738), 175-192.

[3] U. Faigle and A. Schoenhuth: *Asymptotic mean stationarity of sources with finite evolution dimesion.* IEEE Transactions on Information Theory 53 (2007), 2342-2348.

[4] G. Th. Fechner, *Elemente der Psychophysik.* Breitkopf und Härtel, Leipzig, 1860.

[5] J. L. Kelly: *A new interpretation of information rate.* Bell System Technical J. 35 (1956), 917-926.

[6] R. C. Merton and P.A. Samuelson: *Fallacy of the log-normal approximation to optimal portfolio decision-making over many periods.* Journal of Financial Economics 1 (1974), 67-94.

[7] L. M. Rotando and E. O. Thorp: *The Kelly criterion and the stock market.* The American Math. Monthly 99 (1992), 992-931.

[8] P. A. Samuelson: *Why we should not make mean log of wealth big though years to act are long.* Journal of Banking and Finance 3 (1979), 305-307.

[9] E .O. Thorp: *The Kelly criterion in blackjack, sports betting and the stock market.* Presentation at 10th International Conference on Gambling and Risk Taking, Montreal, 1997.

Formulations and heuristics for the k-Piecewise Affine Model Fitting problem

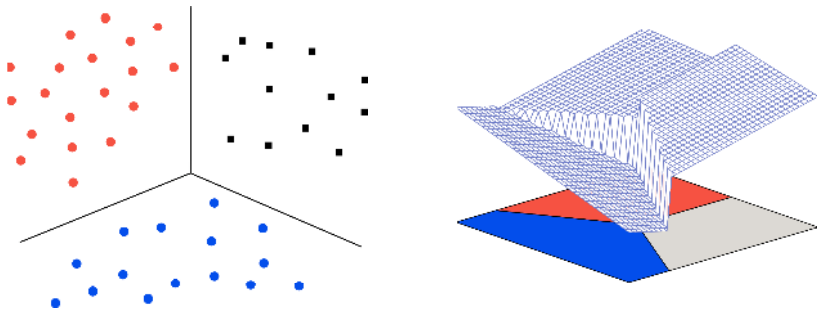
E. Amaldi, S. Coniglio, L. Taccari

Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza L. da Vinci 32, 22133, Milano
{amaldi,coniglio}@elet.polimi.it
leonardo.taccari@mail.polimi.it

Key words: piecewise affine models, data fitting, mixed-integer linear programming, symmetry breaking, metaheuristics

1 Introduction

We consider the problem of fitting a piecewise affine model to a set of observations sampled from an unknown function $f : \mathbb{R}^n \to \mathbb{R}$. Formally, given a set of m points $\{\underline{a}_1, \ldots, \underline{a}_m\}$ in $\mathbb{R}^n$, with indices in $I = \{1, \ldots, m\}$, and the corresponding observations $\{b_1, \ldots, b_m\}$ in $\mathbb{R}$, where $b_i = f(\underline{a}_i)$ for $i \in 1, \ldots, m$, the problem of k-*Piecewise Affine Model Fitting* (k-PAMF) amounts to linearly partitioning $\mathbb{R}^n$ into k domains $D_1, \ldots, D_k$, with indices in $J = \{1, \ldots, k\}$, (see the figure on the left) and to finding k corresponding affine functions $\hat{f}_j : D_j \to \mathbb{R}$, for $j = 1, \ldots, k$, of type $\hat{f}_j(\underline{p}) = \underline{y}_j^T \underline{p} + w_j$ (see the figure on the right) so as to minimize an overall fitting error.



In this work, we consider as the error the sum over all the points $\underline{a}_i$ of the absolute value of the difference between the corresponding observation b_i and the value taken in $\underline{a}_i$ by the affine function $\hat{f}_{j(i)} : D_j \to \mathbb{R}$, where $\underline{a}_i \in D_j$. More precisely, we minimize $\sum_{i=1}^{m} |b_i - \underline{y}_{j(i)}^T \underline{a}_i - w_{j(i)}|$.

Previous approaches usually tackle k-PAMF in a heuristic way, by splitting it in three subproblems which are solved in sequence: 1) a clustering problem (where

similar points are grouped into k subsets), 2) a fitting problem (where an affine submodel is found for each subset), and 3) a multi-category linear classification problem (where $\mathbb{R}^n$ is subdivided into $D_1, \ldots, D_k$ according to the clustering solution). See, for instance, [4]. Since the points belonging to a domain may be used to fit an affine function defined over another domain, such an approach is likely to produce solutions of very poor quality. As a simple example, consider the following figure where with a solution exhibiting this drawback (left) is compared to a correct fitting (right).

In this work, we propose exact methods to solve the k-PAMF problem that look for a partition of the points which, at the same time, allows for a good fitting and induces a coherent partition of the domain.

2 Mixed-Integer Linear Programming formulation

We propose the following Mixed-Integer Linear Programming formulation:

$$\min \quad \sum_{i=1}^{m} z_i \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad i \in I \tag{2}$$

$$z_i \geq b_j - \underline{a}_j^T \underline{a}_i - w_j - M(1 - x_{ij}) \qquad i \in I, j \in J \tag{3}$$

$$z_i \geq -b_j + \underline{y}_j^T \underline{a}_i + w_j - M(1 - x_{ij}) \qquad i \in I, j \in J \tag{4}$$

$$-(\underline{y}'_j - \underline{y}'_h)^T \underline{a}_i + w'_j - w'_h + 1 - M(1 - x_{ij}) \leq 0 \qquad i \in I, j, h \in J : j \neq h \tag{5}$$

$$x_{ij} \in \{0, 1\} \qquad i \in I, j \in J \tag{6}$$

$$z_i \geq 0 \qquad i \in I \tag{7}$$

$$\underline{y}_j, \underline{y}'_j \in \mathbb{R}^n \qquad j \in J \tag{8}$$

$$w_j, w'_j \in \mathbb{R} \qquad j \in J. \tag{9}$$

The variables $(\underline{y}_j, w_j)$ denote the parameters of each affine function. The binary variable x_{ij}, involved in the assignment Constraints (1), takes value 1 if the domain D_j contains the point $\underline{a}_i$, and 0 otherwise. If $x_{ij} = 1$, $\underline{a}_i$ contributes to the fitting error related to the affine function $\hat{f}_j$, which is accounted for by the variable z_i due to Constraints (2)–(3). The big-M value is chosen so as to guarantee that the constraints are not active if $x_{ij} = 0$. Constraints (4) and the variables $(\underline{y}'_j, w'_j)$ guarantee that, if $x_{ij} = 1$, then the domain D_j must contain the point $\underline{a}_i$. These constraints are derived from those of the multi-category linear classification problem, see [2] for more detail, and are adapted to the case where the partition of the points to be

classified is itself a part of the problem.

Let $X \in \mathbb{R}^{m \times k}$ be the assignment matrix with entries x_{ij} for $i \in I, j \in J$. Given any feasible solution of our formulation, another equivalent solution can be obtained by permuting the columns of X, that is by permuting the labels $1, \ldots, k$ with which the sets in the partition are indexed. Therefore there is a symmetric group acting on the columns of X. We remove the symmetry by restricting the feasible set to that of matrices X whose columns are sorted in lexicographic order. This is achieved by introducing a class of symmetry-breaking constraints called Shifted Column Inequalities (SCI) [5]. In our Branch-and-Cut algorithm, valid inequalities of this family which are violated by the current relaxation are generated, at each node of the enumeration tree, by means of the polynomial-time dynamic programming separation algorithm proposed in [5].

Since, due to the presence of the big-M, Constraints (2)-(3) do not allow for tight continuous relaxations, we propose an alternative formulation which is based on a decomposition method that uses Combinatorial Benders' Cuts [3]. The approach seems promising, since it removes some of the numerical instability that is introduced by the big-Ms and yields tighter bounds, although in its current version it is still not computationally competitive with the original formulation.

3 A combinatorial metaheuristic

To tackle large-size instances, we propose a metaheuristic which combines an adapted version of the algorithm for k-Hyperplane Clustering proposed in [AC09], called Adaptive Point-Reassignment (APR), with a multi-category linear classification method.

At each iteration, given a point-to-affine function assignment, our modified APR method identifies a set of candidate points which are likely to yield a better solution if reassigned to other affine functions, performs the reassignment, and recomputes the affine functions parameters by solving a linear program. Then, a multi-category classification problem is solved to find a linear partition of $\mathbb{R}^n$ into $D_1, \ldots, D_k$ such that each domain contains as many points as possible among those that were assigned to the corresponding affine function. The points that belong to a domain D_j but are assigned to an affine function $\hat{f}_{j'}$ with $j \neq j'$ are then reassigned to $\hat{f}_j$ and the corresponding parameters are updated. The two operations are iteratively applied until a stopping criterion is met, also adopting Tabu Search-inspired techniques to avoid loops.

4 Computational results

Computational experiments are carried out for a set of realistic, randomly generated instances. Each instance is obtained by first generating a random piecewise affine function, either continuous (`sr`) or noncontinuous (`ncr`), with an additive Gaussian noise, which is then randomly sampled. An extra set (`wave`), obtained

by randomly sampling noiseless wave functions, is also used. Our Branch-and-Cut method is implemented in and solved with CPLEX/Concert, whereas the metaheuristic is implemented in C++.

The following table reports the average CPU time and number of Branch-and-Bound nodes needed to solve k-PAMF to optimality, with or without applying the SCI inequalities. The results are averaged on instances of the same type. The symmetry breaking techniques turn out to be pivotal in reducing the solution times, allowing to achieve optimal solutions for instances with up to 150 points in $\mathbb{R}^3$ within an hour of CPU time.

The second table shows the optimality gap, on average, of the solutions found via our metaheuristic as well as the CPU time taken by the algorithm, reported as a percentage of the CPU time which is required to solve the enhanced formulation. The results show that our metaheuristic provides good solutions, on average within less than 7% of the optimal value, in a very short computing time.

References

[1] E. Amaldi and S. Coniglio. An adaptive point-reassignment metaheuristic for the k-hyperplane clustering problem. In *Proc. of Metaherustic International Conference*, pages 1–10, 2009.

[2] K.P. Bennet and O.L. Mangasarian. Multicategory discrimination via linear programming. *Optim. Method. Softw.*, 3:27–39, 1994.

[3] G. Codato and M. Fischetti. Combinatorial Benders' cuts for mixed-integer linear programming. *Oper. Res.*, 54:756–766, 2006.

[4] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine and hybrid systems. *Automatica*, 39:205–217, 2003.

[5] V. Kaibel and M.E. Pfetsch. Packing and partitioning orbitopes. *Math. Program. A*, 114:1–36, 2008.

On cycle bases with limited edge overlap

E. Amaldi, [a] C. Iuliano, [a] R. Rizzi [b]

[a] *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy*
{amaldi,iuliano}@elet.polimi.it

[b] *Dipartimento di Matematica e Informatica, Università di Udine, Udine, Italy*
romeo.rizzi@uniud.it

Key words: undirected graphs, cycle basis, edge overlap

1 Introduction

Consider connected undirected graph $G = (V, E)$ that is simple, i.e., without loops and multiple edges. Let $n = |V|$ and $m = |E|$ be respectively the number of nodes and edges. A *cycle* C is a subgraph in which every node has even degree (number of incident edges). It can be represented by an edge incidence vector $\chi(C)$ in $\{0,1\}^m$, where the component $\chi_e(C)$ corresponding to the edge $e \in E$ is 1 if $e \in C$, and 0 otherwise. The composition of two cycles C_1 and C_2 is a cycle C_3 where $\chi(C_3)$ is equal to the *sum modulo 2* of $\chi(C_1)$ and $\chi(C_2)$. All the cycles of G form a vector space over $GF(2)$, the so-called *cycle space*. A *cycle basis* $\mathcal{C}$ is a maximal set of independent cycles. Every cycle can thus be obtained by the composition of some cycles of $\mathcal{C}$. For a connected graph, the dimension of $\mathcal{C}$ is equal to $\nu := m - n + 1$.

Cycle bases have been attracting growing attention in combinatorial optimization. In particular, assigned a nonnegative weight to each edge, the problem of finding a cycle basis of minimum total weight in undirected graphs and several variants arising from applications have been extensively studied (e.g. [1]). See survey [3] and the references therein.

In this work, we investigate a natural related problem, that we refer to as the *Cycle basis with limited edge overlap* (CBEO). Given an undirected graph G with a nonnegative integer bound b_e on each edge $e \in E$, find a cycle basis $\mathcal{C}$ such that each edge $e \in E$ belongs to at most b_e cycles of $\mathcal{C}$, i.e., the overlap $\sum_{C_i \in \mathcal{C}} \chi_e(C_i)$ of e is at most b_e for each edge $e \in E$. In the *uniform case* all the edge bounds are equal. Such a cycle basis is relevant for example in the analysis of electrical networks. When one wishes to check that Kirchhoff's law is satisfied along all the loops, it is possible to focus only on those corresponding to the cycles of a basis [3]. In order to reduce the probability of damaging the links, it may be desirable to limit the number of times each link is tested.

2 Computational complexity

The problem of, given an undirected graph with a nonnegative bound b_e for each edge e, determining whether there exists a cycle basis with limited edge overlap can be solved in polynomial time when $b_e \leq 2$ for each e. Indeed, a cycle basis such that each edge belongs to at most 2 cycles exists if and only if the graph is planar (see e.g. [4]). Such a basis consists of all faces but one of the planar embedding of the graph and it can be checked in polynomial time whether the edges with $b_e = 1$ belong to that face [4].

Proposition 1 *The above problem is* NP*-complete when* $b_e \leq 3$ *for each* e.

Proof (sketch) For the case $b_e \leq 3$, we proceed by polynomial-time reduction from the following problem. Given a balanced tripartite graph $G = (V, E)$, where $V = A \cup B \cup C$ and $k = |A| = |B| = |C|$, determine if there exists a packing of node-disjoint triangles (3-edges cycles) covering all nodes. This problem can be easily shown NP-complete by polynomial-time reduction from 3*-dimensional matching problem* [2]). In the following when we use the indices i or j we mean in fact all the values of i, j such that $1 \leq i, j \leq k$. We assume that $A = \{a_i\}$, $B = \{b_i\}$, and $C = \{c_i\}$. The set of edges E can be partitioned into E_{AB}, E_{BC}, and E_{CA}, containing respectively the edges $\{a_i, b_j\}$, $\{b_i, c_j\}$ and $\{c_i, a_j\}$. For each instance of the original problem in such a tripartite graph we construct a special instance of the CBEO problem such that the answer to the former is yes if and only if the answer to the latter is yes.

We proceed in two steps. First, we construct a graph $G' = (V', E')$, where $V' = R' \cup H'$, with R' containing two copies of each node in G and H' containing 6 new nodes, namely $R' = \{a'_i, a''_i, b'_i, b''_i, c'_i, c''_i\}$, and $H' = \{x', x'', y', y'', z', z''\}$. The set of edges E' consists of 4 classes of edges: $\{a''_i, b'_j\}$, $\{b''_i, c'_j\}$, $\{c''_i, a'_j\}$ corresponding respectively to $\{a_i, b_j\}, \{b_i, c_j\}, \{c_i, a_j\} \in E$; $\{a'_i, a''_i\}$, $\{b'_i, b''_i\}$, $\{c'_i, c''_i\}$ linking the two copies of each node of G; $\{a'_i, x'\}$, $\{a''_i, x''\}$, $\{b'_i, y'\}$, $\{b''_i, y''\}$, $\{c'_i, z'\}$, $\{c''_i, z''\}$, where one endpoint is in R' and the other one is in H'; $|E_{AB}| - k$ parallel edges $\{x'', y'\}$, $|E_{BC}| - k$ parallel edges $\{y'', z'\}$, and $|E_{CA}| - k$ parallel edges $\{z'', x'\}$. Note that we introduce parallel edges only for sake of simplicity. Each one of them can be simply substituted by two edge in series in order to obtain a cycle-equivalent graph which is simple. We denote by $d'(u)$ the degree of each node $u \in V'$. The edge bounds are set equal to $d'(u) - 3$ for edges $\{u, v\}$ with $u \in R'$ and $v \in H'$, and to 1 for all the other edges. Finally, each edge $\{a''_i, b'_j\}$, $\{b''_i, c'_j\}$, and $\{c''_i, a'_j\}$ is subdivided as to become 2 edges in series with the same edge bound and each edge $\{u, v\}$ with $u \in R'$ and $v \in H'$ is subdivided as to become 3 edges in series always with the same edge bound. Let F' denote the set of the following $|E| - 2k$ edges: all edges $\{a'_i, a''_i\}$ and the parallel edges $\{x'', y'\}$, $\{y'', z'\}$, and $\{z'', x'\}$. A cycle that contains at most one edge in F' is called F'-cycle. Note that due to edge subdivision, each F'-cycle has length (number of edges) at least 12.

The graph G admits a packing of node-disjoint triangles covering all nodes if and only if there exists a packing of $|F'|$ cycles in G' consisting only of F'-cycles and such that the prescribed edge bounds are satisfied. The packing in G' consists

only of F'-cycles of length 12, one for each edge of F' (since all the edges in F' have edge bounds of 1), with all the edge bounds satisfied at equality. The $|E| - 3k$ cycles associated to parallel edges $\{x'', y'\}$, $\{y'', z'\}$, and $\{z'', x'\}$ are forced to contain $|E_{AB}| - k$ edges $\{a''_i, b'_j\}$, $|E_{BC}| - k$ edges $\{b''_i, c'_j\}$, and $|E_{CA}| - k$ edges $\{c''_i, a'_j\}$. The remaining ones belong to the cycles associated to edges $\{a'_i, a''_i\}$, that correspond to the packing of triangles in G.

Then, we extend the graph G' to $G'' = (V'', E'')$ by adding a node w linked to all other nodes, i.e., $V'' = V' \cup \{w\}$ and $E'' = E' \cup \{\{u, w\} : u \in V'\}$. The edge bound remains 1 for the edges of F', whereas it is incremented by 1 w.r.t. the value in G' for the edges of $E' \backslash F'$. For the new edges $\{u, w\}$ with $u \in V'$ the edge bound is set equal to $d'(u)$ if u is not an endpoint of an edge of F', $d'(u) - 1$ otherwise.

The graph G admits a packing of node-disjoint triangles covering all nodes if and only if in G'' there is a cycle basis with limited edge overlap. The cycle basis consists of the triangles given by w and edges in $E' \backslash F'$ and the $|F'|$ cycles of the above-mentioned packing in G'. The cycle basis is *weakly fundamental* [4], i.e., there exists an ordering of the cycles in the basis (first the triangles and then the $|F'|$ cycles) such that each cycle in the basis contains an edge that does not belong to any previous cycle. Note that, since all the edges in F' have edge bounds of 1, only F'-cycles can belong to a cycle basis. Due to the values of the edges bounds, every cycle basis must consist of $|E' \backslash F'|$ triangles containing w and $|F'|$ cycles of length 12 not containing w, such that all the edge bounds are satisfied at equality. The triangles are forced, since each edge in $E' \backslash F'$ has at least one endpoint u not incident to an edge of F' and $\{u, w\}$ has an edge bound of $d'(u)$. After selecting the triangles, the resulting graph is equivalent to G' and the remaining $|F'|$ cycles must correspond to the above-mentioned packing in G'.

Finally, it is possible to prove that each edge e in G' with bound equal to $b_e > 3$ can be substituted by $b_e - 3$ parallel edges with bound equal to 3 and the result is still valid. $\diamond$

3 A constructive heuristic *à la* de Pina

To tackle the CBEO problem we propose a constructive heuristic based on de Pina's method to compute a minimum cycle basis in an undirected graph. De Pina's method determines sequentially the ν cycles $C_1, \ldots, C_\nu$ of a minimum basis maintaining at each step $i = 1 \ldots \nu$ a basis $S_i, \ldots, S_\nu$ of the linear space orthogonal to the subspace spanned by the cycles $C_1, \ldots, C_{i-1}$ selected so far. At each step, given the current vector S_i of the orthogonal basis, C_i is the minimum weight cycle orthogonal to S_i. See [1,3] for a description of the method. Our heuristic proceeds according to de Pina's scheme with the difference that at each step $i = 1 \ldots \nu$ a special weighting of the edges is used to compute the minimum weight cycle C_i. We denote by $b_{\max}$ the maximum edge bound over all edges, i.e., $b_{\max} = \max\{b_e : e \in E\}$ and by o_{ei} the overlap of each edge $e \in E$ at the beginning of the i-th step, i.e., $o_{ei} = \sum_{C_j : j<i} \chi_e(C_j)$. At the beginning of the i-th step we have a so-called

residual edge bound of $b_e - o_{ei}$ on each edge $e \in E$. Hence, we set a weight of $(n+1)^{b_{\max}-(b_e-o_{ei})}$ for each edge $e \in E$ so as to penalize the choice of edges with a low residual edge bound. Note that using this special type of weighting is equivalent to determining a set of edges with maximum bottleneck value, where the bottleneck is given by the minimum residual edge bound over all edges. In case of ties, a set where fewer edges have the bottleneck value is preferred.

In the following table we report the results of the computational experiments for the uniform case on random graphs with n from 50 to 200 and edge density equal to 0.3, 0.5, and 0.9.

	random 0.3			random 0.5			random 0.9		
n	ν	$o_{\max}$	Time	ν	$o_{\max}$	Time	ν	$o_{\max}$	Time
50	320	4.00	0.06	555	3.83	0.21	932	3.00	0.69
100	1392	4.00	2.45	2368	4.00	11.81	3865	4.00	60.44
150	3205	4.00	35.21	5425	4.00	183.93	8802	4.00	777.61
200	5773	4.00	229.74	9754	4.00	1130.19	15719	4.00	5488.34

Since $b_{\max} = b_e$ for each $e \in E$, our approach amounts to minimizing the maximum overlap over all edges. The heuristic is implemented in C and the experiments are carried out on a UNIX Xeon 64 bit with 2.0 GHz processor and 4 GB RAM. The information in the table includes: the number of cycles of the basis ν, the maximum overlap $o_{\max}$ and the computing time in seconds. The results, averaged on 30 different random instances, show that our heuristic always finds a cycle basis with a very low edge overlap (at most 4). Since the computing time of de Pina's scheme rapidly increases with the dimension of the graphs, we are developing an adaptive version of our heuristic along the line described in [1] for finding minimum cycle bases.

References

[1] E. Amaldi, C. Iuliano, and R. Rizzi. Efficient deterministic algorithms for finding a minimum cycle basis in undirected graphs. In F. Eisenbrand and F. B. Shepherd, eds., *IPCO*, vol. 6080 of *LNCS*, 397–410. Springer, 2010.

[2] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, USA, 1979.

[3] T. Kavitha, C. Liebchen, K. Mehlhorn, D. Michail, R. Rizzi, T. Ueckerdt, and K. A. Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4):199–243, 2009.

[4] C. Liebchen and R. Rizzi. Classes of cycle bases. *Discrete Applied Mathematics*, 155(3):337–355, 2007.

Sorting Common Operations to Minimize Tardy Jobs

Claudio Arbib, [a] Giovanni Felici, [b] Mara Servilio [c]

[a] *Università degli Studi dell'Aquila, Dipartimento di Informatica,*
via Vetoio, Coppito — 67010 L'Aquila, Italy
claudio.arbib@univaq.it

[b] *Consiglio Nazionale delle Ricerche,*
Istituto di Analisi dei Sistemi e Informatica "Antonio Ruberti"
viale Manzoni, 30 — 00185 Roma, Italy
felici@iasi.cnr.it

[c] *Università degli Studi dell'Aquila, Dipartimento di Informatica,*
via Vetoio, Coppito — 67010 L'Aquila, Italy
mara.servilio@univaq.it

Key words: Linear arrangement, Scheduling, Stable set problem, Integer linear programming

---

1 Extended Abstract

We consider the following problem (P): *given a discrete finite set N with n elements, a finite collection $S = \{S_1, \ldots, S_m\}$ of (possibly duplicated) subsets of N and a positive integer d_j for each $S_j \in S$, assign every $i \in N$ to a distinct positive integer $k(i)$ so that the number of sets in S with $\max_{i \in S_j}\{k(i)\} > d_j$ is minimized.*
A motivation for (P) can be found in an operation scheduling problem, where a set J of m jobs with specific due dates $d_1, \ldots, d_m$ must be completed by a machine: job j is completed as soon as a subset S_j of operations is done. Jobs share *common* operations: this means that once operation i is completed, it is completed for all the jobs j that require i. A practical application of this problem is pattern sequencing in stock cutting [2]. The case with $|S_j| = 2$ ($j \in J$), mentioned in [1], is a graph ordering problem and was shown to be NP-hard (reduction from CLIQUE). In this paper we formulate the general case as a STABLE SET on a special graph. We investigate the structure of the graph, and discuss facet-defining and valid inequalities (the former include some Chvátal-Gomory lifted odd holes). A preliminary computational experience provides difficult instances to be tested in future research.

Problem Fomulation. The following ILP formulation of (P) uses:
binary variables x_{ik} that get value 1 if operation i is assigned to the k-th position, and 0 otherwise ($i \in N$, $k = 1, \ldots, n$);
binary variables y_j that get value 1 when job j is on time (i.e., the last of its opera-

tion is scheduled before d_j), and 0 otherwise ($j \in J$);
weights $c_{ik} > m$ associated with each pair ik ($i \in N$, $k = 1, \ldots, n$).

$$\max \sum_{j \in J} y_j + \sum_{i \in N} \sum_{k=1}^{n} c_{ik} x_{ik} \tag{1}$$

$$\sum_{k=1}^{n} x_{ik} \leq 1 \qquad i \in N \tag{2}$$

$$\sum_{i \in N} x_{ik} \leq 1 \qquad k = 1, \ldots, n \tag{3}$$

$$y_j + \sum_{k > d_j} x_{ik} \leq 1 \qquad j \in J, i \in S_j \tag{4}$$

$$y_j + \sum_{i \in S_j} x_{ik} \leq 1 \qquad j \in J, k > d_j \tag{5}$$

$$x_{ik} \geq 0, y_j \geq 0 \text{ and integer} \tag{6}$$

Note that x_{ik} does not need to be constrained to integrality; also, due to the choice of c_{ik}, any optimal solution fulfils (2) and (3) with equality.

Call $G_I = (V_I, E_I)$ the intersection graph of the constraint matrix of (1)-(6):
V_I contains a node ik for each variable x_{ik} (x-node) and a node j for each variable y_j (y-node);
E_I contains an edge between any two nodes associated with variables that appear in the same constraint.
Similarly to [4,6], formulation (1)-(6) describes a STABLE SET on G_I.

Observe that $(ik, hl) \in E_I$ if and only if either $i = h$ (horizontal edges) or $k = l$ (vertical edges); secondly, $(ik, j) \in E_I$ if and only if $i \in S_j$ and $k > d_j$: finally, $(j, h) \notin E_I$ for all $j, h \in J$.

Let Q be the convex hull of all the $(\mathbf{x}, \mathbf{y}) \in \{0,1\}^{n^2 \times m}$ that are feasible solutions of (1)-(6). From the above observations,

$i)$ any set of horizontally (vertically) adjacent x-nodes is a clique;
$ii)$ the horizontal (vertical) neighborhood of a y-node, plus the y-node itself, is a clique.

As (i) describes all the connections of the subgraph induced by the x-nodes, any x-subgraph of G_I is perfect. Moreover, any set of y-nodes is stable, so we conclude that constraints (2)-(5) are all the maximal-clique inequalities of G_I, and hence define facets of Q.

Odd holes and wheels. Besides maximal cliques, several classes of inequalities are known to be valid for the STABLE SET polytope: we considered odd holes, odd anti-holes and wheels. Odd anti-holes are promptly ruled out by the following theorem, whose proof is omitted for brevity:

Theorem 1.1 *G_I does not contain any odd anti-hole as an induced subgraph.*

Let us focus on odd-hole inequalities. From the perfectness of any x-subgraph of G_I we know that every odd hole of G_I must contain at least one y-node. We start our analysis from *odd holes with a single y-node*.

Let W denote the vertex set of an odd-hole formed by

(1) A y-node $j \in J$, corresponding to variable y_j.
(2) Four x-nodes associated with operations $i_1, i_2 \in S_j$, and with positions $d_1, d_2 > d_j$ and $k_1, k_2 \leq d_j$, and corresponding to variables $x_{i_1k_1}, x_{i_1d_1}, x_{i_2k_2}$, $x_{i_2d_2}$.
(3) $(|W| - 5)$ x-nodes associated associated with operations $i_3, \ldots, i_{\frac{|W|-1}{2}} \in N$ and with positions $k_3, \ldots, k_{\frac{|W|-3}{2}} \leq d_j$, and corresponding to variables $\{x_{i_3,k_1}, x_{i_3,k_3}, x_{i_4,k_3}, x_{i_4,k_4}, \ldots, x_{i_{\frac{|W|-1}{2}},k_{\frac{|W|-3}{2}}}$.

Then inequality $y_j + \sum_{ik \in W} x_{ik} \leq \left\lfloor \frac{|W|}{2} \right\rfloor$, with $y_j \in W$, is valid for (1)-(6).

Moreover, it is easy to see that the above inequality can be enforced to a *Chvátal-Gomory lifted odd hole* (CGH inequalities, see [3] and [4] for details). This is done via a Chvátal-Gomory derivation that combines with coefficient $\frac{1}{2}$ the constraints (2), (3), and just one out of (4), (5), where the hole variables occur. Additional considerations, here omitted for brevity, lead to conclude that:

Theorem 1.2 *The inequality $y_j + \sum_{x_{ik} \in X^W} x_{ik} + \sum_{x_{ik} \in W} x_{ik} \leq \left\lfloor \frac{|W|}{2} \right\rfloor$ corresponding to an odd hole induced by $W \subseteq V_I$ and containing just one y-node $y_j \in W$ is a CGH inequality and is facet defining for Q.*

Let us now consider the wheel inequalities obtained by lifting an x-variable that plays the role of *hub* node in an odd hole inequality. It is easy to see that the only induced wheels of G_I derive from 5-hole lifting. Moreover, the form of G_I is such that no 5-hole with a single y-node can be extended to a wheel. Thus all the wheels of G_I are obtained by 5-holes containing two y-nodes. We can now detail the general form of a wheel-inequality of G_I. Let $W \subseteq V_I$ induce a 5-hole of G_I such that $j_1, j_2 \in W$, for some $j_1, j_2 \in J$; let also $x_{i_rk_c} \in V_I \setminus W$ be the hub of a wheel obtained by lifting the hole inequality associated with W. Then the following holds true:

Theorem 1.3 *The wheel inequality $y_{j_1} + y_{j_2} + 2x_{i_rk_c} + \sum_{ik \in W} x_{ik} \leq 2$ is facet defining for Q.*

Cover Inequalities. We next consider a class of optimality cuts, called *cover*, and their separation. Call $(\bar{P})$ a version of (1)-(6) — equivalent in terms of optimality — where constraints (2), (3) hold with equality. Let $\bar{J} \subseteq J$ and $L = \bigcup_{j \in \bar{J}} S_j \subseteq N$. Then

Theorem 1.4 *If $|L| > \max_{j \in \bar{J}}\{d_j\}$, then inequality $\sum_{j \in \bar{J}} x_j \leq |\bar{J}| - 1$ is valid for $(\bar{P})$.*

Proof. The completion time of the operations in L (and hence of the jobs in $\bar{J}$) cannot be squeezed below $|L|$; thus, if $|L| > \max_{j \in \bar{J}}\{d_j\}$ at least one job in $\bar{J}$ is late. $\diamond$

In order to separate exactly this class of inequalities we need to solve a potentially difficult problem. Given a fractional solution $\bar{x}$ of the linear relaxation of $(\bar{P})$, let $\bar{y} = (\bar{y}_j)_{j=1}^m \in \{0,1\}^m$ be the incidence vector of $\bar{J}$ and $z = (z_i)_{i=1}^n \in \{0,1\}^m$ be the incidence vector of L. Finding a maximally violated inequality in the above class amounts then to solving the following separation problem:

$$\min\{\sum_{j\in J}(1-\bar{x}_j)\bar{y}_j : \sum_{i\in N} z_i > d_j\bar{y}_j, j\in J; \sum_{S_j\ni i}\bar{y}_j \geq z_i, i\in N\}$$

Note that in the above problem at least one z_i must be > 0 in order to fulfil the first set of constraints, and that the second constraint set requires $\bar{y}_j \geq 0$ in at least one case. For $d_j = n-1$ the problem is a general SET COVERING problem. However, in the simpler case studied by [1], separation is a special EDGE COVERING [5] and can be done in polynomial time by matching.

Preliminary computational experience. We tested formulation (1)-(6) on instances elaborated from solutions of the Cutting Stock Problem given in the form of sets of cutting patterns [2]. We observed small gaps (in the range of 3%) when the density of patterns (number of part types per pattern) is relatively low; but the gap increases (up to 97% in some cases) as the pattern density increases from 10% to 50%. Additionally, we observed that wheel inequalities do not provide any contribution at the root node, while slight improvements are obtained by cover inequalities. Work is in progress to evaluate the contribution of CGH inequalities, and to set up a more elaborated test of the described inequalities within a branch-and-cut code.

References

[1] C. Arbib, M. Flammini, F. Marinelli, Minimum flow time graph ordering, in H.L. Bodlaender (Ed.): *WG 2003*, Lecture Notes in Comp. Sci. **2880** (2003) 23-33

[2] G. Belov, G. Scheithauer, Setup and open-stacks minimization in one-dimensional stock cutting. *INFORMS Journal on Computing* **19**(1) 27-35

[3] A. Caprara, M. Fischetti, $0\text{-}\frac{1}{2}$ Chvátal-Gomory cuts, *Mathematical Programming* **74** (1996) 221-235

[4] A. Caprara, J.J. Salazar, Separating lifted odd-hole inequalities to solve the index selection problem. *Discrete Applied Mathematics*, **92** (1999) 11-134

[5] J. Plesník, Constrained weighted matchings and edge coverings in graphs, *Discrete Applied Math*. **92** (1999) 229-241

[6] F. Rossi, S. Smriglio, A set packing model for the ground holding problem in congested networks, *European J. of Operational Research* **131** (2001) 400-416

Generalized row family inequalities for the set covering polyhedron

G. Argiroffo, [a] Wagler A.K. [b]

[a] *Dept. de Matemática, Universidad Nacional de Rosario, Rosario, Argentina,*
garua@fceia.unr.edu.ar

[b] *Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes, Université Blaise Pascal and CNRS, Clermont-Ferrand, France,*
wagler@isima.fr

Key words: set covering polyhedron, valid inequality, row family inequality

1 Introduction

Let M be a clutter matrix, i.e., a $0,1$-matrix without dominating rows and zero columns. The set covering polyhedron (SCP), denoted by $Q^*(M)$, is the convex hull of integer points in $Q(M) = \{x \geq \mathbf{0} : Mx \geq \mathbf{1}\}$. The set packing polyhedron (SPP), denoted by $P^*(M)$, is the convex hull of integer points in $P(M) = \{x \geq \mathbf{0} : Mx \leq \mathbf{1}\}$. Note that $P^*(M)$ equals the stable set polytope $STAB(G)$ of a graph G, if M is the matrix having the incidence vectors of maximal cliques of G as rows.

From the polyhedral point of view, the set covering polyhedron SCP and the set packing polyhedron SPP have strong similarities. In particular, many key concepts have been transferred from SPP to SCP, see for example the works of Balas and Ng [2], Cornuéjols and Sassano [3], Nobili and Sassano [4] and Sassano [7].

A canonical question is to which extend this also applies to classes of valid inequalities. Recently, a new class of inequalities for the SCP, called row family inequalities, has been introduced by Argiroffo and Bianchi [1] as the counterpart of so-called clique family inequalities for SPP, see Oriolo [5] for their definition.

Consider a clutter matrix M and a row submatrix F of M. Let p be an integer with $1 \leq p \leq |F|$, where $|F|$ denotes the number of rows of F, and define the following two column sets

$$C_p = \{i \in \{1, \ldots, |M|\} : \ 1 \leq |F_j \in F : \ F_{ji} = 1| \leq p\},$$
$$C_{p+1} = \{i \in \{1, \ldots, |M|\} \ : |F_j \in F : \ F_{ji} = 1| = p + 1\}.$$

The *row family inequality* (F, p) is defined as

$$(p+r)\sum_{i\in C_p} x_i + (p+r+1)\sum_{i\in C_{p+1}} x_i \geq (p+r)\left\lceil \frac{|F|}{p} \right\rceil \tag{1}$$

with $r = |F| - p\lceil \frac{|F|}{p} \rceil$.
While clique family inequalities are valid for the stable set polytope of *every* graph [5], the above row family inequalities are not valid for $Q^*(M)$ in general, but only under some conditions [1]. According to Argiroffo and Bianchi [1] a row family inequality (F, p) is valid for $Q^*(M)$ if

$$p|B \cap C_p| + (p+1)|B \cap C_{p+1}| \geq |F|$$

holds for every cover B of M, i.e., for every $0,1$-solution of $Mx \geq \mathbf{1}$. This is particularly satisfied for $F = M$ and $p = \beta(M)$, where $\beta(F) = \max\{\mathbf{1}^T F^i : F^i \text{ column of } F\}$.
In other words, a row family inequality (F, p) is valid for $Q^*(M)$, provided that p is sufficiently large. Our aim is to extend the concept of row family inequalities in such a way that we obtain a new class of inequalities being valid for the SCP in general, and involving more than two non-zero coefficients. For that, we adapt the concept of general clique family inequalities, introduced by Pêcher and Wagler [6] for the SPP, to the set covering case. Finally, we discuss the potential of general row family inequalities for describing the SCP.

2 General row family inequalities

Consider a clutter matrix M and a row submatrix F of M. Let $\beta(F) = \max\{\mathbf{1}^T F^i : F^i \text{ column of } F\}$ and integers p with $1 \leq p \leq \beta(F)$, r with $0 \geq r \geq |F| - p\lceil \frac{|F|}{p} \rceil$ and J with $0 \leq J \leq \beta(F) - p$. For $j = 1, \ldots, J$, we consider the following column sets

$$C_p = \{i \in \{1, \ldots, n\} : \ 1 \leq |F_k \in F : \ F_{ki} = 1| \leq p\},$$
$$C_{p+j} = \{i \in \{1, \ldots, n\} \ : |F_k \in F : \ F_{ki} = 1| = p + j\}.$$

We define the *general row family inequality* (F, p, r, J, b) by

$$\sum_{j\in\{0,\ldots,J\}} (p + r + j)\, x(C_{p+j}) \geq b. \tag{2}$$

Note that a row family inequality (F, p) is clearly a general row family inequality (F, p, r, J, b) with $r = |F| - p\lceil \frac{|F|}{p} \rceil$, $J = 1$, and $b = (p + r)\lceil \frac{|F|}{p} \rceil$, but valid only if p is sufficiently large [1].
Our main result is that choosing the minimal value for r, the maximal value for J, and an appropriate right hand side b yields in general valid inequalities:

Theorem 2.1 *Let F be a row submatrix of a clutter matrix M and p be an integer with $1 \leq p \leq \beta(F)$. The general row family inequality (F, p, r, J, b) is valid for $Q^*(M)$ if $r = |F| - p\lceil \frac{|F|}{p} \rceil$, $J = \beta(F) - p$, and $b = (p + r)\lceil \frac{|F|}{p} \rceil$.*

Thus, with the above choice of parameters, the general row family inequality (F, p, r, J, b) is valid for $Q^*(M)$. Such inequalities do not always define facets, but have the potential to cut off fractional extreme points from $Q(M)$, as the following example shows. Consider the clutter matrix

$$M = \begin{bmatrix} 1\,1\,1\,1\,0\,0\,0 \\ 1\,1\,1\,0\,0\,0\,1 \\ 1\,1\,0\,1\,0\,1\,0 \\ 1\,0\,1\,1\,1\,0\,0 \\ 1\,1\,0\,0\,1\,1\,0 \\ 1\,1\,0\,0\,1\,0\,1 \end{bmatrix}$$

and let F be its submatrix defined by the first 4 rows. Taking $p = 2$ and r, J, b as defined in Theorem 2.1 yields the valid general row family inequality $4x_1 + 3(x_2 + x_3 + x_4) \geq 4$. It is not a facet of $Q^*(M)$, but it cuts off the fractional extreme point $(0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ of $Q(M)$.

Moreover, in [1] it is reported that several (0,1)-, (1,2)-, and (0,1,2)-valued facet-defining inequalities of the covering polyhedra of special matrices are row family inequalities. Hence, a canonical question is when the general row family inequalities (F, p, r, J, b) suffice to describe all non-trivial facets of $Q^*(M)$.

This motivates the following lines of future research:

- Consider general row family inequalities (F, p, r, J, b) with other parameter choices (e.g. based on structural properties of the selected rows in F).
- Investigate how general row family inequalities (F, p, r, J, b) with different parameter choices can be generated with the help of the Chvátal-Gomory procedure or other methods to strengthen valid inequalities.
- Evaluate the corresponding ranks of the resulting inequalities and linear relaxations to determine their strength.
- Interprete known valid or facet-defining inequalities (with more than two different non-zero coefficients) in this context.
- Investigate for which matrices M and which parameter choices the general row family inequalities (F, p, r, J, b) suffice to describe all non-trivial facets of $Q^*(M)$.

References

[1] G. Argiroffo and S. Bianchi, *Row family inequalities for the set covering polyhedron*, Electronic Notes in Discrete Mathematics **36** (2010), 1169–1176.

[2] E. Balas and S. M. Ng, *On the set covering polytope: I. All the facets with coefficients in {0,1,2}*, Mathematical Programming **43**, 57–69, 1989.

[3] G. Cornuéjols and A. Sassano, *On the 0, 1 facets of the set covering polytope*, Mathematical Programming **43** (1989), 45–55.

[4] P. Nobile and A. Sassano, *Facets and lifting procedures for the set covering polytope*, Mathematical Programming **45** (1989), 111–137.

[5] G. Oriolo, *Clique family inequalities for the stable set polytope of quasi-line graphs*, Discrete Applied Mathematics **132** (2004), 185–201.

[6] A. Pêcher and A. Wagler, *Generalized clique family inequalities for claw-free graphs*, Electronic Notes in Discrete Mathematics **25** (2006), 117–121.

[7] A. Sassano, *On the facial structure of the set covering polytope*, Mathematical Programming **44** (1989), 181–202.

Identifying a Graph's Connected Components by Solving Only One Network Flow Problem

Joanna Bauer

Department of Informatics, University of Bergen, PB. 7800, 5020 Bergen, Norway
jba081@uib.no

Abstract

This article shows how by solving only one minimum-cost network flow problem, all (strongly) connected components of a (directed) graph can be identified. Thus, the separation problem for generating constraints for linear programming formulations seeking optimal trees within a graph can be solved conveniently and efficiently.

Key words: Linear Programming, Connected Components, Separation Problem, Constraint Generation

1 Introduction

Identifying a graph's connected components is a well know problem. For a graph with node set V and edge set E, it can readily be solved by depth-first-search (DFS) in time $O(|V|+|E|)$ [1]. Nevertheless, it is for several reasons interesting to identify them by linear programming (LP):

It is generally interesting to study LP formulations for graph problems, even if there exist polynomial time exact algorithms. For example, a network flow (NF) formulation for the minimum spanning tree in planar graphs is presented in [4] "*for reasons of convenience, flexibility, and adaptability to related and difficult problems*."

Convenience is the most prominent factor in the context of the separation problem: Often, the solution to a problem must be a connected graph, and the LP formulation has exponentially many constraints, for example, when using subtour elimination or cutset LP formulations for many NP-hard network design problems, such as telecommunication routing problems or VLSI design [3]. Then, identifying a graph's connected components is useful for solving problem instances by constraint generation. When using an LP solver in connection with a modelling language like AMPL [2], it is most convenient to generate constraints by solving an LP instance instead of DFS. In this context, two NF formulations to generate constraints for a subtour elimination formulation were proposed in [3].

It is especially useful to identify the connected components by solving an NF instance, as they are significantly faster to solve than other LP instances. The methods in [3] solve $|V| - 1$ NF instances when applied to a graph with $|V|$ nodes. They yield at least one violated cut, or show that none exists.

This article shows how by solving only one minimum-cost NF instance, all (strongly) connected components of a (directed) graph can be identified. Thus, one violated constraint for every connected component can be generated.

2 Preliminaries

For an undirected graph $G = (V = \{v_1, \ldots, v_n\}, E)$, a node subset $C \subseteq V$ is a *connected component* if every pair of nodes in C is connected by a path in G, and there is no node in $V \setminus C$ for which G contains a path to every node in C (i.e. C is maximal).

For a digraph $\underline{G} = (V, A)$, a node subset $C \subseteq V$ is a *strongly connected component* if $\underline{G}$ contains a directed path from v to w and a directed path from w to v for every pair $v, w \in C$, and there is no node in $u \in V \setminus C$ for which $\underline{G}$ contains paths to every node in C and paths from every node in C to u.

Let κ be the number of (strongly) connected components of a graph, and let $\gamma_k = \min\{i : v_i \in C_k\}$ be the least identifier of the nodes in $C_k, k = 1, \ldots, \kappa$.

For an edge set E, let $A_E = \bigcup_{\{v,w\}\in E}\{(v, w)\} \cup \{(w, v)\}$ be the set of the corresponding directed arcs. For a node s, let $A_s = \bigcup_{v\in V}\{(s, v)\}$.

Let $\underline{G_s}(V, A) = (V \cup \{s\}, A \cup A_s)$ be the extension of $\underline{G}$ by one node s and arcs from s to all nodes in V (see Fig. 1 – 2).

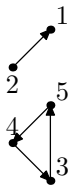


Fig. 1. $\underline{G}$

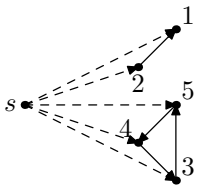


Fig. 2. $\underline{G_s}$

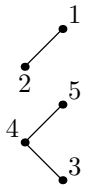


Fig. 3. $G = (V, E)$

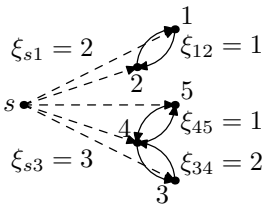


Fig. 4. $(V \cup \{s\}, A_E \cup A_s)$ with positive ξ_{vw}

3 The Minimum-Cost Network Flow Problem for Identifying the Connected Components of a Graph

For $\underline{G} = (V, A)$, let the min-cost NF problem $\mathcal{P}(V, A)$ be defined over $\underline{G_s}$ as

$$\text{minimize } c^T x \tag{1}$$
$$\text{subject to } \mathcal{A}x = b\,, \tag{2}$$
$$x \geq 0\,, \text{where}$$

x_{vw} is the decision variable on the amount of flow on arc $(v, w) \in A \cup A_s$,

$c_{vw} = \begin{cases} i_w & \text{if } (v, w) \in A_s \\ 0 & \text{if } (v, w) \in A \end{cases}$ is the unit transportation cost on $(v, w) \in A \cup A_s$,

$\mathcal{A} \in \{-1, 0, 1\}^{|V \cup \{s\}| \times |A \cup A_s|}$ is the node-arc incidence matrix of $\underline{G_s}$, that is

$$a_{u,vw} = \begin{cases} 1 & \text{if } u = v \\ -1 & \text{if } u = w \\ 0 & \text{otherwise,} \end{cases}$$

and $b_v = \begin{cases} |V| & \text{if } v = s \\ -1 & \text{otherwise} \end{cases}$ is the demand at $v \in V \cup \{s\}$.

$$\text{The dual problem } \mathcal{D}(V, A) \text{ of } \mathcal{P}(V, A) \text{ is to} \qquad \begin{array}{l} \text{maximize } b^T y \\ \text{subject to } \mathcal{A}^T y \leq c\,. \end{array} \qquad (3)$$

The connected components of an undirected graph $G = (V, E)$ are identified by solving $\mathcal{D}(V, A_E)$, where $\mathcal{P}(V, A_E)$ is defined over the digraph $\underline{G_s}(V, A_E) = (V \cup \{s\}, A_E \cup A_s)$ (see Fig. 3 – 4).

Proposition 1 *For $G = (V, E)$ with connected components C_k, $k = 1, \ldots, \kappa$, the optimal solution ξ of $\mathcal{P}(V, A_E)$ is $\xi_{sv} = \begin{cases} |C_k| & \text{if } v = \gamma_k\,, \\ 0 & \text{otherwise}\,, \end{cases}$ and there is an optimal solution η of $\mathcal{D}(V, A_E)$ of the form $\eta_v = \begin{cases} 0 & \text{if } v = s\,, \\ \gamma_k & \text{if } v \in C_k\,. \end{cases}$*

proof 1 *In (1), only the arcs leaving s have positive cost, while all other arcs have zero cost. Since there are no upper bounds on the flow variables x in $\mathcal{P}(V, A_E)$, all optimal solutions send all flow destined to the nodes in C_k along the cheapest arc entering C_k, which is (s, γ_k). From there, the flow is distributed within C_k according to the flow balance constraints (2).*

By complementary slackness, η fulfils to equality those constraints in (3) which correspond to those x_{vw} for which $\xi_{vw} > 0$.

For $\xi_{s\gamma_k}$, this constraint is $y_{\gamma_k} - y_s = \gamma_k$. These constraints form a system of κ equations on $\kappa + 1$ variables, where κ is the number of connected components of G. Setting $\eta_s = 0$ results in $\eta_{\gamma_k} = \gamma_k$.

For $\xi_{vw} > 0$ with $v \neq s$, this constraint is $\eta_v - \eta_w = 0$. Since for every $v \in C_k$, there is a path $\gamma_k = w_0, w_1, \ldots, w_i = v$ in C_k from γ_k to v with positive flows $x_{j,j+1}$, it follows that $\eta_v = \eta_{\gamma_k} = \gamma_k \forall v \in C_k$.

Corollary 1 *Let η be the optimal solution of $\mathcal{D}(V, A_E)$. Then, every maximal subset $C \subseteq V$ where $v, w \in C \Leftrightarrow \eta_v = \eta_w$ is a connected component of G.*

3.1 Identifying the strongly connected components of a digraph

Let η and $\bar{\eta}$ be the optimal solutions of $\mathcal{D}(V, A)$ and $\mathcal{D}(V, \bigcup_{(v,w)\in A}\{(w, v)\})$, respectively. Then, every maximal subset $C \subseteq V$ where $v, w \in C \Leftrightarrow \eta_v =$

η_w and $\bar{\eta}_v = \bar{\eta}_w$ is a strongly connected component of $\underline{G}$.

3.2 Identifying the directed subtrees of a directed forest

For constraint generation, the following case is especially interesting: Many applications require the solution to be a directed tree rooted at one node in V. In a directed rooted tree, every node except the root node has exactly one incoming arc. Therefore, LP formulations for these problems often contain a constraint restricting the indegree δ_v^- of every node v to at most 1. Thus, if for $\underline{F} = (V, A)$ we have $\delta_v^- \leq 1 \forall v \in V$, then the directed subtrees of $\underline{F}$ correspond to the connected components of the undirected graph $(V, \bigcup_{(v,w)\in A}\{\{v, w\}\})$, and can thus be identified by solving $\mathcal{D}(V, \bigcup_{(v,w)\in A}\{\{v, w\}\})$.

3.3 Most violated subtour elimination constraint as presented in [3]

For the subtour elimination constraints $\zeta(S) = \sum_{\substack{(v,w)\in E \\ v,w\in S}} x_{vw} - (|S| - 1) \leq 0 \, \forall S \subset V$, the second method in [3] finds a *most violated constraint*, that is, a subset $S \subset V$ maximizing $\zeta(S)$. Since one of these subsets equals a connected component, a most violated subtour elimination constraint can be identified by solving $\mathcal{D}(V, A_E)$.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[2] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A modeling language for mathematical programming*. Duxbury Press, 2002.

[3] T.L. Magnanti and L.A. Wolsey. Optimal trees. *Handbooks in operations research and management science*, 7:503–615, 1995.

[4] J.C. Williams. A linear-size zero-one programming model for the minimum spanning tree problem in planar graphs. *Networks*, 39(1):53–60, 2002.

Computing the differential of a graph

S. Bermudo, [a] [1] H. Fernau [b]

[a] *Department of Economy, Quantitative Methods and Economic History, Pablo de Olavide University 41013-Sevilla, Spain*
sbernav@upo.es

[b] *Fachbereich 4—Abteilung Informatik, Universität Trier, D-54286 Trier, Germany*
fernau@uni-trier.de

Key words: Differential (of a graph), computational complexity

1 Introduction

Let $\Gamma = (V, E)$ be a graph of order n and let $B(S)$ be the set of vertices in $V \setminus S$ that have a neighbor in the vertex set S. The *differential of* S is defined as $\partial(S) = |B(S)| - |S|$ and the *differential of a graph* Γ, written $\partial(\Gamma)$, is equal to $\max\{\partial(S) : S \subseteq V\}$. A set S satisfying $\partial(S) = \partial(\Gamma)$ is also called a *∂-set*. If S has minimum cardinality among all ∂-sets, S is called a *minimum ∂-set*. The size of a minimum ∂-set S can model the cheapest way of influencing the whole graph, assuming that the vertices in S serve as multiplicators by influencing their neighbors. The graph parameter ∂ was introduced in [3]. There, also several basic properties were derived.

Notice that for a graph Γ of order n, $0 \leq \partial(\Gamma) \leq n - 2$.

Proposition 1.1 *[3] For paths P_n, $n \geq 1$ and cycles C_n, $n \geq 3$, $\partial(C_n) = \partial(P_n) = \left\lfloor \frac{n}{3} \right\rfloor$.*

Hence (and not surprisingly), it is easy to compute the differential of a graph with a maximum degree of two. However, as we will show in this paper, this picture changes if we ask the same question for (sub)cubic graphs or for split graphs. This is interesting, as several graph parameters are computable in polynomial time on (sub)cubic or split graphs, although they are NP-hard on general graphs, see [4,5].

2 Complexity of the differential of a graph

Given a graph $\Gamma = (V, E)$ and an integer k, we consider the following decision problem: Is $\partial(\Gamma) \geq k$? We refer to this problem as *Maximum Differential Set*

[1] Partially supported by Ministerio de Ciencia y Tecnología, (MTM 2009-09501) and by the Junta de Andalucía (FQM-260) and (P06-FQM-02225).

(MDS). A derived problem, called *Minimum Maximum Differential Set* (MMDS), is to determine if there is a set S with $\partial(S) = \partial(\Gamma)$, $|S| \leq l$ and $\partial(S) \geq k$, where k, l are two given parameters.

We will prove that the problems (MDS) and (MMDS) are NP-complete in two well known graph classes, split graphs and cubic graphs.

2.1 Complexity of the differential of split graphs.

In our reduction, we consider the *3-Dimensional Perfect Matching with Maximum Degree Three* (3DPM3) problem that is known to be NP-hard: given three sets X, Y, Z that comprise the ground set G, with $|X| = |Y| = |Z| = q$, and a relation $R \subseteq X \times Y \times Z$, such that, for each $v \in G = X \cup Y \cup Z$, there are at most three triples in R that contain v. Are there q triples $r_1, \ldots, r_q$ in R such that each element of G occurs exactly once in one of the triples?

As mentioned in the classical reference of Garey and Johnson [2], 3DPM3 is NP-complete. We can transform a 3DPM3 instance into an instance of MDS as follows: Consider $\Gamma = (V, E)$ with $V = G \cup R$ and $E = E_1 \cup E_2$, where $E_1 = \{v(x, y, z) : v \in G, (x, y, z) \in R, v \in \{x, y, z\}\}$ and $E_2 = \{(x, y, z)(x', y', z') : (x, y, z) \in R, (x', y', z') \in R \backslash \{(x, y, z)\}\}$. Let us note that, given G and R, this transformation can be computed in polynomial time.

Lemma 2.1 *If $\partial(S) = \partial(\Gamma)$, then $G \cap S = \emptyset$.*

Let us observe that, if $\partial(S) = \partial(\Gamma)$ and x is not a private neighbor of $r = (x, y, z) \in S \cap R$, we can take $S' = S \setminus \{r\}$ to obtain

$$\partial(S') = |B(S')| - |S'| \geq |(B(S) \cup \{r\}) \setminus \{y, z\}| - |S| + 1 = |B(S)| - |S| = \partial(S).$$

Therefore, we can assume that any $r \in R \cap S$ has three private neighbors in G. It is also clear that r cannot have more than three private neighbors in G. Moreover, Any $S \subseteq R$ that, as a vertex set of Γ, obeys that any $r \in S$ has three private neighbors in G corresponds to a three-dimensional matching in the instance (X, Y, Z, R) of 3DPM3.

Lemma 2.2 *If $\partial(S) = \partial(\Gamma)$, then $|S| \leq q$ and $\partial(S) \leq q + |R|$.*

Proof (sketch) On the one hand, it is clear that $|S| \leq q$. On the other hand,

$$|B(S)| = |(R \setminus S) \cup (B(S) \cap G)| = |R| - |S| + 3|S| = 2|S| + |R|,$$

so $\partial(S) = |B(S)| - |S| = |S| + |R| \leq q + |R|$.
$\diamond$

If $\partial(\Gamma) \geq q + |R|$, then the original 3DPM3 instance (X, Y, Z, R) has a solution. This solution is given by the set S with $\partial(S) = \partial(\Gamma)$ of minimum cardinality.

If the original 3DPM3 instance (X, Y, Z, R) has a solution, then $\partial(\Gamma) \geq q + |R|$. To see this, it is enough to consider the solution $S \subseteq R$, which satisfies $|S| = q$ and, interpreting this as a vertex set of Γ, we obtain that

$$\partial(S) = |B(S)| - |S| = ((|R| - |S|) + 3|S|) - |S| = q + |R|.$$

Hence, $\partial(\Gamma) \geq q + |R|$.

Taking $k = q + |R|$ and by the results above we have the following results.

Theorem 2.3 *MDS in split graphs is NP-complete.*

The previous reasoning also yields the following result, where we have to choose $l = q$ as our parameter in the otherwise identical transformation.

Theorem 2.4 *MMDS in split graphs is NP-complete.*

2.2 Complexity of the differential of cubic graphs.

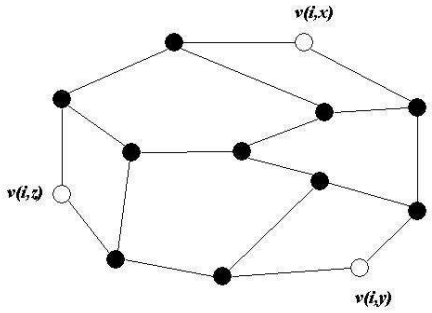
The following might be of independent interest:

Lemma 2.5 *If S is a minimum ∂-set in a subcubic graph, then S is an independent set.*

Now, in our reduction, we consider the *3-Dimensional Perfect Matching with Degree Three* (3DPM3'): given three sets X, Y, Z that comprise the ground set G, with $|X| = |Y| = |Z| = q$, and a relation $R \subseteq X \times Y \times Z$, such that, for each $v \in G = X \cup Y \cup Z$, there are three triples in R that contain v. Are there q triples $r_1, \ldots, r_q$ in R such that each element of G occurs exactly once in one of the triples?

Theorem 2.6 *3DPM3' is NP-complete.*

To see the complexity of MDS in a cubic graph, let us transform a 3DPM3' instance into an instance of MDS as follows: For every $r(i) = (x, y, z) \in R$ we consider the following graph $\Gamma_i = (V_i, E_i)$



and we join the vertex x with $v(i, x)$, y with $v(i, y)$ and z with $v(i, z)$. With this construction we obtain a graph $\Gamma = (V, E)$ such that

$$|V| = 3q + 13|R| = 3q + 13(3q) = 42q.$$

Surpressing details, we can show with the help of this gadget:

Corollary 2.7 *MDS and MMDS in cubic graphs are NP-complete.*

Further refined work allows to sharpen this assertion towards MAXSNP completeness for the maximization problem corresponding to MDS on subcubic graphs.

3 Algorithmic Aspects

Having established NP-hardness of some graph parameters, it is natural to ask questions about approximation, parameterized and exact algorithms. In another paper, we obtained the following result:

Theorem 3.1 (B&F 2011) *In a connected graph Γ of order n, $n \geq 3$, $\partial(\Gamma) \geq \left\lceil \frac{n}{5} \right\rceil$.*

Since isolated vertices can be deleted by preprocessing, we obtain:

Corollary 3.2 *MDS admits a problem kernel with at most $5k$ vertices and is hence fixed-parameter tractable on general graphs.*

This result trivially transfers to the two-parameter problem MMDS.

Since any ∂-set S of Γ also gives a packing of Γ with paths of length two, known algorithmic approximability results for P_3-packings also yield approximability results for computing $\partial(\Gamma)$, losing an additional factor of $(\Delta - 1)$ on graphs of maximum degree Δ compared to the P_3-packing approximation factor.

Finally, we obtained the following result, using a Measure & Conquer analysis of a simple branching algorithm very similar to one discussed in [1] for solving MINIMUM DOMINATING SET:

Theorem 3.3 *MDS can be solved in time $O(1.755^n)$ on arbitrary graphs of order n.*

In view of the motivation from influencing networks, it would be also interesting to study edge-weighted problem variants.

References

[1] Fernau, H. and D. Raible, *Searching trees: an essay*, in: J. Chen and S. B. Cooper, editors, *Theory and Applications of Models of Computation TAMC*, LNCS **5532** (2009), pp. 59–70.

[2] Garey, M. R. and D. S. Johnson, "Computers and Intractability," New York: Freeman, 1979.

[3] Mashburn, J. L., T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi and P. J. Slater, *Differentials in graphs*, Utilitas Mathematica **69** (2006), pp. 43–54.

[4] Speckenmeyer, E., *On feedback vertex sets and nonseparating independent sets in cubic graphs*, Journal of Graph Theory **3** (1988), pp. 405–412.

[5] Woeginger, G. J., *The toughness of split graphs*, Discrete Mathematics **190** (1998), pp. 295–297.

Enumeration of Labeled Split Graphs and Counts of Important Superclasses

Vladislav Bína

Department of Information Management, Faculty of Management in Jindrichuv Hradec, University of Economics in Prague
Jarosovská 1117/II, 37701 Jindrichuv Hradec, Czech Republic
bina@fm.vse.cz

Key words: Labeled Graphs, Graph Enumeration, Split Graphs, Decomposable Graphs, Undirected Graphs

1 Introduction

The contribution concerns counts of the labeled graphs in three well known and interesting graphical classes, namely split graphs and in the superclasses of decomposable (chordal) graphs and all undirected graphs.

The choice of labeled graphs is motivated by author's efforts in representation of multidimensional probability distributions and learning of probabilistic models where the labeled graphs can represent the structure of conditional independence relations among (labeled) variables (see e.g. [1]). The enumeration of unlabeled split graphs can be found in [2] but these figures are not interesting for our purpose.

This paper derives an exact formula for number of all different labeled split graphs on n vertices. The number of all labeled undirected graphs is very straightforward to derive; for graph on n vertices there are $2^{\binom{n}{2}}$ possible labeled undirected graphs. As far as we know, there is no theoretical formula for counts of labeled decomposable graphs; the enumeration was performed for graphs on up to 12 vertices in [3]. Results for these three graph classes are compared. Besides this the exact formula for counts of labeled split graphs is compared with the asymptotic approximation (published in [4]).

**The research was partially supported by Ministry of Education, Youth and Sports of the Czech Republic under project No. 2C06019: "Knowledge Mining and Modelling (ZIMOLEZ)" and University of Economics in Prague under project IG632021: "Algorithms for learning of compositional models and related graph classes".

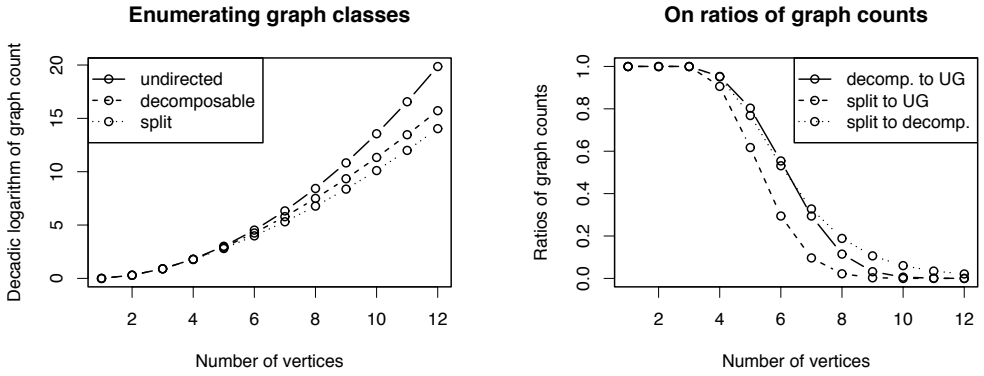


Fig. 1. The left part shows decadic logarithms of counts of decomposable, split and all undirected graphs. In the right part the proportion of decomposable and split graphs among all undirected graphs (dashed and full lines) can be seen. The dotted line denotes a ratio of split among decomposable graphs.

2 Properties and Enumeration of Split Graphs

Split graph is an undirected graph with vertex set which can be partitioned into two subsets such that one of them is clique and the other independent set. Split graphs can be easily recognized by means of degree sequence using the following assertion (see [5] or [6]).

Theorem 2.1 (Hammer and Simeone) *Let G be an undirected graph with degree sequence $d_1 \geq d_2 \geq \cdots \geq d_n$ and $k = \max\{i \,|\, d_i \geq i-1\}$. Then G is split if and only if*

$$\sum_{i=1}^{k} d_i = k(k-1) + \sum_{i=k+1}^{n} d_i.$$

Furthermore, its clique number is $\omega(G) = k$.

The class of split graphs is a subclass of both above mentioned classes. They are nested in the following way

$$\text{split graphs} \subseteq \text{decomposable graphs} \subseteq \text{undirected graphs}.$$

The contribution concerns a careful derivation of the exact formula (Theorem 2.2) which is verified for graphs up to eight vertices. This check is done by exhaustive testing of all undirected graphs whether they are split using Theorem 2.1.

Theorem 2.2 (Number of Split Graphs on n Vertices) *Number of all labeled split graphs on n vertices can be computed using the formula*

$$N_n^{\mathrm{S}} = 1 + \sum_{k=2}^{n} \binom{n}{k} \left[\left(2^k - 1\right)^{n-k} - \sum_{j=1}^{n-k} \frac{jk}{j+1} \binom{n-k}{j} \left(2^{k-1} - 1\right)^{n-k-j} \right].$$

This result, implementation of formula in R language and exact numbers of labeled split graphs up to twenty vertices can be found in Online Encyclopedia of Integer

Table 3. Comparison of counts in particular graph classes.

n	Number of split graphs	Number of decomposable graphs	Number of undirected graphs
1	1	1	1
2	2	2	2
3	8	8	8
4	58	61	64
5	632	822	1024
6	9654	18154	32768
7	202484	617675	2097152
8	5843954	30888596	268435456
9	233064944	2192816760	68719476736
10	12916716526	215488096587	35184372088832
11	998745087980	28791414081916	36028797018963968
12	108135391731689	5165908492061926	73786976294838206464

Sequences [7].

3 Comparisons of Graph Counts

Counts of labeled split graphs are compared with counts of important superclasses of decomposable and all undirected graphs for graphs up to 12 vertices (see Figure 1 and Table 3). For numbers of vertices up to twenty the exact counts of split graphs are compared to the asymptotic formula (see [4])

$$N_n^{\mathrm{S}} \sim \sum_k \binom{n}{k} 2^{k\cdot(n-k)} \tag{1}$$

which is quite precise particularly for higher numbers of vertices (see Table 4, especially focus on the trend of percentage error). For large graphs the asymptotic estimate appears to be sufficiently accurate for the vast majority of purposes, for instance the error for split graph on fifty vertices is smaller than one millionth. But the above cited article [4] claims that the Formula 1 also asymptotically estimates the number of decomposable graphs. In this case it seems that this asymptotics provides much less reasonable approximation. Moreover, the results in Table 3 and on Figure 1 seem to contradict to the assertion from the title of contribution [4], namely we claim that: “Almost all chordal graphs do not split”. But deeper analysis of this assertion lays already beyond the scope of this paper.

Table 4. Comparison of asymptotic counts and exact expression, percentage error for split graphs on 1 to 20 vertices.

n	1	2	3	4	5
Exact	1	2	8	58	632
Asymptotic	1	5	25	161	1441
Error	0.0 %	150.0 %	212.5 %	177.6 %	128.0 %
n	6	7	8	9	10
Exact	9654	202484	5843954	$2.33 \cdot 10^8$	$1.29 \cdot 10^{10}$
Asymptotic	18305	330625	8488961	$3.09 \cdot 10^8$	$1.60 \cdot 10^{10}$
Error	89.6 %	63.3 %	45.3 %	32.8 %	24.0 %
n	11	12	13	14	15
Exact	$9.99 \cdot 10^{11}$	$1.08 \cdot 10^{14}$	$1.64 \cdot 10^{16}$	$3.51 \cdot 10^{18}$	$1.06 \cdot 10^{21}$
Asymptotic	$1.17 \cdot 10^{12}$	$1.22 \cdot 10^{14}$	$1.80 \cdot 10^{16}$	$3.77 \cdot 10^{18}$	$1.11 \cdot 10^{21}$
Error	17.6 %	13.0 %	9.7 %	7.2 %	5.3 %
n	16	17	18	19	20
Exact	$4.49 \cdot 10^{23}$	$2.69 \cdot 10^{26}$	$2.28 \cdot 10^{29}$	$2.73 \cdot 10^{32}$	$4.62 \cdot 10^{35}$
Asymptotic	$4.67 \cdot 10^{23}$	$2.77 \cdot 10^{26}$	$2.33 \cdot 10^{29}$	$2.77 \cdot 10^{32}$	$4.67 \cdot 10^{35}$
Error	4.0 %	2.9 %	2.2 %	1.6 %	1.2 %

References

[1] Edwards, D., Havránek, T.: A fast procedure for model search in multidimensional contingency tables. Biometrika **72**(2), 339–351 (1985)

[2] Royle, G.F.: Counting Set Covers and Split Graphs. J. Integer Sequences **3**, 1–5 (2000)

[3] Castelo Valdueza, J.R.: The Discrete Acyclic Digraph Markov Model in Data Mining. PHD Thesis. Fac. Wiskunde en Informatica, Universiteit Utrecht (2002)

[4] Bender, E.A., Richmond, L.B., Wormald, N.C.: Almost All Chordal Graphs Split. J. Austral. Math. Soc., Series A **38**(2), 214–221 (1985)

[5] Hammer, P.L., Simeone, B.: The splittance of a graph. Combinatorica **1**(3), 275–284 (1981)

[6] Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs (Ann. Discrete Math., Vol. 57). North–Holland Publishing Co., Amsterdam (2004)

[7] Bína, V.: Sequence A179534 in The On-Line Encyclopedia of Integer Sequences, `http://oeis.org/A179534` (2010)

Local Computation of Vertex Colorings

Thomas Böhme, Jens Schreyer

TU-Ilmenau, Fakultät für Mathematik und Naturwissenschaften, Postfach 100565, 98684 Ilmenau

{thomas.boehme,jens.schreyer}@tu-ilmenau.de

Key words: vertex coloring, distributed computing

Abstract

We investigate algorithms for local vertex colorings in the following sense. Given a graph G the vertices are colored in rounds by agents corresponding to the vertices of the graph. In each round, each agent decides for a color out of the set $\{1, ..., k\}$ for the corresponding vertex in G and observes the colors of its neighbors. Kearns et. al [3] proposed to embed the problem into a game by giving each vertex a reward if there is no conflict between its color and the colors of its neighbors. Later Chaudhuri et. al. [2] presented a random local algorithm based on that game. The algorithm colors a given graph G of order n properly with probability $1 - \delta$ in $O(\log \frac{n}{\delta})$ rounds using $\Delta(G) + 2$ colors, where $\Delta(G)$ is the maximum degree of G. Moreover, in [1] it is shown, that a simple local algorithm for the coloring of G with $\chi(G)$ colors exists, but may take a lot of rounds to converge.
In this paper we present a random local algorithm which colors G properly in $O(n \log \frac{n}{\delta})$ rounds with probability $1 - \delta$ using k colors, where k is any upper bound on $col(G) + 1$ and $col(G)$ is the coloring number of G.

1 Introduction

Starting with the pioneering work of Linial [4] a rich literature of what can be done and what cannot be done using only local information emerged in the field of distributed computing. Lineal proposed a computational model characterized by the following assumptions:

- Every vertex of the graph is endowed with a unique identifier.
- The computation happens synchronously in rounds.
- In every round every vertex of the graph may send a message of arbitrary size to some or all of its neighbors.
- Every vertex may execute an arbitrarily difficult computation in one time step.

The complexity measure for this model is the number of communication rounds. This is because the communication usually takes more time than the computation. Using this model it is clear, that a local algorithm should take no more rounds than the diameter $diam(G)$ of the graph, since each vertex can explore the complete structure of the graph G in $diam(G)$ communication rounds. Lineal [4] showed that graph colorings are hard to compute in this setting by proving that every algorithm, that fits the model and has the property to compute a proper coloring of a d-ary tree of depth r within at most $\frac{2}{3}r$ communication rounds, will need at least $\sqrt{d}$ different colors.
Here we propose a different approach inspired by the paper of Kearns et al[3] on an experimental study of social network behavior. Their idea is, that the agents corresponding to the vertices of the network graph have no common goal but own incentives, and the solution of the graph problem corresponds to a Nash equilibrium or another suitable solution of a game that reflects these incentives. In contrast to Linial's model described above there is no direct communication between the agents but only some observation of the other agents behavior. Later Chaudhuri et al. [2] theoretically investigated this game. They show, that if the number of colors available to the agents is at least $\Delta(G) + 2$, then with arbitrarily high probability $1 - \delta$ the graph is colored properly within $O(\log \frac{n}{\delta})$ rounds, where n denotes the number of vertices. Moreover, in [1] it has been shown, that even if the behavior of other agents is not observable and each agent only knows its reward in each round, a coloring with $\chi(G)$ colors is possible, where $\chi(G)$ is the chromatic number of the graph, but it may take very long to achieve that aim.

2 Preliminaries

A graph G is a k-degenerate graph if every subgraph H of G contains a vertex of degree at most k. Let $[k]$ denote the set $\{1, ..., k\}$. For every k-degenerate graph G there exists an enumeration $x_1, ..., x_n$ of its vertices such that for each $i \in [n]$ the vertex x_i has at most k neighbors in the set $\{x_1, ..., x_i\}$. Now the vertices can be colored successively with $k+1$ colors using a greedy coloring. The coloring number $col(G)$ of a graph G is the smallest integer k such that G is $(k-1)$-degenerate. Hence, for every graph G

$$\chi(G) \leq col(G) \leq \Delta(G) + 1.$$

The gaps between the numbers $\chi(G)$, $col(G)$, and $\Delta(G) + 1$ may be arbitrarily large.

3 Results

We consider the following setup for a k-degenerate graph $G = (V, E)$ on n vertices which is to be colored. Every vertex $x \in V$ knows its neighborhood $N(x)$ and can see the color of each neighbor. The graph is colored in rounds. Each vertex x stores 2 sets $P(x)$ and $S(x)$, which are initially empty. In every step of the algorithm $P(x)$ contains at most k vertices. Moreover, after at most $|V|$ rounds these sets do not change anymore, and for every edge $\{x, y\}$ of G it holds that $x \in P(y)$ or $y \in P(x)$. Hence, it will be sufficient if every vertex x is colored differently from the vertices of $P(x)$. The following distributed algorithm works in discrete time steps $t = 1, 2...$ In each time step, each vertex executes the following pseudocode, where $c(x)$ denotes the color of x, which is initially set to 1.

Algorithm

1 **if** $P(x) = \emptyset$
2 **then**
3 $S(x) \leftarrow S(x) \cup \{y \in N(x) \mid c(y) \neq 1\}$
4 **if** $|(N(x) \setminus S(x)| \leq k$
5 **then** $P(x) \leftarrow N(x) \setminus S(x)$
6 **elseif** $\exists y \in P(x)\ c(y) = c(x)$
7 **then** choose $c(x)$ uniformly at random from the set $[k + 2] \setminus \{c(y) \mid y \in P(x)\}$

Now it can be shown, that if all agents apply the algorithm, the graph will eventually be colored properly:

Theorem 3.1 *Let δ be any positive number. If all vertices of a k-degenerate graph G on n vertices apply the algorithm then:*

$$Prob(G \text{ is colored properly after } t \text{ rounds}) \geq 1 - \delta,$$

where $t = \left\lceil n + (\log \frac{2^k}{2^k - 1})^{-1} \cdot n \log \frac{n}{\delta} \right\rceil$.

Moreover, if the degeneracy of the graph is unknown, but there is an upper bound on the total number of vertices known by all agents, a modification of the algorithm leads to a similar result. Furthermore, it turns out, that the result still holds, if the synchrony assumption of the model is dropped.

References

[1] Böhme,T., Schreyer,J. A game theoretic approach to graph problems,*9th International Conference on Innovative Internet Community Systems, 2009*

[2] Chaudhuri, K., Chung, F., and Jamall M.S., A network coloring game, *WINE* 2008: 522-530

[3] Kearns, M., Suri, S., and Montfort, N., An experimental study of the coloring problem on human subject networks, *Science* Vol 313 (2006), 824-827
[4] Linial, N., Locality in distributed graph algorithms, *SIAM J. Comput* 21(1): 193-201 (1992)

A primal algorithm for the minimum weight clique cover problem on a class of claw-free perfect graphs

Flavia Bonomo, [a] [1] Gianpaolo Oriolo, [b] Claudia Snels, [b]

[a] *CONICET and Departamento de Computación, FCEyN, Universidad de Buenos Aires, Argentina.*
fbonomo@dc.uba.ar

[b] *Dipartimento di Informatica, Sistemi e Produzione, Università di Roma Tor Vergata, Italia*
{snels, oriolo}@disp.uniroma2.it

Key words: claw-free graphs, perfect graphs, minimum weight clique cover

---

On a perfect graph $G(V, E)$ with a strictly positive integer weight function on the vertices $w(v)$, $v \in V$, the minimum weight clique cover problem (MWCC for short) is the dual problem of the maximum weight stable set (MWSS for short) and it consists of building a collection of cliques $\mathcal{C}$, each one with a non negative weight y_C, such that, for each $v \in V$, $\sum_{C \in \mathcal{C}: v \in C} y_C \geq w(v)$, and $\sum_{C \in \mathcal{C}} y_C$ is minimum. The MWCC can be solved in polynomial time on perfect graphs with the ellipsoid method, however, for particular classes of perfect graphs, there also exist poly-time combinatorial algorithms.

This is the case, for instance, for claw-free perfect graphs, where a combinatorial algorithm has been proposed by Hsu and Nemhauser [3]. (A graph is claw-free if none of its vertices has a stable set of size three in its neighborhood.) To the best of our knowledge, this is so far the only available combinatorial algorithm to solve the problem in claw-free perfect graphs. The algorithm by Hsu and Nemhauser relies on a crucial property of the MWCC in (general) perfect graphs, namely, there always exists a clique which intersects all the maximum weight stable sets: we will call such a clique *crucial*. In fact, it is shown in [3] that a crucial clique can be found in polynomial time in claw-free perfect graphs, provided that we are able to find a maximum weight stable set in polynomial time: for the latter problem, we have, nowadays, several algorithms (e.g. [4,1]).

The algorithm in [3] is then heavily based on the solution of the MWSS, so, in a way, the algorithm it is essentially a *dual* algorithm. In this paper, we propose an algorithm for the MWCC for a relevant subclass of claw-free perfect graphs which

[1] Partially supported by ANPCyT PICT-2007-00518 and PICT-2007-00533, and UBACyT Grants X069 and 20020090300094.

is essentially *primal*, in the sense that it builds an optimal clique cover, without solving *any* MWSS. The algorithm exploits the following algorithmic decomposition theorem by Faenza et al. [1] for quasi-line graphs. A graph is quasi-line if the neighborhood of each vertex can be covered by two cliques. Quasi-line graphs are a generalization of line graphs and a subclass of claw-free graphs, and it is well known that a graph is claw-free perfect if and only if it is quasi-line perfect.

Theorem 4 [1] *Let $G(V, E)$ be a connected quasi-line graph with n vertices. In time $O(|V|^3)$ we can:*

(i) either recognize that G is net-free;

(ii) or provide a decomposition into $k \leq n$ strips $(G^1, \mathcal{A}^1), \ldots, (G^k, \mathcal{A}^k)$, with respect to a partition $\mathcal{P}$, such that each graph G^j is distance simplicial with respect to each clique $A \in \mathcal{A}^j$.

Our new algorithm is designed for case (ii), i.e. when the claw-free perfect graph G is the composition of strips.

For a set $S \subseteq V$, let $N(S) := \bigcup_{v \in S} N(v) \setminus S$, $N_j(S) := \{v \in V \setminus S :$ the minimum distance between v and $s \in S$ is $j\}$, $j \geq 1$, where $N(v) := \{u : uv \in E\}$. A clique K is distance simplicial in a graph D if, for every j, $\alpha(N_j(K)) \leq 1$. A connected graph D is distance simplicial w.r.t. a clique K if K is distance simplicial in D.

A strip $(G, \mathcal{A})$ is a graph G (not necessarily connected) with a multi-family $\mathcal{A}$ of either one or two designated non-empty cliques of G. Let $\mathcal{G} = (G^1, \mathcal{A}^1), \ldots, (G^k, \mathcal{A}^k)$ be a family of k vertex disjoint strips, and let $\mathcal{P}$ be a partition of the multi-set of the cliques in $\mathcal{A}^1 \cup \ldots \cup \mathcal{A}^k$. The composition of the k strips w.r.t. $\mathcal{P}$ is the graph G such that is obtained from the union of $G^1, \ldots, G^k$, by making adjacent vertices of $A \in \mathcal{A}^i$ and $B \in \mathcal{A}^j$ (i, j not necessarily different) if and only if A and B are in the same class of the partition $\mathcal{P}$.

We follow the methodology proposed in [5] for the MWSS. We replace each strip with a suitable simple graph, a *gadget*, and we assign to the vertices of the gadget the value of some special minimum weight clique cover computed on the strip. Each gadget is designed in order to produce a graph G', that is the composition of the gadgets, which is line and perfect. We show that a minimum weight clique cover of G can be easily built as soon as we know: 1) a minimum weight clique cover of G' ; 2) a minimum weight clique cover of a graph that is distance simplicial w.r.t. some clique.

1) can be easily managed in the following way. Let us denote with H the root graph of G'; by construction H is a line perfect graph (i.e the root graph of a perfect line graph). Every clique of G' corresponds in H to a set of edges $\delta(v) := \{uv \in E : u \in N(v)\}$, that we call a *star* or to a set of edges inducing a triangle. It is then straightforward that finding a MWCC of G' is equivalent to assigning a non negative value z to each star S and triangle T of H such that $\sum_{S:uv \in S} z_S + \sum_{T:uv \in T} z_T \geq w_e$ $\forall e \in E(H)$ and $\sum_{S \in \mathcal{S}} z_S + \sum_{T \in \mathcal{T}} z_T$ is minimum, where $\mathcal{S}$ and $\mathcal{T}$ are respectively the collection of stars and triangles of H. From a theorem of Trotter ([6]) we have that this problem in a line perfect graph is exactly the dual of the maximum weight matching. Therefore we can apply to H one of the many available primal dual algorithms for the maximum weight matching and obtain a half integer solution for the

problem (more precisely, only variables associated to the stars of H can have an half integer value). The half integer solution can be easily adjusted to an integer solution using an algorithm proposed again by Trotter [6] which solves the unweighted version of the edge cover problem with stars and triangles in line perfect graphs. The resulting complexity of computing a minimum weight clique cover of G' is then $O(|V(H)|^2 log|V(H)|) = O(|V(G')|^2 log|V(G')|) = O(|V(G)|^2 log|V(G)|)$ (using the primal dual algorithm for maximum weight matching by Gabow [2]).

In order to solve 2) we have designed a primal algorithm for the MWCC in graphs which are distance simplicial w.r.t. some clique. First we observe that in every graph, given a clique K s.t. $N(K)$ is a clique, then $K \cup \{v \in N(K) : v$ is complete to $K\}$ is a crucial clique. Using this observation and the structure of graphs which are distance simplicial w.r.t. a clique K, we have developed Algorithm 3.

Require: A graph $D(V, E, w)$ that is distance simplicial graph w.r.t. a clique K_1. (Assume $K_{j+1} := N_j(K_1) \neq \emptyset$, for every $1 \leq j \leq t$, $N_{t+1}(K_1) = \emptyset$).

Ensure: A MWCC for $D(V, E, w)$.

1: Let $i \leftarrow 1$; $Q \leftarrow V$; y = 0;

2: While $Q \neq \emptyset$ let $D \leftarrow D[Q]$ and do:

2.1 Let $j \in [t]$ be such that $K_1 \cap Q = \ldots = K_{j-1} \cap Q = \emptyset$ and $K_j \cap Q \neq \emptyset$.

2.2 Let $K \leftarrow K_j \cup \{v \notin K_j : v$ is complete to K_j in the graph $D[Q]\}$.

2.3 Let $\bar{v}$ be the vertex of K with minimum (current) weight w.

2.4 Let $Q \leftarrow Q \setminus \{v \in K : w(v) = w(\bar{v})\}$.

2.5 For each $v \in K$, let $w(v) \leftarrow w(v) - w(\bar{v})$.

2.6 Let $y_K \leftarrow w(\bar{v})$.

3: **Return** y.

We briefly describe Algorithm 3. In step 2.2, we build a crucial clique K following our previous observation and exploiting the "levels structure" of D. Then we have to assign a value to this clique (step 2.6), and we update the weight function of D, deleting the vertices with weight 0 (steps 2.4 and 2.5). After one iteration the resulting graph may not be distance simplicial w.r.t. K_1, but we can prove that still maintains all the nice structural properties we need to let the algorithm run correctly. Beside the selection of the crucial clique, the "levels structure" of D is used to show that at each iteration the value of the MWSS of D is dropping exactly by y_K, or in other words we are building pieces of an optimal clique cover. It is trivial to observe that steps 2.1 to 2.6 can be implemented as to run in $O(|V(D)|)$-time, and we can easily observe that they will be repeated at most $|V(D)|$ because each time we perform step 2.4 cardinality of the set Q strictly decreases, therefore Algorithm 3 runs in time $O(|V(D)|^2)$.

In order to build each gadget, Algorithm 3 is run a constant number of times for every strip, so we can conclude that we can build G' in time $O(|V(G)|^2)$.

We can conclude that our resulting algorithm solves the MWCC for claw-free perfect graphs from case (ii) in Theorem 4 in time $O(|V(G)|^2 log|V(G)|)$, which substantially improves upon the complexity of the algorithm by Hsu and Nemhauser, running in time $O(|V(G)|^6)$. We are now dealing with case (i), and we believe we

can manage this case also, but with a slightly different technique.

References

[1] Faenza, Y., G. Oriolo and G. Stauffer, *An algorithmic decomposition of claw-free graphs leading to an $O(n^3)$-algorithm for the weighted stable set problem*, in: D. Randall, editor, *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, 2011, pp. 630–646.

[2] Gabow, H., *Data structures for weighted matching and nearest common ancestors with linking*, in: *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, 1990, pp. 434–443.

[3] Hsu, W. and G. Nemhauser, *Algorithms for maximum weight cliques, minimum weighted clique covers and cardinality colorings of claw-free perfect graphs*, Annals of Discrete Mathematics **21** (1984), pp. 317–329.

[4] Minty, G., *On maximal independent sets of vertices in claw-free graphs*, Journal of Combinatorial Theory. Series B **28** (1980), pp. 284–304.

[5] Oriolo, G., U. Pietropaoli and G. Stauffer, *A new algorithm for the maximum weighted stable set problem in claw-free graphs*, in: A. Lodi, A. Panconesi and G. Rinaldi, editors, *Proceedings of the 13th Integer Programming and Combinatorial Optimization International Conference*, Lecture Notes in Computer Science **5035**, Bertinoro, Italy, 2008, pp. 77–96.

[6] Trotter, L., *Line perfect graphs*, Mathematical Programming **12** (1977), pp. 255–259.

Characterization of classical graph classes by weighted clique graphs

F. Bonomo, [a] [1] J. L. Szwarcfiter [b]

[a] *CONICET and Departamento de Computación, FCEyN, Universidad de Buenos Aires, Argentina.*
fbonomo@dc.uba.ar

[b] *COPPE and NCE, Universidade Federal do Rio de Janeiro, Brazil.*
jayme@nce.ufrj.br

Key words: weighted clique graphs, graph classes structural characterization.

1 Introduction

A *complete set* is a set of pairwise adjacent vertices. A *clique* is a complete set that is maximal under inclusion. We will denote by $M_1, \ldots, M_p$ the cliques of G, and by $\mathcal{C}_G(v)$ the set of cliques containing the vertex v in G.

Consider a finite family of non-empty sets. The *intersection graph* of this family is obtained by representing each set by a vertex, two vertices being connected by an edge if and only if the corresponding sets intersect. The *clique graph* $K(G)$ of G is the intersection graph of the cliques of G.

Clique graphs have been characterized by Roberts and Spencer in [13], but the problem of deciding if a graph is a clique graph is NP-complete [1].

Let $\mathcal{F}$ be a family of subsets of a set S. $\mathcal{F}$ is *separating* when, for each x in S, the intersection of all the subsets in $\mathcal{F}$ containing x is $\{x\}$; $\mathcal{F}$ satisfies the *Helly property* when every subfamily of it consisting of pairwise intersecting subsets has a common element. A graph is *clique-Helly* when its cliques satisfy the Helly property. Clique-Helly graphs are clique graphs [7].

Given a graph H, a *weighting of H of size m*, or *m-weighting of H*, consists on giving a weight w to every complete set of H of size m.

A *weighting of $K(G)$ of size m*, consists on defining the weight w for a subset of its vertices $\{M_{i_1}, \ldots, M_{i_m}\}$ as $w(M_{i_1}, \ldots, M_{i_m}) = |M_{i_1} \cap \ldots \cap M_{i_m}|$. (In the right-hand side, we consider $M_{i_1}, \ldots, M_{i_m}$ as cliques of G.) We will denote by $K^w_{m_1,\ldots,m_\ell}(G)$ the clique graph of G with weightings of sizes $m_1, \ldots, m_\ell$.

[1] Partially supported by ANPCyT PICT-2007-00518 and PICT-2007-00533, and UBACyT Grants X069 and 20020090300094 (Argentina), CAPES/SPU Project CAPG-BA 008/02 (Brazil-Argentina), and Math-AmSud Project 10MATH-04 (France-Argentina-Brazil).

Weighted clique graphs with weightings restricted to size 2 were considered in [9,10], and in [3,4,5,6,8,11,12,14], in the context of chordal graphs.

In this work, we give a characterization of weighted clique graphs similar to Roberts and Spencer's characterization for clique graphs, and we characterize several classical and well known graph classes by means of their weighted clique graph. We prove a characterization of hereditary clique-Helly graphs in terms of K_3^w and show that $K_{1,2}^w$ is not sufficient to characterize neither hereditary clique-Helly graphs nor clique-Helly graphs. For chordal graphs and their subclass UV graphs, we obtain a characterization by means of $K_{2,3}^w$. We show furthermore that $K_{1,2}^w$ is not sufficient to characterize UV graphs. We describe also a characterization of interval graphs in terms of $K_{2,3}^w$ and of proper interval graphs in terms of $K_{1,2}^w$. Besides, we prove that $\{K_1^w, K_2^w\}$ is not sufficient to characterize proper interval graphs. For split graphs, we give a characterization by means of $K_{1,2}^w$, and prove that $\{K_1^w, K_2^w\}$ is not sufficient to characterize split graphs. Due to a lack of space, for definitions of graph classes we refer the reader to [2].

2 Main results

The characterization of clique graphs is as follows.

Theorem 5 (Roberts and Spencer, 1971 [13]) *A graph H is a clique graph if and only if there is a collection $\mathcal{F}$ of complete sets of H such that every edge of H is contained in some complete set of $\mathcal{F}$, and $\mathcal{F}$ satisfies the Helly property.*

A similar characterization for 2-weighted clique graphs was presented in [9,13]. We can extend this characterization to weighted graphs.

Theorem 6 *Let H be a graph, provided with weightings w of sizes $m_1, \ldots, m_\ell$. Then there exists a graph G such that $H = K^w_{m_1,\ldots,m_\ell}(G)$ if and only if there is a collection $\mathcal{F}$ of complete sets of H, not necessarily pairwise distinct, such that $\mathcal{F}$ satisfies the Helly property and is separating, every edge of H is contained in some set of $\mathcal{F}$, and for every $1 \leq j \leq \ell$, each complete set $M_{i_1}, \ldots, M_{i_{m_j}}$ of H is contained in exactly $w(M_{i_1}, \ldots, M_{i_{m_j}})$ sets of $\mathcal{F}$.*

It would be interesting to analyze the computational complexity of deciding if a weighted graph is a weighted clique graph. For 1-weightings, we show that the problem is NP-complete. It remains as an open question to analyze the problem for other weighting sizes.

Some graph classes can be naturally defined in terms of their weighted clique graphs, for example, a graph G is clique-Helly if and only if $K^w_{3,\ldots,\omega(K(G))}(G)$ satisfies $w(M_{i_1}, \ldots, M_{i_\ell}) > 0$ for every complete set $M_{i_1}, \ldots, M_{i_\ell}$ of $K(G)$.

The examples in Figure 1 show that $K_{1,2}^w$ is not sufficient to characterize neither hereditary clique-Helly graphs nor clique-Helly graphs. Nevertheless, we can obtain a characterization of hereditary clique-Helly graphs in terms of K_3^w.

Theorem 7 *Let G be a graph. Then G is hereditary clique-Helly if and only if $K_3^w(G)$ satisfies $w(M_i, M_j, M_k) \geq \min\{w(M_i, M_j, M_\ell), w(M_j, M_k, M_\ell), w(M_i, M_k, M_\ell)\}$, for every complete set M_i, M_j, M_k, M_ℓ of size four in $K(G)$.*

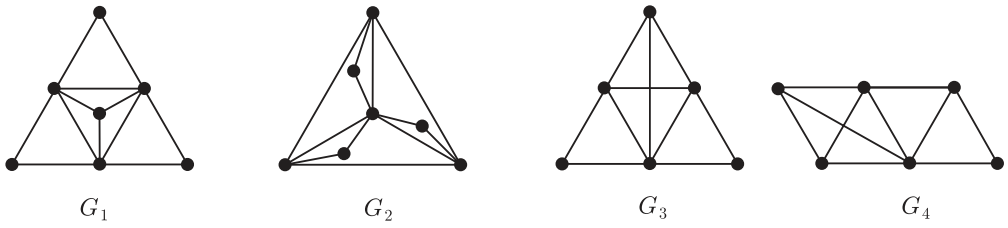


Fig. 1. Graphs G_1 and G_2 satisfy $K^w_{1,2}(G_1) = K^w_{1,2}(G_2)$, but G_2 is hereditary clique-Helly and G_1 is not even clique-Helly. Besides, G_1 is UV and G_2 is not. Graphs G_3, G_4 satisfy $K^w_1(G_3) = K^w_1(G_4)$ and $K^w_2(G_3) = K^w_2(G_4)$, but G_3 is split and G_4 is not, while G_4 is proper interval and G_3 is not.

The examples in Figure 1 show that K^w_1 and K^w_2 do not characterize split graphs. A characterization of split graphs in terms of $K^w_{1,2}$ is the following.

Theorem 8 *A graph G is split and connected if and only if $K^w_{1,2}(G)$ is a star with center M_1, and $w(M_1, M_j) = w(M_j) - 1$, $2 \leq j \leq |K(G)|$.*

For interval and proper interval graphs, the following characterizations hold.

Theorem 9 *A graph G is an interval graph if and only if $K^w_{2,3}(G)$ admits a linear ordering $M_1, \ldots, M_p$ of its vertices such that for every $1 \leq i < j < k \leq p$, $w(M_i, M_j, M_k) = w(M_i, M_k)$.*

Theorem 10 *A graph G is a proper interval graph if and only if $K^w_{1,2}(G)$ admits a linear ordering $M_1, \ldots, M_p$ of its vertices such that for every triangle M_i, M_j, M_k, $1 \leq i < j < k \leq p$, $w(M_j) = w(M_i, M_j) + w(M_j, M_k) - w(M_i, M_k)$.*

The examples in Figure 1 show that K^w_1 and K^w_2 are not sufficient to characterize proper interval graphs.

It is a known result that clique graphs of chordal graphs are dually chordal graphs. Moreover, it holds that, for a chordal graph G, there is some canonical tree T of $K(G)$ such that, for every vertex v of G, the subgraph of T induced by $\mathcal{C}_G(v)$ is a subtree. Such a tree is called a *clique tree* of G. McKee proved [12] that those trees are exactly the maximum weight spanning trees of $K^w_2(G)$.

Theorem 11 *A connected graph G is chordal if and only if $K^w_{2,3}(G)$ admits a spanning tree T such that for every three different vertices M_i, M_j, M_k of T, if M_j belongs to the path M_i—M_k in T, then $w(M_i, M_j, M_k) = w(M_i, M_k)$.*

A graph is a *UV graph* if it is the intersection graph of paths of a tree. So, UV graphs are a subclass of chordal graphs.

Theorem 12 *A connected graph G is UV if and only if $K^w_{2,3}(G)$ admits a spanning tree T such that for every three different vertices M_i, M_j, M_k of T, if M_j belongs to the path M_i—M_k in T, then $w(M_i, M_j, M_k) = w(M_i, M_k)$, and for every M in T and M_i, M_j, M_k in $N_T(M)$, it holds $w(M_i, M_j, M_k) = 0$.*

The examples in Figure 1 show that $K^w_{1,2}$ does not characterize UV graphs.

References

[1] L. Alcón, L. Faria, C. de Figueiredo, and M. Gutierrez, The complexity of clique graph recognition, *Theor. Comput. Sci.* **410** (2009), 2072–2083.

[2] A. Brandstädt, V.B. Le, and J.P. Spinrad, *Graph Classes: A Survey*, SIAM Monographs on Discrete Mathematics, vol. 3, SIAM, Philadelphia, 1999.

[3] F. Gavril, Generating the maximum spanning trees of a weighted graph, *J. Algorithms* **8** (1987), 592–597.

[4] M. Gutierrez, J. Szwarcfiter, and S. Tondato, Clique trees of chordal graphs: leafage and 3-asteroidals, *Electron. Notes Discrete Math.* **30** (2008), 237–242.

[5] M. Habib and J. Stacho, A decomposition theorem for chordal graphs and its applications, *Electron. Notes Discrete Math.* **34** (2009), 561–565.

[6] M. Habib and J. Stacho, *Reduced clique graphs of chordal graphs*, manuscript, 2010.

[7] R. Hamelink, A partial characterization of clique graphs, *J. Combin. Theory, Ser. B* **5** (1968), 192–197.

[8] I.J. Lin, T.A. McKee, and D.B. West, The leafage of a chordal graph, *Discuss. Math., Graph Theory* **18** (1998), 23–48.

[9] T.A. McKee, Clique multigraphs, In: *Graph Theory, Combinatorics, Algorithms and Applications* (Y. Alavi, F.R.K. Chung, R.L. Graham, and D.S. Hsu, eds.), SIAM, Philadelphia, 1991, pp. 371–379.

[10] T.A. McKee, Clique pseudographs and pseudo duals, *Ars Combin.* **38** (1994), 161–173.

[11] T.A. McKee, Restricted circular-arc graphs and clique cycles, *Discrete Math.* **263** (2003), 221–231.

[12] T.A. McKee and F.R. McMorris, *Topics in Intersection Graph Theory*, SIAM, Philadelphia, 1999.

[13] F. Roberts and J. Spencer, A characterization of clique graphs, *J. Combin. Theory, Ser. B* **10** (1971), 102–108.

[14] Y. Shibata, On the tree representation of chordal graphs, *J. Graph Theory* **12**(2–3) (1998), 421–428.

The gateway location problem for hazardous material transportation

M. Bruglieri, [a] P. Cappanera, [b] M. Nonato [c]

[a] *INDACO, Via Durando 38/a, 20158 Milano (Italy)*
maurizio.bruglieri@polimi.it

[b] *DSI, Via Santa Marta 3, 50139 Firenze (Italy)*
paola.cappanera@unifi.it

[c] *ENDIF, Via Saragat 1, 44100 Ferrara (Italy)*
nntmdl@unife.it

Key words: gateway location, multicommodity flow, hazmat transport

1 Introduction

Everyday in our cities the shipments of hazardous materials such as explosives, gases, flammable liquids, toxic substances, and radioactive materials are demanded. These shipments are indispensable to our modern way of life (to fuel vehicles, to heat buildings, to carry out hospital therapies, etc.), although they can be harmful to the people and the environment in case of an accident. While there are many possible interventions for risk mitigation [4], here we focus on the policies of government agencies to regulate the itineraries of the hazmat shipments. Two main research lines can be identified in the literature: i) enforce specific itineraries to vehicles; ii) prescribe a set of rules that vehicles itineraries have to respect. The first research line tackles determining, for a single origin-destination pair, the set of routes of minimum total risk which spread risk the most evenly over the territory, leading to the search for spatially dissimilar paths e.g. by means of the Iterative Penalty Method, by mean of the Gateway Shortest Path [1], or by exploiting the k-shortest paths [6]. However, often (e.g. in Europe and North America) government agencies do not have the authority to dictate routes to hazmat carriers. Therefore, a second research line has focused on the indirect control of hazmat shipments by way of rules that carriers have to respect: the closure of certain road segments to hazmat vehicles solved either exactly [2] or heuristically [3]; the *toll-setting policy* [7] where tolls are used to deter hazmat carriers from using certain roads and to channel the shipments on less-populated roads. In the present work we propose an alternative policy tool for indirectly regulating the hazmat transport. It consists of imposing check points along the routes, so called *gateways*, and stating the rule that each carrier, on its route from origin to destination, must pass through a specific

gateway. Each carrier will select the minimum cost detour by the assigned check point. In turn, the administrator will select the location of a given number of check points on the network, and which check point has to be assigned to which carrier, so that carrier responses will minimize total risk. We call such problem the *Gateway Location Problem for multicommodity flow* (GLP). To the best of our knowledge, this is the first work that formalizes GLP and proposes it as an effective means to mitigate hazmat transport risk. While we are unaware of the use of this policy tool by regulators around the world, the road infrastructure of several cities is sufficiently technologically advanced to implement the gateway policy. For instance [5] describes a tracking technology, used by the Municipality of Milan, based on automated plate number recognition at some gates. The same technology could be employed to implement the risk mitigation method here proposed, where gateways are used not for tracking vehicles but for rerouting.

2 A bilevel multicommodity flow formulation

We model the GLP by way of an arc based formulation which captures the hierarchical relationship between the location-assignment decisions and the routing ones. Indeed, the network administrator locates gateways at selected candidate sites and assigns them to carriers in order to optimize its own objective function r over the shortest gateway paths, that is, over the rational users reaction set. This interaction is captured by a bilevel multicommodity uncapacitated minimun cost flow problem. Let us introduce some mathematical notation. Let $V = \{1 \dots, n\}$ be the set of carriers, each associated with its origin-destination pair, (o_v, d_v) and shipment request $\varphi_v,\ \forall v \in V$. Let N^g be the set of the m available candidate sites for hosting one of the k gateways, with $k < n$ and $k << m$. $G = (N, A)$ is a weighted directed graph, with $N^g \cup \{o_v : v \in V\} \cup \{d_v : v \in V\} \subseteq N$, and $A \subseteq N \times N$; $\forall (i, j) \in A$ two non-negative coefficients c_{ij} and r_{ij} are defined, being the coefficients of the network users (cost) and the administrator (risk) objective functions. Let ρ_v^c (ρ_v^r) denote the c-optimal (r-optimal) path from o_v to d_v for each $v \in V$. $y_h,\ \forall h \in N^g$ are binary variables to select open gateways, $z_h^v, \forall\ h \in N^g,\ \forall v \in V$ are binary variables that assign a gateway to a carrier, while variables $\gamma_v,\ \forall v \in V$ model the possibility of letting carrier v flow along ρ_v^c (if its risk cannot be decreased by way of a gateway path deviation). Each carrier, if assigned to gateway h, will route its flow along the shortest path from o_v to h and the one from h to d_v, i.e., the shortest gateway path. Each subpath is modeled by a separate family of flow variables, namely: flow variables $\overline{x}_{ij}^v,\ \forall v \in V,\ \forall (i, j) \in A$ for the path from o_v to the assigned gateway or along ρ_v^c, while $\underline{x}_{ij}^v,\ \forall v \in V,\ \forall (i, j) \in A : i \neq o_v$ for the path from the assigned gateway to destination d_v. As usual, $\delta_i^+(\overline{x})$ stands for $\sum_{(i,j)\in FS(i)} \overline{x}_{ij}$ and $\delta_i^-(\overline{x})$ for $\sum_{(j,i)\in BS(i)} \overline{x}_{ji}$; the same notation holds for flow variables $\underline{x}_{ij}$. Now we can provide a mathematical formulation for GLP.

$$P^{BL.MCF} : \min \sum_{v=1\dots n} \varphi_v \sum_{(i,j)\in A} r_{ij}(\overline{\xi}_{ij}^v + \underline{\xi}_{ij}^v) \quad \text{subject to:}$$

$$\sum_{h=1,\ldots,m} z_h^v = 1 - \gamma_v \, \forall v \in V \tag{1}$$

$$y_h \geq z_h^v \quad \forall h \in N^g, \forall v \in V \tag{2}$$

$$\sum_{h=1,\ldots,m} y_h = k \tag{3}$$

$$z_h^v \in \{0,1\} \, \forall h \in N^g, \forall v \in V \tag{4}$$

$$\gamma_v \in \{0,1\} \qquad \forall v \in V \tag{5}$$

$$y_h \in \{0,1\} \qquad \forall h \in N^g \tag{6}$$

where $\overline{\xi}_{ij}^v$, $\underline{\xi}_{ij}^v \in argmin \, P^{SP} : \min \sum_{v \in V} \sum_{(i,j) \in A} c_{ij}(\overline{x}_{ij}^v + \underline{x}_{ij}^v)$ subject to:

$$\delta_{o_v}^+(\overline{x}^v) = 1 \qquad \forall v \in V \tag{7}$$

$$\delta_h^-(\overline{x}^v) - \delta_h^+(\overline{x}^v) = z_h^v \qquad \forall h \in N^g, \forall v \in V \tag{8}$$

$$\delta_i^-(\overline{x}^v) - \delta_i^+(\overline{x}^v) = 0 \, \forall i \neq o_v, d_v, \, i \notin N^g, \forall v \in V \tag{9}$$

$$\overline{x}_{ij}^v \geq 0 \qquad \forall (i,j) \in A, \forall v \in V \tag{10}$$

$$\delta_{d_v}^-(\overline{x}^v) = \gamma_v \qquad \forall v \in V \tag{11}$$

$$\delta_{d_v}^-(\underline{x}^v) = 1 - \gamma_v \qquad \forall v \in V \tag{12}$$

$$\delta_h^+(\underline{x}^v) - \delta_h^-(\underline{x}^v) = z_h^v \qquad \forall h \in N^g, \forall v \in V \tag{13}$$

$$\delta_i^+(\underline{x}^v) - \delta_i^-(\underline{x}^v) = 0 \, \forall i \neq o_v, d_v, \, i \notin N^g, \forall v \in V \tag{14}$$

$$\underline{x}_{ij}^v \geq 0 \qquad \forall (i,j) \in A, \forall v \in V \tag{15}$$

Variables z_v^h and γ_v are decision variables at the outer level and right hand side coefficients of the flow balance constraints at the inner level. Cardinality constraints (3) impose that exactly k gateways are installed at that many locations, while (1) assign to each carrier either one open gateway or ρ_v^c. Indeed, when $\gamma_v = 1$ the flow goes from o_v to d_v along ρ_v^c and no gateway is assigned to carrier v. Constraints (2) ensure that a gateway must be open in order to be assigned.

Due to the lack of capacity constraints in the inner flow problem, this one is separable into $2n$ shortest path problems, namely, $SP_h^v(\overline{x})$ (7–11) and $SP_h^v(\underline{x})$ (12–15). While bilevel programming is usually hard to tackle, we will show how to get rid of the inner level and reformulate the problem as a Mixed Integer Linear Programming problem. The unimodularity of the constraint matrix of each subproblem $SP_h^v(\overline{x})$ and $SP_h^v(\underline{x})$ allows the exploitation of linear duality and the explicit statement of the optimality conditions of the inner objective function by linear constraints.

3 Computational results

We solved the proposed model using Cplex 12.1 and performed all testing on a AMD Athlon (tm) 64x2 Dual Core Processor 4200+ (CPU MHz 2211.186). Basically, we built our instances on the same data set described in [3], i.e.: an undi-

rected graph with $|N| = 105$ nodes and $|A| = 134$ arcs, abstraction of the road network of Ravenna (Italy), and the same cost function and risk functions (*on-arc*, *around-arc* and *aggregate*); $|V| = 35$ carriers. On this network, for each kind of risk measure, we generated 130 instances of the GLP considering five sizes of set N^g (10%, 20%, 30%, 50%, and 100% of N) and for each size below 100%, three independent random samples of N^g have been generated with uniform distribution: for each one of these thirteen combinations, k varies in $1, \ldots, 10$, thus yielding 130 instances. Considering the risk reduction of the GLP solution with respect to the unregulated scenario, thanks to variables γ_v, we can guarantee that it is never negative and it increases with k. The value of k beyond which the risk reduction can be considered negligible is said *stable*. We will present results showing that in our testbed stability occurs within $k = 4$ for all the three risk measures. For this reason we analyze more in depth the case $k = 4$. Let $G^\star$, $R^\star$ and $C^\star$ denote respectively the optimal solution of GLP (i.e. the gateway path set), the min risk path set (over-regulated scenario) and the min cost path set (unregulated scenario); let then $c(P)$ and $r(P)$ denote the total cost and the total risk of path set P. The quantity $(r(G^\star) - r(R^\star))/(r(C^\star) - r(R^\star)) \cdot 100$ yields a solution quality index since it measures the increase in risk with respect to the over-regulated scenario. For a percentage of candidate gateways fixed to 30% and for $k = 4$, the average of such index is 34.85 for the *on-arc* risk, 4.64 for the *around-arc* risk and 14.30 for the *aggregate* risk. These values are very close to those obtained when $N^g = N$. We can conclude that the use of gateways is an effective strategy for mitigating transport risk, especially for the *around-arc* and *aggregate* risk measures; good levels of risk reduction are already obtained with a low percentage of candidate gateways (30%), provided that they are chosen appropriately, and opening few gateways ($k = 4$).

References

[1] Lombard K, Church RL (1993) The Gateway Shortest Path Problem: Generating Alternative Routes for a Corridor Routing Problem. Geographical Systems 1:25–45.

[2] Kara BY, Verter V (2004) Designing a Road Network for Hazardous Materials Transportation. Transportation Science 38(2):188–196.

[3] Erkut E, Gzara F (2008) Solving the hazmat transport network design problem. Computers and Operations Research 35:2234–2247.

[4] Bruglieri M, Maja R, Marchionni G, Rainoldi G (2008) Safety in hazardous material road transportation: state of the art and emerging problems. In: C. Bersani et al. (Eds), Adv. techn. and method. for risk management, IOS Press:88-129.

[5] Ciccarelli D, Colorni A, Lué A (2007) A GIS-Based Multi-Objective Travel Planner for Hazardous Material Transport in the Urban Area of Milan. In: C. Bersani et als (Eds), Adv. techn. and method. for risk management, IOS Press:217-236.

[6] Carotenuto P, Giordani S, Ricciardelli S (2007) Finding minimum and equitable risk routes for hazmat shipments. Computers and Operations Research 34(5):1304–1327.
[7] Marcotte P, Mercier A, Savard G, Verter V (2009) Toll policies for mitigating hazardous materials transport risk. Transportation Science 43(2):228–243.

Oriented L$(2,1)$-Labeling of Planar Graphs (Extended Abstract)

T. Calamoneri,, B. Sinaimeri

Department of Computer Science, "Sapienza" University of Rome, Via Salaria 113, 00198 Roma, Italy
{calamo,sinaimeri}@di.uniroma1.it

Abstract

In this paper we study the $L(2,1)$-labeling problem on oriented planar graphs with particular attention to the subclasses of oriented prisms, Halin and cacti.

Key words: $L(2,1)$-labeling, oriented graph coloring, digraphs, prisms, Halin graphs, cacti.

1 Introduction and preliminaries

The $L(2,1)$-*labeling* of a graph G is a function l from the vertex set $V(G)$ to the set of all nonnegative integers such that :
– $|l(x) - l(y)| \geq 2$ if x and y are adjacent in G, and
– $|l(x) - l(y)| \geq 1$ if x and y are at distance 2 in G and the aim is to minimize the maximum used color over all the labellings. Such minimum value is called $\lambda(G)$. This problem has been introduced first by Griggs and Yeh [5] as a variation of the frequency assignment problem of wireless networks. A natural extension, introduced in [2], is the $L(2,1)$-labeling on *directed graphs*. The definition is the same as in the non directed case but the distance between vertices x and y is defined as the length of the shortest directed path from x to y. In agreement with the non directed case, the $L(2,1)$-labeling number $\underline{\lambda}(D)$ of a digraph D, is the minimum over all the $L(2,1)$-labellings l of D of $\max\{l(v) : v \in V(D)\}$. By extension, for a class $\mathcal{C}$ of digraphs, we denote by $\underline{\lambda}(\mathcal{C})$ the maximum $\underline{\lambda}(D)$ over all $D \in \mathcal{C}$. Here we will consider the $L(2,1)$-labeling problem only on digraphs that are orientations of finite simple graphs (i.e. graphs without loops or multiple edges). A related concept to the $L(2,1)$-labeling of oriented graphs is the *oriented chromatic number*. Given an oriented graph $\underline{G}$, its oriented chromatic number, $\underline{\chi}(\underline{G})$, is the smallest integer κ such that there is a coloring $f : V \to \{1, \ldots, \kappa\}$ such that $f(u) \neq f(v)$ if $(u,v) \in A$ and for any two arcs (u,v) and (x,y), if u and y have the same color then the colors assigned to v and x cannot be the same. The following is proved in [4]:

Proposition 1.1 *If $\underline{G}$ is an orientation of a graph G, then*

$$2\Big(\chi(G)-1\Big) \leq \underline{\lambda}(\underline{G}) \leq 2\Big(\underline{\chi}(\underline{G})-1\Big).$$

Note that the oriented chromatic number of a digraph and the chromatic number of its underlying graph can be far from each other. However in the case of oriented planar graphs this result turns out to be quite useful. Indeed, combining Proposition 1.1 with the results in [6,7] bounding the oriented chromatic number of a planar graph, we obtain the following:

Proposition 1.2 *Given an oriented planar graph $\underline{G}$ with girth (i.e. the length of a shortest cycle in G) g, it holds:*

- *If $g \geq 16$ then $\underline{\lambda}(\underline{G}) \leq 8$.*
- *If $g \geq 11$ then $\underline{\lambda}(\underline{G}) \leq 12$.*
- *If $g \geq 7$ then $\underline{\lambda}(\underline{G}) \leq 22$.*
- *If $g \geq 6$ then $\underline{\lambda}(\underline{G}) \leq 62$.*
- *otherwise $\underline{\lambda}(\underline{G}) \leq 158$.*

2 Subclasses of oriented planar graphs

Here we list some results concerning $\underline{\lambda}$ for some particular subclasses of planar graphs. Due to space limitations almost all the proofs are omitted.

A ***planar prism graph*** Pr_n is a graph isomorphic to $C_n \times P_2$, where C_n is a cycle on n vertices and P_2 a path of length 2. The $L(2,1)$-labeling number of an unoriented prisms is well-known. The following theorem proves that the same result holds in the oriented case.

Theorem 2.1 *Let $\mathcal{P}_n$ be the set of all the orientations of the planar prism graph Pr_n, $\underline{\lambda}(\mathcal{P}_n) = 5$ if $n \equiv 0 \mod 3$ and $\underline{\lambda}(\mathcal{P}_n) = 6$ otherwise.*

A ***Halin graph*** H is a planar graph resulting from the union of a tree T with at least four vertices and with no vertex of degree 2, and a cycle C connecting all the leaves of T cyclic order defined by a plane embedding of T.

Theorem 2.2 *Let $\mathcal{H}$ be the set of all the oriented Halin graphs, $8 \leq \underline{\lambda}(\mathcal{H}) \leq 11$.*

Proof. We sketch the proof of the upper bound. Let $\underline{H}$ be an oriented Halin graph constituted by the oriented tree $\underline{T}$ and the oriented cycle $\underline{C}$. For a vertex v let $N^+(v)$ ($N^-(v)$) be the set of out (in-) neighbors of v and let $L^+(v)$ ($L^-(v)$) be the sets of labels used in $N^+(v)$ ($N^-(v)$). Following the algorithm in [2], we label the inner vertices of T, ignoring the arcs of the cycle $\underline{C}$, and using labels $0, 4, 8$. We will label the vertices on C using labels in $\{0, \ldots, 11\} \setminus \{0, 4, 8\}$. Thus, there is no conflict between the label of any vertex on the cycle and the label of a vertex of the tree being at distance two from it. We define an *Out-segment* (*In-segment*) of a vertex v as a maximal consecutive sequence of vertices in the cycle such that all its vertices belong to the set $N^+(v)$ ($N^-(v)$). Let us now consider the following six sets of labels:

$I_0 = \{2,5,9\}, I_4 = \{1,6,9\}, I_8 = \{1,5,10\}, O_0 = \{3,6,10\}, O_4 = \{2,7,10\}$

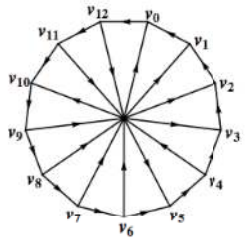


Fig. 1. An oriented Halin graph for which $\underline{\lambda} = 8$.

and $O_8 = \{2, 6, 11\}$. We will use labels from set I_x (respectively O_x) to label the In-segments (respectively Out-segments) of a vertex labeled with x according to the labeling of the tree. The following properties hold:

Property 3 *For every $x \in \{0, 4, 8\}$ it holds $I_x \cap O_x = \emptyset$.*

Note that this property is necessary as in order to guarantee a feasible labeling it must be $L^+(v) \cap L^-(v) = \emptyset$ for any vertex v.

Property 4 *The intersection of any pair of label sets not corresponding to the same label consists of exactly one label.*

Property 5 *For any label $x \in A$ the intersection of the set $\{x - 1, x, x + 1\}$ with any of the label sets consists of exactly one label.*

Let S, be a segment of maximum length. It is not restrictive to assume that S is an In-segment of a vertex labeled 0. Let $P_1, \ldots, P_r = S$ be the sequence of segments in clockwise order starting from the segment that follows S and ending with S. From Properties 4 and 5, once the segments $P_1, \ldots, P_{i-1}$ have been labeled, it is always possible to label the segment P_i corresponding to a vertex labeled x, using labels from I_x or O_x. Finally, we label S using labels from $B = I_0 \cup \{7, 11\}$. (Observe that $B \cap O_0 = \emptyset$ so Property 3 still holds). Suppose $|S| \geq 3$ (the other cases can be handled easily) and consider the following sequence of vertices in a clockwise order: $\ldots, w, u, s_1, s_2 \ldots, s_m, v, k \ldots$, where $s_1, s_2 \ldots, s_m$ is the sequence of vertices of S. Note that in order to label S there are no other vertex interferences to be considered. Using the Properties 4 and 5 it can be proved that once the vertices v, k, w and u are labeled, it is always possible to conclude the labeling.

Concerning the lower bound, we proved that the oriented Halin graph in Fig. 1 has $\underline{\lambda} = 8$. ∎

The next theorem shows that this lower bound is tight for oriented wheels, i.e. Halin graphs whose tree is a star.

Theorem 2.3 *Let $\mathcal{W}$ be the set of all oriented wheels. It holds $\underline{\lambda}(\mathcal{W}) = 8$.*

A ***cactus*** is a connected graph in which any two simple cycles have at most one vertex in common.

Theorem 2.4 *Let $\mathcal{Y}$ be the set of all the oriented cacti then $6 \leq \underline{\lambda}(\mathcal{Y}) \leq 8$.*

The proof follows by showing that there is a homomorphism from any cactus $\underline{Y}$ to a tournament (i.e. an orientation of a complete graph) on 5 vertices.

3 Concluding remarks and open problems

In this extended abstract we approached the problem of determining $\underline{\lambda}$ for oriented planar graphs and for some particular subclasses we provided nearly tight bounds for $\underline{\lambda}$. It is worth to note that in the unoriented case there is a strong relationship between λ and the graph's maximum degree Δ, and thus it efficiently expresses the value of λ. However the $L(2,1)$-labeling problem on oriented graphs presents different issues with respect to the unoriented case and Δ is not an appropriate parameter anymore. Due to the fact that very few classes of oriented graphs have been investigated, it is not possible yet to identify the most natural graph parameter that efficiently expresses the value of $\underline{\lambda}$ for arbitrary oriented graphs. However, for the class of oriented planar graphs it seems reasonable to suggest that the girth of the underlying graph is in some relation with $\underline{\lambda}$. This is suggested by Proposition 1.2 and the following Conjecture in [1]:

Conjecture 2 *Every oriented planar graph D whose underlying graph has girth $g \geq 5$ has $\underline{\lambda}(D) \leq 5$.*

This relation cannot be as strong as in the unoriented case between λ and Δ. Thus, investigating in this direction is an interesting open problem.

References

[1] T. Calamoneri. The $L(2,1)$-Labeling Problem on Oriented Regular Grids, *The Computer Journal* , (to appear), 2011.

[2] G. J. Chang and S.-C. Liaw. The $L(2,1)$-labeling problem on ditrees. *Ars Combinatoria*, 66: 23–31, 2003.

[3] G.J. Chang, J.-J. Chen, D. Kuo and S.-C. Liaw. Distance-two labelings of digraphs. *Discrete Applied Mathematics*, 155: 1007–1013, 2007.

[4] Y.-T. Chen, M.-L. Chia and D. Kuo. $L(p,q)$-labelings of digraphs. *Discrete Applied Mathematics*,157: 1750–1759, 2009.

[5] J.R. Griggs and R.K. Yeh. Labeling graphs with a Condition at Distance 2. *SIAM J. Disc. Math*, 5:586–595, 1992.

[6] J. Nesetril, A. Raspaud and E.Sopena. Colorings and Girth of Oriented Planar Graphs *Discrete Math.*, 165-166: 519–530, 1997.

[7] A. Raspaud and E. Sopena. Good and semi-strong colorings of oriented planar graphs. *Inform. Process. Lett.*, 51: 171–174, 1994.

Effective drawing of proportional symbol maps using GRASP

Rafael G. Cano, Guilherme Kunigami, Cid C. de Souza, Pedro J. de Rezende

Institute of Computing, State University of Campinas, Campinas, SP, Brazil 13084-852

Key words: GRASP, computational geometry, cartography, symbol maps

1 Introduction

Proportional symbol maps are a cartographic tool that employs symbols to represent data associated with specific locations, e.g., the magnitudes of earthquakes and the population of cities. Each symbol is drawn at a location and its size is proportional to the numerical data collected at that point on the map. The symbols considered here are opaque disks, although other geometric shapes are also common, such as triangles and squares. When two or more disks overlap, part of their boundaries may not be visible. When a disk has little or no visible boundary, it is difficult to gauge its size. Therefore, the order in which the disks are drawn affects the visual quality of a map.

Let S be a set of n disks and $\mathcal{A}$ be the arrangement formed by the boundaries of the disks in S. An intersection of the boundaries of two or more disks defines a *vertex* of $\mathcal{A}$. We say that an *arc* is a portion of the boundary of a disk that connects two vertices and contains no other vertices. A *drawing* of S is a subset of the arcs and vertices of $\mathcal{A}$ that is drawn on top of the filled interiors of the disks in S. We focus on *stacking drawings*, i.e., a drawing that corresponds to the disks being stacked up, in sequence, starting from the one at the bottom of the stack. To measure the quality of a drawing, two values can be considered: the minimum visible boundary length of any disk and the total visible boundary length over all disks. The *Max-Min* and the *Max-Total* problems consist in maximizing the former and the latter values, respectively.

⋆ Supported by FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo – Grants #2009/17044-5, #2007/52015-0, CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico – Grants #830510/1999-0, #301732/2007-8, 472504/2007-0, 473867/2010-9, 483177/2009-1 and a Grant from FAEPEX/UNICAMP.

Cabello et al. [1] present a greedy algorithm to solve the Max-Min problem in $O(n^2 \log n)$ time. Kunigami et al. [2] propose an integer linear programming formulation for the Max-Total problem, as well as several families of facet-defining inequalities. In addition, they introduce decomposition techniques that split the disks into smaller components which can be solved independently. The computational complexity of the Max-Total problem for stacking drawings remains open.

Our contribution. In this work, we propose a sophisticated heuristic based on GRASP [3] with path-relinking for the Max-Total problem for stacking drawings. Our heuristic includes most of the advanced techniques described in the literature for this procedure. We tested both sequential and parallel implementations on benchmark instances and the comparison against optimal solutions confirms the high quality of the heuristic. To the best of our knowledge, this is the first time a metaheuristic is presented for this problem. The heuristic is described in Section 2 and some computational results are shown in Section 3.

2 GRASP heuristic

Initial solutions are generated using a randomized version of the algorithm presented in [1] for the Max-Min problem. The experimental results reported in that work show that this algorithm performs well for the Max-Total problem and, for this reason, it was chosen for the GRASP construction phase. Initially, the stack of disks is empty. Given a disk i, denote by b_i the length of its visible boundary if it were placed above the disks that have already been stacked and below the others. Each iteration computes b_i for each disk i not yet on the stack. The disk with maximum such value is placed on top of the current stack and this procedure is repeated until all disks have been stacked.

For the randomized version, instead of choosing the disk i with maximum b_i, we create a restricted candidate list (RCL) based on a parameter $\alpha \in [0, 1]$. Let b^{min} and b^{max} be the minimum and the maximum b_i values over all disks i not yet on the stack, respectively. A disk i is inserted into the RCL whenever $b_i \geq b^{min} + \alpha(b^{max} - b^{min})$. We experiment with strategies to select a value for α which not only yields solutions of high quality but also promotes their diversity (see e.g. [3]). After the creation of the RCL, one of the disks in it is randomly selected and stacked on top of the disks already chosen.

Two neighborhoods are considered for the local search, which we denote by *insertion* and *swap* neighborhoods. An insertion move removes a disk from its current level and inserts it in another position. A swap move exchanges the position of two disks. These neighborhoods are also used to implement a Variable Neighborhood Descent (VND). A *first improvement* strategy is used with both of them because it takes less time than a *best improvement* strategy and the quality of the results is very similar. To search the insertion neighborhood quickly, we use a variation of a segment tree to manipulate geometric objects in an efficient way (see [1] for more details). We also developed fast procedures to evaluate the change produced

by each swap move.

Path-relinking is applied to every local optimum generated by the local search. We create an *elite set* to store the best N_{elite} solutions found during the execution of the heuristic. After the local search, a solution from the elite set is randomly chosen and is relinked to the local optimum, i.e., a sequence of disk moves is executed to transform one solution into the other in such a way that the distance between the solutions always decreases. Afterwards, local search is applied to the best intermediate solution found. This improved solution is a candidate for the elite set and its inclusion follows the rules described in [3].

In order to measure the distance between two solutions, we consider three metrics, namely, the minimum number of insertions (*insertion metric*) and the minimum number of swaps (*swap metric*) necessary to transform one solution into the other, and the number of *inversions* between the solutions (*inversion metric*). Given two disks i and j, we say that an inversion occurs when $level(i) > level(j)$ in one of the solutions and $level(i) < level(j)$ in the other. We execute insertion moves with the insertion and the inversion metric and swap moves with the swap metric.

Evolutionary path-relinking is used after the end of the GRASP iterations by applying path-relinking to pairs of elite solutions. We also implement a variation of this method in which each elite solution s is relinked to the solution s' obtained by reversing the order of the disks in s. Thus, the distance between s and s' is maximum under the inversion metric.

We also implement a parallel version of our heuristic using MPI as follows. Let N_i and N_p be the number of iterations performed by the heuristic and the number of processors, respectively. Each processor is assigned an initial subset of N_i/N_p iterations but whenever a processor's set is completed, it is free to take over some from other busy processors. After every 10 iterations, each processor requests all others to send their elite sets and the solutions received are candidates for its own elite set. To avoid idle processors, non-blocking communication is used. Evolutionary path-relinking is executed independently by each processor. Since the processors may have very similar elite sets, each one stores a local elite set that contains only solutions found by itself. Evolutionary path-relinking is applied only to these local elite sets.

3 Computational results

Preliminary experiments show that setting $\alpha = 0.4$ produces solutions of the highest total visible boundary values. Furthermore, using the insertion neighborhood leads to faster runs and finds better solutions than the swap neighborhood. Therefore, our VND searches the insertion neighborhood first. The experiments also indicate that path-relinking finds the best solutions with the inversion metric and an elite set of size $N_{elite} = 10$. We run 1000 GRASP iterations for each instance and the parallel implementation uses 4 processors.

With these configurations, we performed experiments with 28 real-world instances

generated from data on the population of cities. We apply the decomposition techniques from [2] and compare the results obtained by our heuristic with the ones found by the exact method described in that work. The experiments were run on an Intel Core 2 Quad 2.83GHz CPU with 8GB RAM. The sequential and the parallel GRASP heuristics found optimal solutions for 15 and 14 instances, respectively. Moreover, when no optimal solution was found, the average relative gaps between the value of the best heuristic solution and the optimal was 0.07% for the sequential and 0.04% for the parallel GRASP. On average, the sequential GRASP ran 110.25 times faster than the exact method and the parallel implementation was 2.45 times faster than the sequential. A linear speedup is not achieved because the initial decomposition is executed sequentially and the evolutionary path-relinking runs independently on each processor. Some results are shown in Table 5. The values are calculated ignoring arcs that are always visible in every solution. Optimal values found by the GRASP are highlighted in bold.

Instances	n	Sequential GRASP		Parallel GRASP		Exact	
		Value	Time	Value	Time	Value	Time
China	141	**2409.15**	20.4	**2409.15**	6.7	2409.15	2882
Brazil	150	**2553.52**	11.3	**2553.52**	5.1	2553.52	982
Russia	150	1565.51	10.5	1565.51	4.0	1566.56	1003
Switzerland	186	**3573.34**	14.0	3571.74	5.9	3573.34	2453
Netherlands	367	4719.09	27.8	4720.41	10.3	4720.45	4175

Table 5. Results for real-world instances. Times are given in seconds.

References

[1] S. Cabello, H. Haverkort, M. van Kreveld, and B. Speckmann. Algorithmic aspects of proportional symbol maps. *Algorithmica*, 58(3):543–565, 2010.

[2] G. Kunigami, P. J. de Rezende, C. C. de Souza, and T. Yunes. Optimizing the layout of proportional symbol maps. *Optimization Online*, 2010.

[3] M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In F. Glover and G. A. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*, chapter 8, pages 219–249. Springer, second edition, 2009.

Sensor Network Localization Using Compressed Extended Kalman Filter

M. Carli, [a] F. Pascucci [b]

[a] *Dip. Elettronica Applicata, Università degli Studi "Roma Tre"*
Via della Vasca Navale, 84, 00146 Roma – IT
carli@uniroma3.it

[b] *Dip. Informatica e Automazione, Università degli Studi "Roma Tre"*
Via della Vasca Navale, 79, 00146 Roma – IT
pascucci@uniroma3.it

Key words: Wireless Sensor Networks, Distributed Algorithms, Disk Graph

1 Introduction

Recent advances in wireless communication and electronics have enabled the development of low cost, low power, and multifunctional sensors that are small in size and communicate in short distances. Smart sensors, connected through wireless links and deployed in large numbers, promise of many new applications in the area of monitoring and controlling. In these applications it is necessary to accurately locate the nodes with respect to a global coordinates framework, in order to correlate sensor readings with physical locations. Since manually recording and entering the position of each sensor is impractical for very large networks, self localization capability is highly desirable characteristic of Wireless Sensor Networks (WSNs). Nowadays, the most simple, off-the-shelf mechanism for determining the sensor node location is to use Global Positioning System (GPS). However, it is unsuitable for low-cost, ad-hoc sensor networks, since it is based on extensive infrastructure, satellites, and it cannot be used indoor. For these reasons, localization in WSNs has drawn considerable attention and several localization protocols have been proposed in literature. A complete discussion about issues in WSN localization can be found in [1]. Here we focus our attention to distributed localization algorithms, based on ranging technique. The localization problem is mapped into a stochastic estimation problem for systems with uncertainty. The proposed procedure exploits the benefits of Cluster Based Routing Protocol (CBRP)[2] and of Compressed Extended Kalman Filter (CEKF)[3]. The overall system is self-organizing (i.e., it does not depend on global infrastructure), robust (i.e., it is tolerant to node failures and range errors), and energy efficient (i.e., low computational complexity and communication overhead).

2 Problem setting

The localization scenario here considered refers to a set of sensors deployed in the plane (i.e., they have coordinates in $\mathbb{R}^2$) assuming that few nodes, called anchors, are equipped with absolute positioning devices. Since only nodes within a communication range can measure their relative distances, the localization problem can be regarded as a graph realization problem over a quasi unit disk graph, which is known to be NP-HARD [4]. Many strategies have been proposed in literature as [5], [6]. Most of them relies on centralized approach and do not take into account both computational load and communication overhead. Moreover they lack of robustness, as do not consider node failures and outliers.

In this work the localization problem is formulated as a stochastic estimation problem for systems with uncertainty described by the following equations:

$$\begin{aligned} x_k &= f(x_{k-1}, w_k) = x_{k-1} + w_k \\ z_k &= h(x_k, v_k) \end{aligned} \tag{1}$$

where $x_k = [p^x_{1,k}, p^y_{1,k}, \ldots, p^x_{N,k}, p^y_{N,k}]$ is a stochastic variable representing the positions of nodes in a 2D geometric coordinates, w_k and v_k are noises affecting the system (i.e., uncertainties), and $h(\cdot)$ characterizes inter-nodes ranging measurements, i.e. $z^{i,j}_k = \sqrt{(p^x_{i,k} - p^x_{j,k})^2 + (p^y_{i,k} - p^y_{j,k})^2}$. In a Maximum Likelihood framework a prediction-correction scheme of Bayes filtering can be exploited to recursively refine position estimate. Since the capability of the nodes are very limited, collaboration between nodes is required.

3 Localization Algorithm

The proposed localization algorithm is based on Extended Kalman Filter (EKF). As already known [8], due to the particular structure of the system, the computation of EKF for WSN localization can be fully distributed over the network. In this way, each node is able to estimate its own position and maintain information about the accuracy of the estimate of all nodes in the network. The main drawback of this approach is in the heavy communication overhead. Moreover it is unreliable for large scale network, since accuracy information grows with the number of the nodes and both, limited memory and low CPU power, are not able to cope with this large amount of data. Here, to solve these problems a *divide et impera* approach is adopted, by decomposing the network into clusters. The localization problem is solved in a coarse to fine two step procedure. During the first step, a cluster based routing protocol is used to discover the topology of the network-graph and provide a rough estimate for the position of the nodes using bounding-box techniques [7]. Then the localization procedure, CEKF, is applied to each cluster. In the following some details of clustering and CEKF are reported.

3.1 Cluster Based Routing Protocol

In the proposed framework, we adopt the hierarchical routing algorithm for ad-hoc networks known as CBRP. It uses clustering's structure for decreasing routing control overhead and improving the networks scalability. Clustering transforms a physical network into a virtual network of interconnected clusters or groups of nodes. Basically it exploits both proactive (each node maintains an updated lists of linked nodes and their routes by periodically distributing routing tables throughout the network) and reactive (each node discover the network path by flooding the network with Route Request packets) routing behavior. Each node acts as a proactive or reactive ones according to its hierarchic level. The 2 steps procedure is based on a setup phase in which the nodes are clustered in 2-hop-diameter interconnected substructures (clusters) followed by a Cluster Head (CH) selection for each cluster. The CH coordinates and maintains the cluster membership information. The routing is therefore established with proactively created routes among the CHs (by dynamically sharing and using the cluster membership information); the remaining nodes are served by reactive flooding on the lower levels. Each node belonging to a cluster holds a Neighbor table wherein the information about the other neighbor nodes is stored. In the following the main aspects of CBRP are reported for sake of clarity. Each node can be a CH if it has the necessary functionality, such as processing, localization capability, or transmission power. Following the cluster formation phase and the CH election one, a gateway discovery procedure is performed. A node acts as a gateway node of a cluster if, from the Cluster Adjacency table, it has link to a node from another cluster. This table is dynamically created in each node during the CH election.

3.2 Compressed Extended Kalman Filter

After the network topology discover procedure is completed, each cluster runs CEKF to refine localization. The process starts in the clusters containing the bigger number of anchors, since their localization will be more reliable. Next, as in a flooding procedure, the other clusters characterized by a lower number of anchors start their localization procedure taking advantage from the high accuracy of the just computed position of neighboring clusters. CEKF relies on the well known equation of EKF, here applied to WSN localization.

During the prediction step a decomposition strategy is easy to find, as the state transition map is represented by an identity matrix. More complex is the decomposition of the correction step, wherein the Kalman gain has to be calculated as well as a full update of the covariance matrix $P_{k|k}$. To achieve the desired decomposition the covariance matrix can be partitioned into M sub matrices, having dimension 2×2, i.e. the cross correlation matrices between nodes. Since a rough estimation for node position in the cluster is available at $k = 0$, it is possible to consider only one measure between two nodes $i - j$ at each instant k. Under these assumptions it is easy to show that a nodes l needs to store information about its own cross correlation matrices and to receive few data from the net (S_k, J_k^h, i.e. the innovation covariance

and the Jacobian of the observation map) in order to perform a complete update.
To further reduce the computational complexity, here a sub-optimal solution is proposed by updating the states, without updating the cross covariance sub matrices of nodes in the cluster which are not neighbor of nodes $i - j$ involved in the measurement process. This partition generate a conservative estimate and is known in literature as CEKF.

4 Conclusion

In this contribution the problem of distributed localization algorithms, based on ranging technique, has been addressed. The localization problem is mapped into a stochastic estimation problem for systems with uncertainty by using Cluster Based Routing Protocol and Compressed Extended Kalman Filter. On the other end, the CEKF reduces the computational complexity of a localization procedure. Even if the sub-optimal solution is achieved, the localization accuracy is suitable for most WSNs application scenarios. Experimental test, performed on random graph having different degree of connection, show the effectiveness of the proposed approach. Future work is devoted to apply the localization procedure to scale-free and small world networks.

References

[1] G. Mao, B. Fidan, B.D.O. Anderson, *Wireless sensor network localization techniques,* Computer Networks, Vol 51–10, 2007, 2529-2553

[2] M. Jiang, J. Li, and Y.C. Tay, *Cluster based routing protocol*, Cluster Based Routing Protocol (CBRP) Functional Specification Internet Draft, http://tools.ietf.org/html/draft-ietf-manet-cbrp-spec, June 1999.

[3] J.E. Guivant and E.M. Nebot, *Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation,* in IEEE Trans. On Robotics And Automation, Vol. 17, No. 3, June 2001

[4] F. Kuhn, R. Wattenhofer, and A. Zollinger. *Ad hoc networks beyond unit disk graphs,* Wirel. Netw. 14, 5 (October 2008), 715-729.

[5] L. Doherty, K. S. J. Pister, and L. El Ghaoui, *Convex position estimation in wireless sensor networks,* Proceedings of IEEE Infocom, 2001.

[6] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, *Semidefinite programming based algorithms for sensor network localization,* ACM Trans. Sensor Netw. 2, 2, 188–220, 2006.

[7] S. Simic and S.S. Sastry, *Distributed Localization in Wireless AdHoc Networks,* technical report, Univ. of California, Berkeley, 2002.

[8] M. Di Rocco and F. Pascucci, *Sensor network localisation using distributed Extended Kalman filter,* IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, 2007, 1–6.

A Generalized Stochastic Knapsack Problem with Application in Call Admission Control

Marco Cello, Giorgio Gnecco, Mario Marchese,
Marcello Sanguineti

Department of Communications, Computer and Systems Science (DIST), University of Genoa - Via Opera Pia, 13 - 16145 Genova (Italy)
{marco.cello,
giorgio.gnecco,mario.marchese,marcello}@dist.unige.it

Key words: Stochastic Knapsack, Feasibility Region, Call Admission Control, Coordinate-Convex Policies.

1 Introduction

In the classical *knapsack problem*, a knapsack of capacity C is given, together with K classes of objects. For every $k = 1, \ldots, K$, each object of the class k has a size b_k and an associated reward r_k. The objects can be placed into the knapsack as long as the sum of their sizes does not exceed the capacity C. The problem consists in placing the objects inside the knapsack so as to maximize the total reward.

Among the various extensions to a stochastic framework available in the literature (see, e.g., [7,5,3]), we consider the *stochastic knapsack problem* proposed in [7]. In such a model, objects belonging to each class become available randomly, according to exponentially-distributed inter-arrivals times with means depending on the class and on the state of the knapsack. Each object has a sojourn time independent from the sojourn times of the other objects and described by a class-dependent distribution. If put into the knapsack, an object from class k generates revenue at a positive rate r_k. Let $n_k \geq 0$ denote the number of objects of class k that are currently inside the knapsack. Then one has the linear constraint

$$\sum_{k \in K} n_k b_k \leq C \,. \tag{1}$$

The problem consists in finding a policy that maximizes the average revenue, by accepting or rejecting the arriving objects in dependence of the current state of the knapsack.

The stochastic knapsack problem that we have just described has application, e.g., in *Call Admission Control* (*CAC*) for telecommunication networks. In such a context, the objects are requests of connections coming from K different classes of

users, each with a bandwidth requirement b_k, $k = 1, \ldots, K$, and a distribution for its duration. In CAC problems, often the constraint (1) arises as a linearization of the nonlinear constraint

$$\sum_{k \in K} \beta_k(n_k) \leq C\,, \tag{2}$$

where the $\beta_k(\cdot)$ are nonlinear nonnegative functions. The model in which the linear constraint (1) is replaced by the nonlinear one (2) is known in the literature as the *generalized stochastic knapsack problem* [1] [4]. In call admission control, the coefficients b_k of the linearized constraint are called *effective bandwidths* [6, Chapter 1].
The sets

$$\Omega_{FR} := \{(n_1, \ldots, n_K) \in \mathbb{N}_0^K : \sum_{k \in K} n_k b_k \leq C\}, \tag{3}$$

in the linear case, and

$$\Omega_{FR} := \{(n_1, \ldots, n_K) \in \mathbb{N}_0^K : \sum_{k \in K} \beta_k(n_k) \leq C\} \tag{4}$$

in the nonlinear case, are called *feasibility regions*. In the context of admission control they model subsets of the call space $\{(n_1, \ldots, n_K) \in \mathbb{N}_0^K\}$, where given *Quality of Service* (*QoS*) constraints are satisfied.
In general, finding optimal policies is a difficult combinatorial optimization task both for the stochastic knapsack problem [6, Chapter 4] and for the generalized stochastic one [1,2]. The a-priori knowledge of structural properties of the (unknown) optimal policies is useful to find the solutions or, at least, good suboptimal policies. For two classes of objects and the linear constraint (1), structural properties were derived in [7] for the optimal policies belonging to the family of *coordinate-convex policies*. Such properties restrict the K-tuple $(n_1, \ldots, n_K)$ to suitable subsets of the feasibility region Ω_{FR}. Some extensions to nonlinearly-constrained feasibility regions of the structural results obtained in [7] for linearly-constrained ones were derived in [2] and other structural results were obtained in [1].

2 Problem formulation

Let $\mathbf{n}$ denote the vector $(n_1, \ldots, n_k)$. For each class $k = 1, \ldots, K$, the inter-arrival time is exponentially distributed with mean value $1/\lambda_k(n_k)$. The sojourn times of the accepted objects are independent and identically distributed (i.i.d.) with mean values $1/\mu_k$, $k = 1, \ldots, K$. At the time of its arrival, each object is either accepted or rejected, according to a coordinate-convex *policy*, defined as follows [6, p. 116].

[1] Note that this is different from the "generalized stochastic knapsack" considered in [6, Chapter 3].

Definition 2.1 *A nonempty set $\Omega \subseteq \Omega_{FR} \subset \mathbb{N}_0^K$ is coordinate-convex iff it has the following property: for every $\mathbf{n} \in \Omega$ with $n_k > 0$ one has $\mathbf{n} - \mathbf{e}_k \in \Omega$, where $\mathbf{e}_k$ is a K-dimensional vector whose k-th component is 1 and the other ones are 0. The coordinate-convex policy associated with a coordinate-convex set Ω admits an arriving object iff after its insertion one has $\mathbf{n} \in \Omega$.*

Note that by (3) or (4), the set Ω_{FR} is itself coordinate-convex. As there is a one-to-one correspondence between coordinate-convex sets and coordinate-convex policies, in the following we use the symbol Ω to denote either a coordinate-convex set or a coordinate-convex policy.

The objective to be maximized in the set $\mathcal{P}(\Omega_{FR})$ of coordinate-convex subsets of Ω_{FR} is given by

$$J(\Omega) = \sum_{\mathbf{n} \in \Omega} (\mathbf{n} \cdot \mathbf{r}) P_\Omega(\mathbf{n}) \,, \tag{5}$$

where $\mathbf{r}$ denotes the vector $(r_1, \ldots, r_K)$ and $P_\Omega(\mathbf{n})$ is the steady-state probability that the current content of the knapsack is $\mathbf{n}$. As Ω is coordinate-convex, one can show that $P_\Omega(\mathbf{n})$ takes on the product-form expression

$$P_\Omega(\mathbf{n}) = \frac{\prod_{i=1}^K q_i(n_i)}{\sum_{\mathbf{n} \in \Omega} \prod_{i=1}^K q_i(n_i)} \,, \text{ where } q_i(n_i) := \frac{\prod_{j=0}^{n_i - 1} \lambda_i(j)}{n_i! \mu_i^{n_i}} \,. \tag{6}$$

Due to (6), in general the objective (5) is nonlinear. What makes the problem difficult is that, given any two coordinate-convex sets $\Omega_1, \Omega_2 \subseteq \Omega_{FR}$, in general the relationship $\Omega_1 \subseteq \Omega_2$ does not imply $J(\Omega_1) \leq J(\Omega_2)$.

3 Contributions

For generalized stochastic knapsack problems with two classes of objects (which model CAC with two classes of users), in [1] we derived for the optimal coordinate-convex policies structural properties that do not depend on the revenue ratio $R := r_2/r_1$. In [2], instead, we obtained properties that do depend on it. In the present work, we develop the investigation in the following directions.

(i) We analyze the optimal choices for some parameters used by a criterion proposed in [1] to improve certain suboptimal coordinate-convex policies. The criterion is based on the removal or addition of rectangular subregions near suitably-defined corner points.

(ii) We propose a greedy algorithm of approximate solution, based on the optimal choice of the parameters in (i). The related simulations show an improvement of the objective over the numerical results derived in [1].

(iii) We address some relationships between general structural properties of the optimal coordinate-convex policies and the greedy algorithm in (ii). In particular, we extend [1, Theorem III.6] by proving that the coordinate-convex set associated with an optimal coordinate-convex policy has a nonempty intersection with the upper boundary $(\partial\Omega_{FR})^+$ of the feasibility region Ω_{FR} (see

Fig. 1), independently of the number of its corner points.

(iv) We exploit another general structural property of the optimal coordinate-convex policies to initialize the greedy algorithm in (ii).

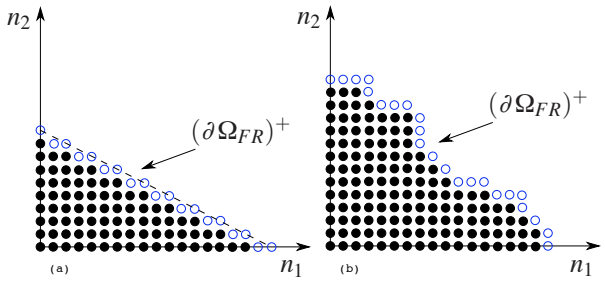


Fig. 1. The upper boundary $(\partial\Omega_{FR})^+$ of a feasibility region Ω_{FR} with two classes of objects in the case of (a) a linearly-constrained Ω_{FR} (stochastic knapsack) and (b) a nonlinearly-constrained Ω_{FR} (generalized stochastic knapsack).

References

[1] M. Cello, G. Gnecco, M. Marchese, and M. Sanguineti. CAC with nonlinearly-constrained feasibility regions. *IEEE Communications Letters*, 15(4):467–469, 2011.

[2] M. Cello, G. Gnecco, M. Marchese, and M. Sanguineti. Structural properties of optimal coordinate-convex policies for CAC with nonlinearly-constrained feasibility regions. In *Proceedings of IEEE INFOCOM (Mini-Conference)*, pages 466–470, 2011.

[3] B. C. Dean, M. X. Goemans, and J. Vondrak. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

[4] T. Javidi and D Teneketzis. An approach to connection admission control in single-hop multiservice wireless networks with QoS requirements. *IEEE Transactions on Vehicular Technology*, 52(4):1110–1124, 2003.

[5] A. J. Kleywegt and J. D. Papastavrou. The dynamic and stochastic knapsack problem with random sized items. *Operations Research*, 49(1):26–41, 2001.

[6] K. W. Ross. *Multiservice Loss Models for Broadband Telecommunication Networks*. Springer, New York, 1995.

[7] K.W. Ross and D.H.K. Tsang. The stochastic knapsack problem. *IEEE Transactions on Communications*, 37(7):740–747, 1989.

A partial characterization by forbidden subgraphs of edge path graphs

Márcia R. Cerioli, [a 1] Hugo Nobrega, [a 2] Petrucio Viana [?? 3]

[a] *COPPE/Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Caixa Postal 68511, 21941-972, Rio de Janeiro, Brazil*
Instituto de Matemática, Universidade Federal do Rio de Janeiro, Caixa Postal 68530, 21941-909, Rio de Janeiro, Brazil
{cerioli,petrucio}@cos.ufrj.br

Key words: edge path graphs, forbidden subgraphs, critical graphs

1 Introduction

Path graphs is the generic name given to the several classes of intersection graphs of paths in trees. An extensive study of the structural properties of these classes was done by C. Monma and V. Wei in [4] where, considering trees which are directed (D) or undirected (U), paths which are given by their vertex sets (V) or edge sets (E), families of paths which have the Helly property (H) or not, and trees which are rooted (R) or not, the classes UV, DV, RDV, UE, UEH, and DE were investigated.

Since every path graph is defined by an intersection property, each one can be characterized by forbidden induced subgraphs (*forbidden subgraphs*, for short). For the classes UV [3], DV [5], and a subclass of UEH [1], the complete families of forbidden subgraphs have already been found, but for no other path graph class is this the case.

Of particular interest is the class UE, also known as *edge path graphs*, which was introduced independently by M. Syslo [6] and M. Golumbic and R. Jamison [2]. This interest is justified by this being one of the relatively few examples of graph classes which are both widely studied and are NP-complete to recognize [7]. Therefore, it is somewhat natural to expect that the forbidden subgraphs for this class will be especially hard to find and describe completely.

In this work, we show how a large family of forbidden subgraphs for UE graphs can be obtained using a construction from the well-known class of 4-critical graphs.

[1] Research supported in part by CAPES, CNPq, and FAPERJ.
[2] Research supported by FAPERJ (Bolsa Mestrado Nota 10).
[3] Research supported in part by CNPq and FAPERJ.

This also implies that this family of forbidden subgraphs for UE is as hard to recognize as 4-critical graphs.

2 Preliminaries

The main tool which has been used in characterizing path graph classes by forbidden subgraphs is the *Separator Theorem*, due to Monma and Wei [4]. This theorem is based on the following concepts. If C is a maximal clique (maxclique, for short) whose removal disconnects G, then C *separates* G, and if V_i is the vertex set of a connected component of $G \setminus C$, then $G_i = G[V_i \cup C]$ is a *separated graph* of G by C. If $v \in C$ has a neighbor $v_i \in V_i$, then G_i is a *neighboring subgraph* of v.

Now, two separated graphs G_i and G_j of G by C are *antipodal* when there exist maxcliques C_i, C_i', C_i'' of G_i and C_j, C_j', C_j'' of G_j, such that:

(1) $C_i \cap C_j' \neq \emptyset$;
(2) $C_i \cap C \not\supseteq C_j'' \cap C$;
(3) $C_j \cap C_i' \neq \emptyset$; and
(4) $C_j \cap C \not\supseteq C_i'' \cap C$.

The *antipodal graph* of G by C, denoted by $\mathscr{A}(G, C)$, is the graph which has the separated graphs of G by C as vertices, and such that two vertices G_i and G_j are adjacent iff G_i and G_j are antipodal.

Note that, if G is a split graph which is separated by its central maxclique C, then separated graphs $G_i = G[\{v_i\} \cup C]$ and $G_j = G[\{v_j\} \cup C]$ are antipodal iff $N_G(v_i)$ and $N_G(v_j)$ have nonempty intersection, but are incomparable.

Although the class UE itself is not characterized by Monma and Wei's Separator Theorem, the subclass of UE graphs which are also chordal, denoted by UEC, is:

Theorem 13 (Separator Theorem for UEC) *Let G be separated by a maxclique C. Then G is a* UEC *graph iff each separated graph G_i is a* UEC *graph, and $\mathscr{A}(G, C)$ has a 3-coloring in which the set of neighboring subgraphs of each $v \in C$ is 2-colored.*

For our main result, we shall also need the following concept. A graph G is *k-critical* when $\chi(G) = k$ and, for each proper subgraph H of G, we have $\chi(H) < k$. It is well known that if G is k-critical, then $\delta(G) \geq k - 1$, where $\delta(G)$ is the minimum degree of G.

The construction

Given G, construct the graph $\mathsf{constr}(G)$ by first subdividing each edge e of G by a new vertex also denoted e, and then transforming the set of new vertices used in these subdivisions into a clique C_G. It is immediate to see that $\mathsf{constr}(G)$ is a split graph. What is more important, and also not hard to see, is that if $\delta(G) \geq 2$, then C_G is the only separating maxclique of $\mathsf{constr}(G)$, and $\mathscr{A}(\mathsf{constr}(G), C_G)$ is isomorphic to G.

3 Main results

Let $\mathcal{C}_4$ be the set of all graphs which are (isomorphic to) $\mathsf{constr}(G)$ for some 4-critical graph G.

Theorem 14 *$\mathcal{C}_4$ is a family of forbidden subgraphs for* UEC.

Proof (sketch) [Sketch of proof] Let H be a 4-critical graph, and $G = \mathsf{constr}(H)$. To see that G is a forbidden subgraph for UEC, note that since $\mathscr{A}(G, C_H) = H$ is not 3-colorable, we immediately have that G is not a UEC graph. However, since G is a split graph, each separated graph of G by C_H consists of at most two max-cliques, and it is easy to see that each one is a UEC graph — indeed, an edge path representation for such a graph is easily found.

Now let $G' = G \setminus v$. If $v \notin C_H$, then since H has more than 3 vertices, we have that C_H still separates G. It follows that $\mathscr{A}(G', C_H)$ is obtained from H by the removal of one vertex, and is 3-colorable since H is 4-critical. Since each $w \in C_H$ has at most two neighboring subgraphs, this set is 2-colored in any coloring of $\mathscr{A}(G', C_H)$. Furthermore, since each separated graph of G' by C_H is UEC, by the separator theorem we have that G' is a UEC graph.

On the other hand, if $v \in C_H$, then let $C' = C_H \setminus \{v\}$. By construction, v subdivides some edge v_1v_2 of H. Therefore, in G' the separated graphs $G[\{v_1\} \cup C']$ and $G[\{v_2\} \cup C']$ are not antipodal (since $N_{G'}(v_1)$ and $N_{G'}(v_2)$ have empty intersection), but since $\delta(H) \geq 3$, all other separated graphs of G' by C' are antipodal iff the corresponding separated graphs of G by C_H are antipodal. Hence, it follows that $\mathscr{A}(G', C')$ is obtained from H by the removal of the edge v_1v_2, and is also 3-colorable since H is 4-critical. By a similar reasoning as the one used above, we have that G' is a UEC graph. ∎

Since $\mathsf{constr}(G)$ is chordal for every G, we also have that $\mathcal{C}_4$ is a family of forbidden subgraphs for UE graphs.

The proof of the following result is omitted due to space constraints.

Theorem 15 *All of the forbidden subgraphs for* UV *graphs are also forbidden subgraphs for* UEC *graphs.*

This implies that the set $\mathcal{C}_4$ is a *complete* subfamily of forbidden subgraphs for the class UEC, in the following sense.

Theorem 16 *If G is a forbidden subgraph for* UEC *graphs, but not for* UV *graphs, such that G is separated by a maxclique C, such that $\mathscr{A}(G, C)$ is not 3-colorable, and such that each $v \in C$ has at most two neighboring subgraphs, then $G \in \mathcal{C}_4$.*

This result is a direct consequence of the following lemmas, whose proofs are also omitted due to space constraints.

Lemma 1 *If G is a forbidden subgraph for* UEC *graphs, but not for* UV *graphs, such that G is separated by a maxclique C, such that $\mathscr{A}(G, C)$ is not 3-colorable, and such that each $v \in C$ has at most two neighboring subgraphs, then G is a split graph, and G is isomorphic to $\mathsf{constr}(\mathscr{A}(G, C))$.*

Lemma 2 *If G is a split graph separated by its central maxclique C, such that each $v \in C$ has at most 2 neighboring subgraphs and $\delta(\mathscr{A}(G, C)) \geq 3$, then for*

each proper subgraph H of $\mathscr{A}(G, C)$ with $\delta(H) \geq 3$, there exist a proper induced subgraph G' of G and a separating maxclique C' of G' such that $\mathscr{A}(G', C') = H$.

Proof (sketch) [Sketch of proof of Theorem 16] Suppose G satisfies all of the hypotheses. By Lemma 1, G is a split graph, and G is isomorphic to $\mathsf{constr}(\mathscr{A}(G, C))$. Hence, if $\mathscr{A}(G, C)$ was not a 4-critical graph, there would exist a proper subgraph H of $\mathscr{A}(G, C)$ which was not 3-colorable. By Lemma 2, this would imply that there exist a proper induced subgraph G' of G and a separating maxclique C' of G' such that $\mathscr{A}(G', C') = H$. Thus we would have $G' \notin$ UEC, contradicting the hypothesis that G is a forbidden subgraph for UEC. ∎

References

[1] M. R. Cerioli and P. Petito. Forbidden subgraph characterization of split graphs that are UEH. *Electronic Notes in Discrete Mathematics*, 19:305 – 311, 2005. Proceedings of GRACO 2005.

[2] M. Golumbic and R. E. Jamison. The edge intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B*, 38:8 – 22, 1985.

[3] B. Lévêque, F. Maffray, and M. Preissmann. Characterizing path graphs by forbidden induced subgraphs. *Journal of Graph Theory*, 62:369 – 384, 2009.

[4] C. L. Monma and V. K. Wei. Intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B*, 41(2):141 – 181, 1986.

[5] B. S. Panda. The forbidden subgraph characterization of directed vertex graphs. *Discrete Mathematics*, 196(1):239 – 256, 1999.

[6] M. M. Syslo. On characterizations of cycle graphs and on other families of intersection graphs. Technical Report N-40, Institute of Computer Science, University of Wroclaw – Poland, 1978.

[7] M. M. Syslo. Triangulated edge intersection graphs of paths in a tree. *Discrete Mathematics*, 55(2):217 – 220, 1985.

Combined Location and Routing Problems in Drug Distribution

Alberto Ceselli, Giovanni Righini, Emanuele Tresoldi

Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano
{alberto.ceselli, giovanni.righini,
emanuele.tresoldi}@unimi.it

Key words: location-routing, column generation, branch-and-price

1 Introduction

Mathematical programming models and algorithms have been successfully used for decades to optimize operations in distribution logistics: typical examples concern freight carriers, mail services and on-demand pick-up and delivery services.

A more recent field of investigation concerns the application of similar techniques to the optimization of logistics operations in health care systems and emergency management. These sectors are characterized by a larger dependency on "human factors", such as the behavior of the customers (which is often unpredictable), fairness in service provision (which is not an issue in industrial logistics) and lack of reliable historical data (because of the uniqueness of the events considered, especially in case of emergency management).

We tackle a variation of the Vehicle Routing Problem (VRP) arising in the context of the distribution of vaccines and anti-viral drugs in case of a pandemic outbreak. The problem requires to reach the maximum number of citizens within a specified time limit. The starting point for our study is a paper by Shen et al. [7], who presented a stochastic VRP model which is then reformulated and solved as a deterministic VRP with a tabu search algorithm. Our problem can also be seen as a special case of the Team Orienteering Problem (TOP), because the minimization of the fraction of population that is not visited is equivalent to maximize a "profit" from visiting a subset of the sites. Exact algorithms for the TOP have been proposed by Boussier et al. [3], while recent heuristics include that of Archetti et al. [1].

We explore the option of reaching citizens in two ways: by establishing depots, and delivering the drugs at homes using an heterogeneous fleet of vehicles, or by establishing distribution centers where the citizens go by their own means to receive treatments or drugs. In this way a particular combined location-routing problem arises. Even if location-routing is a lively research area [12], to the best of our

knowledge this combined location-routing problem has never been addressed before. We present an exact algorithm for such a problem, which is based on column and cut generation and branch-and-bound, and where the pricing subproblem is solved through advanced dynamic programming techniques.

2 A mathematical programming formulation

Our algorithm is based on the following set partitioning formulation for the problem.

$$\min \sum_{i \in \mathcal{N}} d_i s_i \tag{1}$$

$$\texttt{s.t.} \quad s_i + \sum_{l \in \mathcal{L}} \sum_{h \in \mathcal{H}} \sum_{k \in \mathcal{K}_{lh}} a_{ik} z_k + \sum_{l \in \mathcal{L}} \sum_{w \in \mathcal{W}_l} b_{iw} y_w \geq 1 \qquad \forall i \in \mathcal{N} \tag{2}$$

$$\sum_{l \in \mathcal{L}} \sum_{k \in \mathcal{K}_{lh}} z_k \leq V_h \qquad \forall h \in \mathcal{H} \tag{3}$$

$$\sum_{h \in \mathcal{H}} \sum_{k \in \mathcal{K}_{lh}} z_k + \sum_{w \in \mathcal{W}_l} T y_w \leq T \qquad \forall l \in \mathcal{L} \tag{4}$$

$$\sum_{l \in \mathcal{L}} \sum_{w \in \mathcal{W}_l} y_w \leq C \tag{5}$$

$$s_i, y_w, z_k \qquad \texttt{binary} \tag{6}$$

where $\mathcal{N}$ is the set of sites to be visited, $\mathcal{H}$ is the set of types of available vehicles, $\mathcal{L} \subseteq \mathbb{N}$ is the set of sites in which a depot or a distribution center can be located, $\mathcal{K}_{lh}$ is the set of feasible routes for the vehicles of type $h \in \mathcal{H}$ starting from depot $l \in \underline{\lambda}$, $\mathcal{W}_l$ is the set of feasible clusters for the distribution center $l \in \mathcal{L}$, that is each $w \in \mathcal{W}_l$ is a set of sites that can be served by a distribution center placed in l; V_h is the number of vehicles for each type $h \in \mathcal{H}$, $\sum_{h \in \mathcal{H}} V_h = T$, C it the maximum number of distribution centers allowed, d_i is the number of citizens reached when site $i \in \mathcal{N}$ is visited, $a_{ik} = 1$ iff site $i \in \mathcal{N}$ is in the route $k \in \mathcal{K}_{lh}$, $b_{iw} = 1$ iff site $i \in \mathcal{N}$ is in the cluster $w \in \mathcal{W}_l$, z_k is a binary variable corresponding to route $k \in \mathcal{K}_{lh}$, y_k is a binary variable corresponding to the cluster $w \in \mathcal{W}_l$, s_i is a binary variable corresponding to skipping site $i \in \mathcal{N}$.

We refer to (1)-(6) as the *master problem*.

The model above has an exponential number of columns and so, in order to obtain dual bounds, its linear relaxation is solved by column generation. Since the columns in our problem can represents either a route or a cluster there are two different pricing problems everyone associated with a particular distribution strategy. The expression of the reduced costs of the "route columns" is as follows:

$$r'_{hl} = -\sum_i \pi_i a_i + \mu_h + \sigma_l \tag{7}$$

where $\pi \geq 0$ is the vector of dual variables corresponding to constraints (2), $\mu_h \leq 0$ with $h \in \mathcal{H}$ is the scalar dual variable corresponding to constraint (3), $\sigma_l \leq 0$ with

$l \in \mathcal{L}$ is the scalar dual variable corresponding to constraint (4). The constraints of this pricing problem require that: the route is elementary, it must start at the depot, all sites along the route must be visited within a specified deadline and the sum of the demands in the sites reached must be less then or equal to the capacity of the vehicle used. The resulting problem is an elementary shortest path problem with resource constraints.

The "distribution center columns" have the following reduced costs expression:

$$r_l'' = -\sum_i \pi_i b_i + \sigma_l + \rho \tag{8}$$

where $\pi \geq 0$ is the vector of dual variables corresponding to constraints (2), $\sigma \leq 0$ with $l \in \mathcal{L}$ is the scalar dual variable corresponding to constraint (4), $\rho \leq 0$ is the scalar dual variable corresponding to constraint (5). In this case constraints of the pricing problem only require that: the sum of the citizens in the sites covered by the distribution center must be less than or equal to its capacity, and that every covered site must be within a given range of the distribution center. This problem is a classical 0-1 knapsack problem.

Pricing algorithms. Pricing is the most time-consuming part in the branch-and-price exact algorithm relying upon the previous formulation. To speed up the pricing phase we rely on both heuristic and exact pricing algorithms. Exact pricing for the "route columns" is done by bi-directional dynamic programming with decremental state space relaxation, following the approach described in Righini and Salani [5] [6], heuristic pricing relies on greedy algorithms and dynamic programming with relaxed dominance conditions. For the "distribution center columns" we used a modified version of the min-core pseudo-polynomial time algorithm proposed by Pisinger [8].

Cuts generation and branching. In order to devise an efficient and performing branch and price algorithm we implemented a stabilization method to mitigate possible convergence problems [11] and we took into account additional inequalities coming from modifications and extensions of classical cuts such as subset rows inequalities [9] and 2-path cuts[10]. Finally five different branching strategies were developed in order to recover the integrality when a fractional solution arises.

3 Computational Results

For testing purpose we have generated 630 instances, up to 50 nodes, using the same procedure described in [7]. We run the algorithm on all instances using four different scenarios:

VRP: a classical VRP, single depot, homogeneous fleet of vehicles,

VRP-HF: VRP with heterogeneous fleet of vehicles,

VRP-HF-MD: in this scenario an additional location problem is taken into account, in fact multiple possible location for building depots are available,

VRP-HF-MD-DC: this scenario represents the complete problem with all the previous features and the alternative distribution strategy.

We analyzed the variation in the quality of the solution measured in terms of fraction of population served and distance from the optimality for every scenario. Aggregate average results are reported in Table 6.

	VRP	VRP-HF	VRP-HF-MD	VRP-HF-MD-DC
Optimality	opt: 630/630	opt: 622/630 avg gap: 1% max gap: 3%	opt: 599/630 avg gap: 2% max gap: 14%	opt: 612/630 avg gap: 1% max gap: 9%
Pop. reached	Avg: 82.11 %	Avg: 82.45%	Avg: 89.36%	Avg: 90.82%
Time (s)	Max Time: 857 Avg Time: 11	Avg Time: 161	Avg Time: 274	Avg Time: 170

Table 6. Average results on four different scenarios

References

[1] C. Archetti, A. Hertz and M.G. Speranza, "Metaheuristics for the team orienteering problem", *Journal of Heuristics* 13, 49-76 (2007).

[2] R. Baldacci, M. Battarra and D. Vigo, "Routing a Heterogeneous Fleet of Vehicles", in *The Vehicle Routing Problem: latest advances and new challenges*, B.L. Golden, S. Raghavan and E. Wasil (eds), 3-27, Springer Science + Business Media (2008).

[3] S. Boussier, D. Feillet and M. Gendreau, "An exact algorithm for team orienteering problems", *4OR* 5, 211-230 (2007).

[4] L. Ke, C. Archetti and Z. Feng, "Ants can solve the team orienteering problem", *Computers and Industrial Engineering* 54, 648-665 (2008).

[5] G. Righini and M. Salani, "Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints", *Discrete Optimization* 3, 255-273 (2006).

[6] G. Righini and M. Salani, "New Dynamic Programming Algorithms for the Resource Constrained Elementary Shortest Path Problem", *Networks* 51, 155-170 (2008).

[7] Z. Shen, F. Ordóñez and M. Dessouky, "A Two-Stage Vehicle Routing Model for Large-Scale Bioterrorism Emergencies", *Networks* 54(4), 255-269 (2009).

[8] D. Pisinger, "A minimal algorithm for the 0-1 knapsack problem", *Operations Research* 45, 758-767 (1997).

[9] M. Jepsen, B. Petersen, S. Spoorendonk, D. Pisinger, “Subset-Row Inequalities Applied to the Vehicle-Routing Problem with Time Windows” *Operations Research* 56, 497-511 (2008).

[10] N. Kohl, J. Desrosiers, O. Madsen, M. Solomon, F. Soumis, “2-path cuts for the vehicle routing problem with time windows” *Transportation Science* 33, 101-116 (1999).

[11] A. Pessoa, E. Uchoa, M. Poggi de Aragao. “ A new stabilization method for column generation” *Column Generation Workshop*, conference presentation, Aussois, (2008).

[12] G. Nagy, S. Salhi, “Location-routing: Issues, models and methods” *European Journal of Operational Research* 177, 649 - 672 (2007).

The impact of the norm on the k-Hyperplane Clustering problem: relaxations, restrictions, approximation factors, and exact formulations

Stefano Coniglio

Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza L. da Vinci 32, 22133, Milano
coniglio@elet.polimi.it

Key words: hyperplane clustering, nonconvex optimization, global optimization, convex relaxations and restrictions, approximation factors

---

1 Introduction

Given m points $\{\underline{a}_1, \ldots, \underline{a}_m\}$ in $\mathbb{R}^n$, the k-*Hyperplane Clustering* problem is to assign each point to one of k clusters and to determine, for each of them, a hyperplane so as to minimize the sum, over all the points, of the square of their orthogonal distances to the corresponding hyperplane, measured in p-norm.

Given a point $\underline{a}_i$ and a hyperplane of equation $\underline{w}_j^T \underline{x} = \gamma_j$, where $\underline{w}_j \in \mathbb{R}^n$ and $\gamma_j \in \mathbb{R}$, their p-*norm orthogonal distance*, for $p \in \mathbb{N} \cup \{\infty\}$, amounts to $\frac{|\underline{w}_j^T \underline{a}_i - \gamma_j|}{\|\underline{w}_j\|_{p'}}$, where $\frac{1}{p} + \frac{1}{p'} = 1$. Here, we consider the case where $p = 2$ (Euclidean norm), because it allows to find solutions that are invariant to rotations. Since $\frac{1}{2} + \frac{1}{2} = 1$, for $p = 2$ we have $p = p'$ and, therefore, the distance function amounts to $\frac{|\underline{w}_j^T \underline{a}_i - \gamma_j|}{\|\underline{w}_j\|_2}$.

The problem is nonconvex, due to the nonconvexity of the objective function. It is also $\mathcal{NP}$-hard for any choice of p, since the problem of deciding whether k lines can fit m points in $\mathbb{R}^2$ with zero error is $\mathcal{NP}$-complete [MT82].

The k-Hyperplane Clustering problem is relevant in many fields, such as, for instance, data mining [BM00], operations research [MT82], and line detection in digital images [AM02]. Heuristic approaches have been proposed in [BM00] and [AC09]. For the bottleneck version of the problem, see [Dhy09].

In this work, we propose exact formulations for the problem which yield globally optimal solutions, as well as relaxations and restrictions that guarantee a certain approximation factor.

Since, of the $n + 1$ parameters that characterize a hyperplane, only n are independent, a constraint can be introduced for each cluster without loss of generality. By imposing $\|\underline{w}_j\|_2 = 1$, for each $j = 1, \ldots, k$, we can replace the nonconvex expression $\frac{|\underline{w}_j^T \underline{a}_i - \gamma_j|}{\|\underline{w}_j\|_2}$ with $|\underline{w}_j^T \underline{a}_i - \gamma_j|$. In addition, since we can prove that any solution

where $\|\underline{w}_j\|_2 > 1$, for some j, cannot be optimal, we can relax $\|\underline{w}_j\|_2 = 1$ into $\|\underline{w}_j\|_2 \geq 1$. We formulate the problem as the following nonconvex Mixed-Integer Quadratically Constrained Quadratic Program:

$$\min \quad \sum_{i=1}^{m} y_i^2 \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{k} x_{ij} = 1 \qquad i = 1, \ldots, m \tag{2}$$

$$y_i \geq (\underline{w}_j^T \underline{a}_i - \gamma_j) x_{ij} \qquad i = 1, \ldots, m, j = 1, \ldots, k \tag{3}$$

$$y_i \geq (-\underline{w}_j^T \underline{a}_i + \gamma_j) x_{ij} \qquad i = 1, \ldots, m, j = 1, \ldots, k \tag{4}$$

$$\|\underline{w}_j\|_2 \geq 1 \qquad j = 1, \ldots, k \tag{5}$$

$$y_i \in \mathbb{R}^n_+ \qquad i = 1, \ldots, m \tag{6}$$

$$w_j \in \mathbb{R}^n, \gamma_j \in \mathbb{R} \qquad j = 1, \ldots, k \tag{7}$$

$$x_{ij} \in \{0, 1\} \qquad i = 1, \ldots, m, j = 1, \ldots, k. \tag{8}$$

The binary variable x_{ij} equals 1 if the point $\underline{a}_i$ is assigned to cluster j and 0 otherwise. The continuous variables $(\underline{w}_j, \gamma_j)$, for $j = 1, \ldots, k$, correspond to the hyperplane parameters. The variable y_i, due to Constraints (3)–(4), represents the distance from the point $\underline{a}_i$ to the hyperplane associated to cluster j. Constraints (2) are assignment constraints. Constraints (3)–(4) can be linearized via the well-known McCormick envelope. In addition, valid lower and upper bounds for all the continuous variables can be derived and introduced to tighten the formulation. We solve (1)–(8) with a standard Spatial Branch-and-Bound method, as implemented in the solver COUENNE [BLL$^+$09]. It is based on iteratively splitting the nonconvex feasible region into subregions and on finding lower bounds for each subregion by constructing and optimizing over a polyhedral envelope, adopting a Branch-and-Bound technique.

2 Relaxations and restrictions within an approximation factor

The problem that is obtained after reformulating Constraints (3)–(4) is nonconvex due to the 2-norm Constraints (5). We propose to substitute them with either $\|\underline{w}_j\|_1 \geq c$ or $\|\underline{w}_j\|_\infty \geq c$ (adopting the so-called polyhedral norms), for an appropriate $c > 0$ (see below). Since these constraints can be described by introducing linear constraints and mixed-integer variables, the corresponding k-Hyperplane Clustering problem (in 1-norm or ∞-norm) can be formulated as a Mixed-Integer Quadratic Program, which is easier to solve than the original 2-norm one. As summarized in the following theorem, when formulating the problem with either $\|\underline{w}_j\|_1 \geq c$ or $\|\underline{w}_j\|_\infty \geq c$, for an appropriately chosen c, we obtain relaxations and restrictions which yield lower and upper bounds on the optimal value of the original problem within an approximation factor.

Theorem 17 *Let OPT be the value of an optimal solution of the 2-norm k-Hyperplane Clustering problem.*

- *Let LB be the value of an optimal solution obtained after replacing each constraint* $\|w_j\|_2 \geq 1$ *with either* $\|w_j\|_1 \geq 1$ *or* $\|w_j\|_\infty \geq \frac{1}{\sqrt{n}}$. *We have* $\frac{1}{n}$*OPT* $\leq$ *LB* $\leq$ *OPT.*
- *Let UB be the value of an optimal solution obtained after replacing each constraint* $\|w_j\|_2 \geq 1$ *with either* $\|w_j\|_1 \geq \sqrt{n}$ *or* $\|w_j\|_\infty \geq 1$. *We have OPT* $\leq$ *UB* $\leq n$*OPT.*

The proof, which is omitted for lack of space, is based on two ingredients: finding a value $c > 0$ such that the set that is feasible for $\|w\|_p \geq c$ either contains or is contained into that which is feasible for $\|w\|_2 \geq 1$ and applying equivalence relations between p-norms.

Given any $p \in \mathbb{N} \cup \{\infty\}$ and $c > 0$, we can show that the solution that is obtained when imposing $\|w\|_p \geq c$ can be obtained from that corresponding to $\|w\|_p \geq 1$ by scaling the vectors $\underline{w}_j$, for $j = 1, \ldots, k$, by c. Therefore, we can solve just two of the four problems in the theorem, one per norm, and then derive the solution of the other two by scaling.

The formulations are solved via the Branch-and-Bound method for Mixed-Integer Quadratic programs of CPLEX.

3 Enhanced formulations

We propose three enhanced exact formulations for the 2-norm k-Hyperplane Clustering problem, where the constraints $\|w_j\|_2 \geq 1$, $\|w_j\|_1 \geq 1$, and $\|w_j\|_\infty \geq \frac{1}{\sqrt{n}}$ are simultaneously imposed for each $j = 1, \ldots, k$, even if the last two are redundant and weaker than the former. Such formulations allow to achieve a tighter lower bound within fewer nodes of the enumeration tree, thus leading to an overall faster Spatial Branch-and-Bound algorithm.

Note that, due to the presence of the squares in the objective function, by replacing $\underline{w}_j$ with $-\underline{w}_j$, for any $j = 1, \ldots, k$, we obtain an equivalent solution. We break this form of symmetry by constraining the first component of each $\underline{w}_j$ to be nonnegative.

4 Computational results

Computational experiments are carried out on a set of 88 realistic randomly generated instances, with $m = 10, \ldots, 30$, $n = 2, 3$, and $k = 2, 3$. They are obtained with the generator used in [AC09].

We investigate the practical tightness of the relaxations and restrictions that are obtained when using either the 1-norm, the ∞-norm, or both. The first table compares the theoretical approximation factors as in Theorem 17 (columns three and four) with those that are empirically found by solving the formulations (columns five and six). For each value of n, the empirical worst-case approximation factor is reported. Interestingly, the empirical approximation factors of the lower bounds match the theoretical ones, whereas much tighter upper bounds are obtained.

n	p-norm	Theoretical LB	Theoretical UB	Empirical LB	Empirical UB
2	$\|\underline{w}_j\|_1$	0.5	2	0.51	1.61
	$\|\underline{w}_j\|_\infty$			0.51	1.50
	$\|\underline{w}_j\|_1$, $\|\underline{w}_j\|_\infty$	0.5	-	0.63	1.62
3	$\|\underline{w}_j\|_1$	0.33	3	0.36	1.58
	$\|\underline{w}_j\|_\infty$			0.33	1.61
	$\|\underline{w}_j\|_1$, $\|\underline{w}_j\|_\infty$	0.33	-	0.33	1.75

The second table reports the percentage of instances that are solved to optimality within two hours of CPU time, using the exact formulation in (1)-(8) (first row) and those enhanced with 1-norm constraints (2nd row), ∞-norm constraints (3rd row), and both (4th row). The results show that, for both $n = 2$ and $n = 3$, the introduction of redundant constraints in 1 or ∞-norm almost doubles the number of instances that are solved to global optimality within the time limit.

p-norm	% instances solved to optimality $n = 2$	$n = 3$
$\|\underline{w}_j\|_2$	50	21
$\|\underline{w}_j\|_2$, $\|\underline{w}_j\|_1$	90	42
$\|\underline{w}_j\|_2$, $\|\underline{w}_j\|_\infty$	80	47
$\|\underline{w}_j\|_2$, $\|\underline{w}_j\|_1$, $\|\underline{w}_j\|_\infty$	90	42

References

[AC09] E. Amaldi and S. Coniglio. An adaptive point-reassignment metaheuristic for the k-hyperplane clustering problem. In *Proc. of Metaherustic International Conference*, pages 1–10, 2009.

[AM02] E. Amaldi and M. Mattavelli. The MIN PFS problem and piecewise linear model estimation. *Discrete Applied Mathematics*, 118(1-2):115–143, 2002.

[BLL+09] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bound tightening techniques for non-convex MINLP. *Optimization methods and software*, 24:597–634, 2009.

[BM00] P. Bradely and O. Mangasarian. k-plane clustering. *Journal of Global Optimization*, 16:23–32, 2000.

[Dhy09] K. Dhyani. *Optimization models and algorithms for the hyperplane clustering problem*. PhD thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy, 2009.

[MT82] N. Megiddo and A. Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1(5):194–197, 1982.

A Lagrangian Relaxation Approach for Gene Regulatory Networks

Roberto Cordone, [a] Guglielmo Lulli [b]

[a] *University of Milano, Department of Computer Science*
Via Comelico 39, 20135 - Milano, Italy
roberto.cordone@unimi.it

[b] *University of Milano "Bicocca", Department of Informatics, Systems and Communication*
viale Sarca 336, 20122 Milano, Italy
lulli@disco.unimib.it

Key words: Gene regulatory networks, Lagrangean relaxation, Branch-and-bound, Tabu Search

1 The Weighted Gene Regulatory Network problem

A gene regulatory network $\mathcal{G}(N, A \cup I, w)$ is a graph model of the dynamics of gene expression in a living cell: the set of nodes N represents gene products, two disjoint sets of arcs A and I represent, respectively, activation and inhibition influences between the gene products. Each arc has a weight $w : A \cup I \rightarrow (]0; 1]$, derived from statistical correlation indices: $w_{ij} = 0$ denotes a full correlation between i and j; $w_{ij} = 1$ no correlation.

The problem is to identify a subset of nodes which explain the expression of all other nodes, by acting either as activators or as inhibitors. A node should only exert influences coherent with its label, or no influence at all. However, biological evidence suggests the presence of few genes exerting both kinds of influence. To account for this, given the minimum feasible number M of labelled nodes, the model minimizes the total weight of the influences (i.e., arcs) exerted by nodes labelled incoherently with respect to the arc. Let $z_i^{(A)} = 1$ if i is labelled as activator, 0 otherwise, $z_i^{(I)} = 1$ if i is labelled as inhibitor, 0 otherwise. Let $x_{ij} = 1$ if arc $(i, j) \in A \cup I$ represents an incoherent influence, 0 otherwise.

$$\min \quad \phi = \sum_{(i,j) \in A \cup I} w_{ij} \cdot x_{ij}$$

$$
\begin{aligned}
s.t. \sum_{i:(i,j)\in A} z_i^{(A)} + \sum_{i:(i,j)\in A} x_{ij} &\geq 1 \qquad j \in N && (1)\\
\sum_{i:(i,j)\in I} z_i^{(I)} + \sum_{i:(i,j)\in I} x_{ij} &\geq 1 \qquad j \in N && (2)\\
z_i^{(A)} + z_i^{(I)} &\leq 1 \qquad i \in N && (3)\\
\sum_{i\in N} (z_i^{(A)} + z_i^{(I)}) &= M && (4)\\
x_{ij} &\leq z_i^{(I)} \qquad (i,j) \in A && (5)\\
x_{ij} &\leq z_i^{(A)} \qquad (i,j) \in I && (6)\\
x_{ij} &\in \{0,1\} \qquad (i,j) \in A \cup I && (7)\\
z_i^{(A)}, z_i^{(I)} &\in \{0,1\} \qquad i \in N && (8)
\end{aligned}
$$

Each node receives at least one activating and one inhibiting arc from the identified subset, either coherent with the label of the source node or not (1,2). Each gene is labelled as activator, inhibitor or neutral (3). The number of activator and inhibitor nodes is bounded from above, but any optimal solution with less than M labelled nodes can be replaced by an equivalent one with exactly M labelled nodes (4). An incoherent influence requires the arc and the source node to have opposite labels (5,6).

2 A Lagrangian branch-and-bound

To compute lower bounds on the optimum, we dualize the covering constraints (1) and (2) with multipliers λ_i and π_i, respectively :

$$
\mathcal{L}(z,x,\lambda,\pi) = \sum_{(i,j)\in A} \gamma_{ij}\cdot x_{ij} + \sum_{(i,j)\in I} \delta_{ij}\cdot x_{ij} - \sum_{i\in N}(\alpha_i \cdot z_i^{(A)} + \beta_i \cdot z_i^{(I)}) + \sum_{i\in N}(\pi_i + \lambda_i)
$$

where

$$
\begin{aligned}
\alpha_i &= \sum_{j\in N:(i,j)\in A} \lambda_j \quad i \in N \qquad & \gamma_{ij} &= w_{ij} - \lambda_j \quad (i,j) \in A\\
\beta_i &= \sum_{j\in N:(i,j)\in I} \pi_j \quad i \in N \qquad & \delta_{ij} &= w_{ij} - \pi_j \quad (i,j) \in I
\end{aligned}
$$

Once the z variables have been fixed, the optimal values of the x variables are uniquely determined: if $(i,j) \in A$, $x_{ij} = 1$ if $\gamma_{ij} < 0$ and $z_i^{(I)} = 1$, 0 otherwise; if $(i,j) \in I$, $x_{ij} = 1$ if $\delta_{ij} < 0$ and $z_i^{(A)} = 1$, 0 otherwise. Thus, for each node i we can sum to weight α_i (or β_i) all negative weights δ_{ij} (or γ_{ij}) for the arcs exiting that node. This reduces the problem to optimizing the z variables subject to cardinality and disjunctive constraints. To solve it, we sort the nodes by increasing values of the modified weights and label the first M nodes as activator or inhibitor, according to the label with the lower weight.

The multipliers are initially set to zero; then, they are tuned by a modified subgradient procedure. This iteratively solves the Lagrangian relaxed problem and updates the multipliers based on a moving average of the violations of the covering constraints in the relaxed Lagrangian solutions computed so far.

The branching mechanism operates by labelling nodes: it selects a node and generates two subproblems by forcing or forbidding it to be neutral (without specifying its label in the latter case). When all nodes are neutral or nonneutral, a second branching mechanism generates two subproblems by forcing a labelled node to be an activator or an inhibitor. The branching node is chosen on the basis of the Lagrangian multipliers. First, we compute for each node the sum of the Lagrangian multipliers over all the nodes covered by it. Then, we select the node with the largest sum, so that labeling it will have the strongest possible influence on the constraints.

3 A Tabu Search algorithm

To compute feasible solutions of the problem (upper bounds) we implemented a Tabu Search (TS) metaheuristic for the WGRN problem. This computes a starting solution with a greedy procedure, and improves it with a local search procedure based on a natural neighbourhood. As finding a feasible solution is an NP-complete problem, both phases relax the covering constraints (1,2) and evaluate the solutions lexicographically according to: i) the number of violated constraints, v; ii) the cost, ϕ.

The neighbourhood of solution S is the set of solutions $\mathcal{N}_S$ obtained by i) turning a single activator (inhibitor) node into inhibitor (activator); ii) assigning the label of a nonneutral node to a neutral one and turning the former into neutral. These moves do not modify the total number of labelled nodes, M. The neighbourhood is composed of $M+M\,(|N|-M)$ solutions, not necessarily feasible, and it is always completely visited.

In order to avoid a cyclic behaviour, the algorithm stores for each node i the last iterations ℓ_i^N, ℓ_i^A and ℓ_i^I, in which node i was, respectively, neutral, activator or inhibitor. At each iteration, the algorithm visits the whole neighbourhood of the current solution S and divides the solutions into tabu and non tabu, based on the values stored in ℓ_i^N, ℓ_i^A and ℓ_i^I and on a *tabu tenure* L: a solution is tabu if it assigns to a node a label assumed less than L iterations ago. The best non tabu neighbour solution replaces the current one. As an exception, the best tabu neighbour solution is selected if it is better and it improves the best one found so far (*aspiration criterium*).

To enhance the search, the tabu tenure is adaptively tuned within a fixed range $[L_{\min}; L_{\max}]$): L increases when the current solution gets worse, decreases when it improves. The aim is to guide the search away from already visited local optima and towards not yet visited local optima. To diversify the search, the algorithm is terminated and restarted r times from solutions generated selecting M random nodes, half labelled as activators, half as inhibitors.

4 Experimental results

We have generated at random four groups of ten benchmark instances with $|N| = 100$ nodes. First we have randomly drawn the indegree of each node in a suitable range ($[10; 20]$, $[30; 50]$, $[60; 80]$ or $[99; 99]$), then randomly generated the ingoing arcs, classifying them half as activators half as inhibitors, and finally associated to each arc a random cost from $[0; 1]$. The number M of labelled nodes was set to the minimum for which a feasible solution exists, to respect the parsimony principle required by the application.

The benchmark is quite hard for the basic TS (one third of unsolved instances even after $K_{\max} = 100\,000$ iterations and with several different tabu tenure settings). However, random restart overcomes this ineffectiveness: $r = 100$ runs of $1\,000$ iterations perform well (all solutions are feasible and 26 hit the best known result), and $r = 1\,000$ run of 100 iterations even better (all solutions are feasible and 37 hit the best known result).

Neither the Lagrangean branch-and-bound nor CPLEX 11 could solve any instance within one hour, and the gaps are wide. However, the upper bound found by TS is always better than that computed by CPLEX. The Lagrangian lower bound is much stronger than that computed by CPLEX on most instances (two orders of magnitude on the complete ones), except for the sparsest class of instances, where it is comparable. CPLEX generates much more branching nodes than the Lagrangean approach, in sharp contrast with the fact that the Lagrangean subproblem enjoys the integrality property, and therefore provides (at the root node) a bound equivalent to the linear one. This might be due to a stronger effectiveness of the branching strategy adopted.

Bound constraints for Point Packing in a Square

Alberto Costa, [a] Pierre Hansen, [a,b] Leo Liberti [a]

[a] *LIX (UMR CNRS 7161), École Polytechnique, 91128 Palaiseau, France.*
{costa, liberti }@lix.polytechnique.fr

[b] *GERAD and HEC Montreal, Canada.*
pierre.hansen@gerad.ca

Key words: point packing in a square, nonconvex NLP, bound constraints.

1 Introduction

In this paper we present a conjecture about the bounds on the variables for the Point Packing in a Square (PPS) problem. There exist several formulations for this problem, most of them are introduced in [1]; in order to semplify the following presentation, we use this formulation:

> *Place n points in the unit square such that the minimum pairwise distance is maximal.*

This problem can be formulated this way:

$$\max \alpha \tag{1}$$
$$\forall i < j \leq n \quad (x_i - x_j)^2 + (y_i - y_j)^2 \geq \alpha \tag{2}$$
$$\forall i \leq n \quad x_i \leq 1 \tag{3}$$
$$\forall i \leq n \quad y_i \leq 1 \tag{4}$$
$$\forall i \leq n \quad x_i \geq 0 \tag{5}$$
$$\forall i \leq n \quad y_i \geq 0 \tag{6}$$
$$\alpha \geq 0. \tag{7}$$

The positive variable α is the square of the minimum pairwise distance between the points. Constraints (2) are the distance inequalities, while inequalities (3)-(6) mean that the points are inside the unit square.

When we try to solve PPS by means of solvers which implement the spatial Branch-and-Bound algorithm [2,3], like COUENNE [4] or BARON [5], we notice that it is not easy to decrease the value of the upper bound on α during the computation. In

⋆ Financial support by grants: Digiteo 2009-14D "RMNCCO", Digiteo 2009-55D "ARM" is gratefully acknowledged.

next Section, we will show that it depends also on the bound on the variables, and we propose a method to obtain better results by modifying the inequalities (3)-(6). After that, in Section 3 we present some computational results, while in Section 4 there are the conclusions and the future work.

2 Bounds on the variables

PPS is a nonlinear nonconvex problem; when we try to solve it by means of spatial Branch-and-Bound, usually the root node corresponds to a linear relaxation of the problem, whose optimal solution represents an upper bound for the original problem. For the formulation (1)-(7) the relaxation is the following (as explained in [6,7]).

$$\max \alpha \tag{8}$$
$$\forall i < j \leq n \quad -l(i,j) \geq \alpha \tag{9}$$
$$\forall i \leq n \quad x_i \leq 1 \tag{10}$$
$$\forall i \leq n \quad y_i \leq 1 \tag{11}$$
$$\forall i \leq n \quad x_i \geq 0 \tag{12}$$
$$\forall i \leq n \quad y_i \geq 0 \tag{13}$$
$$\alpha \geq 0. \tag{14}$$

where $l(i,j) = -(Lx_i - Ux_j + Ux_i - Lx_j)(x_i - x_j) - (Ly_i - Uy_j + Uy_i - Ly_j)(y_i - y_j) + (Lx_i - Ux_j)(Ux_i - Lx_j) + (Ly_i - Uy_j)(Uy_i - Ly_j)$ is the convex envelope of the nonlinear part of constraint (2) while L and U represent respectively the lower and upper bounds on the variables (in this case, $L = 0$ and $U = 1$ for all the variables).

Proposition 2.1 *The optimal solution of the problem* (8)-(14) *is* $\alpha^* = 2$.

Proof (sketch) It is easy to see that when all the lower bounds have the same value L, and the upper bounds have the same value U, then $-l(i,j) = 2(U-L)^2$. The problem of maximizing α, with the constraints $\forall i < j \leq n \quad \alpha \leq 2(U-L)^2$, has obviously optimal solution $\alpha^* = 2(U-L)^2$. Since $L = 0$ and $U = 1$, then the optimal solution of (8)-(14) is $\alpha^* = 2$. ∎

The bound provided by the previous relaxation is not very good: since α is the square of the minimum distance between the points, the upper bound on the distance is $\sqrt{2}$, that is the optimal solution obtained when there are only 2 points in the square, placed in two opposite vertices. Furthermore, this bound does not depend on the number of points n, nor on the value of the variables x and y: due to the fact that all the lower (upper) bounds have the same value, in the linear relaxation $l(i,j)$ all the coefficients of the terms containing x and y become 0.

In order to improve the bound on α, we should change the value of lower and upper bounds for some variables; thus, the corresponding terms containing x and y in the linear relaxation do not disappear. The following conjecture refers to that idea.

Conjecture 3 *Consider an instance of PPS with* n *points. Divide the unit square in* k^2 *equal subsquares, with* $k = \arg\min_s \left| \frac{n}{2} - s^2 \right|$, $s \in \left\{ \left\lceil \sqrt{\frac{n}{2}} \right\rceil, \left\lfloor \sqrt{\frac{n}{2}} \right\rfloor \right\}$. *There is at least one point of the optimal solution in each subsquare.*

		Original formulation		Bounds constraints formulation	
n	d^*	LB	UB	LB	UB
9	0.5	0.000098	1.414213	**0.300463**	**0.707107**
10	0.421279	0.000098	1.414213	**0.396156**	**0.707107**
11	0.398207	0.000099	1.414213	0.000099	**0.707107**
12	0.388730	0.000099	1.414213	**0.360065**	**0.707107**
13	0.366096	0.000098	1.414213	**0.339654**	**0.502948**
14	0.348915	0.000098	1.414213	**0.340830**	**0.502874**
15	0.341081	0.000098	1.414213	**0.334524**	**0.502793**
16	0.333333	0	1.414213	**0.290033**	**0.502793**
17	0.306153	0	1.414213	**0.000099**	**0.502793**
18	0.300462	0	1.414213	**0.252819**	**0.502793**
19	0.289541	0.000047	1.414213	**0.252337**	**0.502793**
20	0.286611	0	1.414213	**0.276468**	**0.502793**

The meaning of this conjecture is that we can change the value of the bounds for k^2 points. For example, consider the case with $n = 9$: here, $k = 2$, so there are 4 subsquares. According to the conjecture, we can place one point in each subsquare; for instance, if we put the point i is in the bottom left subsquare, we can modify the bounds provided by (3)-(4) obtaining $x_i \leq 0.5$ and $y_i \leq 0.5$.

In order to change other bounds, we can use these properties of the optimal solution, as remarked in [8]:

- at least $n_x = \lceil \frac{n}{2} \rceil$ points are on the left half of the square (x bounds property);
- among the previous n_x points, at least $n_y = \lceil \frac{n_x}{2} \rceil$ are on the bottom half (y bounds property).

After dividing the square in k^2 subsquares, we have placed in the left half of the square $\eta < n_x$ points, so for others $n_x - \eta$ points we can change the upper bounds on the variable x from 1 to 0.5, according to x bounds property. A similar idea can be used for the y bounds property.

3 Results

In this Section we present the values of the upper and lower bounds for some instances of the PPS problem obtained at the root node, using COUENNE, with and without the bound constraints presented in the previous section. Furthermore, we present the values of the optimal distance $d^* = \sqrt{\alpha^*}$ for these solutions (which can be found in `http://www.packomania.com` and [1]).

4 Conclusions and future work

In this paper we showed the effect of the bounds constraints: the upper bounds obtained are better, as well as the lower bounds (namely the best solutions found). Moreover, we can see an improvement of the upper bounds from the instance $n = 12$ (where $k = 3$) to the instance $n = 13$ (where $k = 4$).

The future work has three main directions: first, we want to prove the conjecture presented in this paper. Second, we want to try other kinds of subdivision of the

square. Finally, we will try to adapt these ideas for other formulations of this problem where some symmetry breaking constraints are used [9,10].

References

[1] P. G. Szabó, M. Cs. Markót, T. Csendes, E. Specht, L. G. Casado and I. Garca. New Approaches to Circle Packing in a Square: With Program Codes. Springer Optimization and Its Applications, Springer-Verlag New York, 2007.

[2] E. Smith and C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimization of nonconvex MINLPs. *Computers & Chemical Engineering*, 23: 457–478, 1999.

[3] L. Liberti. Writing global optimization software. In L. Liberti and N. Maculan, editors, *Global Optimization: from Theory to Implementation*, 211-262, Springer, 2006.

[4] P. Belotti, J. Lee, L. Liberti, F. Margot and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4): 597–634, 2009.

[5] N.V. Sahinidis and M. Tawarmalani. BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs. User's Manual, 2005.

[6] M. Locatelli and U. Raber. Packing equal circles in a square: II. A Deterministic Global Optimization Approach. *Technical Report 09-99*, Dip. Sistemi e Informatica, Univ. di Firenze, 1999.

[7] U. Raber. Nonconvex all-quadratic global optimization problems: solution methods, application and related topics. *Ph.D. Thesis*, University of Trier, Germany, 1999.

[8] K. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43(2): 471–484, 2009.

[9] P. Hansen A. Costa and L. Liberti. Formulation symmetries in circle packing. In R. Mahjoub, editor, *ISCO 2010 Proceedings*, Electronic Notes in Discrete Mathematics, 36: 1303–1310, Elsevier, 2010.

[10] A. Costa, P. Hansen and L. Liberti. Static symmetry breaking in circle packing. In U. Faigle, editor, *CTW 2010 Proceedings*, 47–50, University of Köln, 2010.

Bicolored independent sets and bicliques

Jean-François Couturier, Dieter Kratsch

Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France
{couturier,kratsch}@univ-metz.fr

Key words: exact exponential algorithms, graph algorithms, NP-hard problems, bicolored independent sets, bicliques.

1 Introduction

Throughout the paper all graphs $G = (V, E)$ are undirected and simple.

A *bicolored* graph is a graph $G = (V, E)$ in which each vertex is colored with one of two possible colors, let us call them red and blue. We denote such a bicolored graph by $G = (R, B, E)$ where R is the set of red vertices and B is the set of blue vertices. Let us emphasize that the coloring of G by red and blue colors is not required to be a proper coloring. We introduce the following natural generalization of the well-known INDEPENDENT SET problem.

Bicolored Independent Set: The BICOLORED INDEPENDENT SET problem (BIS) has as input a graph $G = (R, B, E)$ and non-negative integers k_1 and k_2; the task is to decide whether there is an independent set $I \subseteq V$ of G such that $|I \cap R| = k_1$ and $|I \cap B| = k_2$.

A simple reduction from the NP-complete INDEPENDENT SET problem shows that BIS is NP-complete. Our original motivation to introduce and study the BIS problem was its strong relation to well-studied biclique problems.

Let the vertex sets X and Y be independent sets of a graph $G = (V, E)$ such that $xy \in E$ for all $x \in X$ and $y \in Y$. The pair (X, Y) is called a *biclique* of G. The pair (X, Y) is called a *non-induced biclique* of the graph G if $xy \in E$ for all $x \in X$ and $y \in Y$. (Note that neither X nor Y are required to be independent sets of G.) Applications of bicliques in data mining, artificial intelligence, automata, language theory and biology are discussed in [1]. The complexity of algorithmic problems on bicliques has been studied extensively (see e.g. [5]).

We are interested in finding bicliques in bipartite graphs and in finding non-induced bicliques in graphs.

⋆ This work has been supported by the ANR project AGAPE.

Bipartite Biclique: The input of the problem is a bipartite graph $G = (X, Y, E)$ and non-negative integers k_1 and k_2; the task is to decide whether there is a biclique (X', Y') of G such that $|X'| = k_1$, $|Y'| = k_2$, $X' \subseteq X$ and $Y' \subseteq Y$.

Noninduced Biclique: The input of the problem is a graph $G = (V, E)$ and non-negative integers k_1 and k_2; the task is to decide whether there is an non-induced biclique (X', Y') of G such that $|X'| = k_1$ and $|Y'| = k_2$.

The best known exact algorithms for these problems run in time $O(1.6914^n)$ for the NONINDUCED BICLIQUE problem and $O(1.3006^n)$ for the BIPARTITE BICLIQUE problem; both needing only polynomial space [2].

Our Results. An algorithm solving the BICOLORED INDEPENDENT SET problem in time $O(1.2691^n)$ (needing exponential space) is our main result. It is established by combining a branching algorithm with a dynamic programming algorithm. The time analysis of the overall algorithm is based on Measure & Conquer. This approach has been used to design and analyse exact exponential-time algorithms for the dominating set problem [3,6].

Our algorithm solving the BICOLORED INDEPENDENT SET problem can be used to establish an $O(1.2691^n)$ time algorithm solving the BIPARTITE BICLIQUE problem and an $O(1.6107^n)$ time algorithm solving the NONINDUCED BICLIQUE problem. Finally let us mention that all three algorithms can be modified such that they solve the corresponding counting problem.

2 The main result

In the remainder of this extended abstracted we present our exact exponential algorithm `bis` solving the BICOLORED INDEPENDENT SET problem.

Algorithm `bis`(G, k_1, k_2)
Input: Bicolored graph $G = (R, B, E)$, non-negative integers k_1 and k_2
Output: Boolean variable being TRUE iff G has an independent set I such that $|I \cap R| = k_1$ and $|I \cap B| = k_2$.

```
1   Choose a vertex v of maximum degree;
2   if d(v) ≤ 4 then
3       return bounded-pathwidth(G, k1, k2)
4   else
5       found ← bis(k1,k2,G\{v})
6       if v ∈ R then
7           return bis(k1-1,k2,G\N[v]) ∨ found
8       else
9           return bis(k1,k2-1,G\N[v]) ∨ found
```

Our algorithm combines branching and dynamic programming. As long as the maximum degree of the instance of the current subproblem is at least 5 the algorithm branches on a vertex v of maximum degree into two subproblems: in the first one (line 5) it discards v, i.e. v is definitely excluded from the independent set (and removed from the graph), in the second one (line 7 or 9) it takes v into the independent set (and removes v and all its neighbors from the graph). The branching stops when the maximum degree of the current graph G' is at most 4. Then a standard

algorithm solving MAXIMUM INDEPENDENT SET on graphs given with a path-decomposition of width ℓ in time $O(2^\ell)$ (see e.g. [4]) is used. It is easy to modify it such that it stores for any partial solution (often called characteristic) the number of red and the number of blue vertices in a corresponding independent set. The modified dynamic programming algorithm `bounded-pathwidth`(G', k_1, k_2) computes for input graph G' (of maximum degree at most 4) all pairs (r, b) such that G' has an independent set with r red and b blue vertices in time $O(2^\ell)$ and returns TRUE iff (k_1, k_2) is among those pairs. Note that while the branching part needs only polynomial space the dynamic programming algorithm requires exponential space.

The correctness of the algorithm follows from the fact that the branching verifies all possible cases and that the dynamic programming algorithm on any instance G' of a leaf of the search tree computes all possible pairs (b, r). But how to analyse the running time of such a combined algorithm?

2.1 Simple Analysis

The branching on a vertex v with a degree at least 5 gives the following recurrence. If we take v in the independent set, we can eliminate v and all its neighbours from the graph, and search an independent set with k_1-1 red vertices and k_2 blue vertices if $v \in R$, or an independent set with k_1 red vertices and k_2-1 blue vertices if $v \in B$. Hence $\Delta_{select} \geq 6$. If we don't take v in the independent set, we remove v from the graph and continue to search an independent set in $G \backslash \{v\}$ with k_1 red and k_2 blue vertices. Hence $\Delta_{discard} = 1$. Consequently the branching vector is at least $(1, 6)$ and thus the running time is $O(1, 2852^n)$.

Consider a leaf G' of the search tree of the branching algorithm, we use a dynamic programming algorithm on a pathwidth decomposition of G'. To analyse the running time we need an upper bound for the width of a path-decomposition of G'. Fomin et al. have shown in [3] that for all $\epsilon > 0$ there exists an n_ϵ such that for all graphs G of maximum degree 4 with $n > n_\epsilon$ holds: $\mathrm{pw}(G) \leq \frac{n_3}{6} + \frac{n_4}{3} + \epsilon n$, where n_i is the number of vertices of degree i. Furthermore such a path-decomposition of width $\frac{n_3}{6} + \frac{n_4}{3} + \epsilon n$ can be computed in polynomial time. Since G' has maximum degree 4 we obtain $\mathrm{pw}(G') \leq \frac{n}{3} + \epsilon n$. Hence the running time is $O(1, 26^{|V(G')|})$ for any leaf G' of the search tree.

To analyse the overall algorithm let us consider all leaves G' on h vertices. The number of such leaves of the search tree is $O^*(\alpha^{n-h})$ (here $\alpha = 1, 2852$) and the dynamic programming on such a leaf runs in time $O^*(\beta^h)$ (here $\beta = 1, 26$). Then as pointed out in [3] the overall running time is $\sum_{h=0}^{n} O^*(\alpha^{n-h} \cdot \beta^h) = O^*(max(\alpha, \beta)^n)$. Consequently our algorithm has running time $O(1, 2852^n)$.

2.2 Measure & Conquer analysis

A better upper bound on the worst-case running time can be achieved by using a more sophisticated measure for instances of subproblems in the branching algo-

rithm. In the simple analysis the measure of a graph is the number of its vertices. We use now the measure $\mu(G) = \sum_{0 \leq i \leq n}(n_i \cdot w_i)$, where n_i denotes the number of vertices of degree i and $0 \leq w_i \leq 1$ for all i.

A careful analysis of the recurrences for branching on a vertex of degree at least 5 in a graph of measure μ, which are all of the type $T(\mu) = T(\mu - \Delta_{select}) + T(\mu - \Delta_{discard})$, establishes a running time of $O(1.2691^n)$ for the branching algorithm using the weights given in the following table.

i	0	1	2	3	4	5	≥ 6
w_i	0	0.5386	0.8059	0.9233	0.9701	0.9999	1

For any μ, the number of leaves G' of the search tree such that $\mu(G') = \mu$ is $O(1.2691^{n-\mu})$. To upper bound the running time of the dynamic programming algorithm we need to upper bound the maximum pathwidth of any graph of maximum degree 4 of measure μ. Using the approach of [3,6] we show that the dynamic programming algorithm on a leaf G' of measure μ runs in time $O(1.2691^{\mu})$. Combining those bounds as in the simple analysis, one obtains that algorithm `bis` has running time $O(1.2691^n)$.

References

[1] J. AMILHASTRE, M.C. VILAREM, P. JANSSEN, Complexity of minimum biclique cover and minimum biclique decomposition for bipartite dominofree graphs, *Discrete Applied Mathematics* **86** (1998), 125–144.

[2] D. BINKELE-RAIBLE, H. FERNAU, S. GASPERS, M. LIEDLOFF, Exact exponential-time algorithms for finding bicliques, *Information Processing Letters* **110** (2010), 64–67.

[3] F.V. FOMIN, S. GASPERS, S. SAURABH, AND A.A. STEPANOV, On two techniques of combining branching and treewidth, *Algorithmica* **54** (2009), 181–207.

[4] F.V. FOMIN, D. KRATSCH, *Exact Exponential Algorithms*, Springer, Texts in Theoretical Computer Science, 2010.

[5] S. GASPERS, D. KRATSCH, M. LIEDLOFF, On independent sets and bicliques, *Proceedings of WG 2008*, LNCS 5344, Springer, Berlin, LNCS 5344, 2008, pp. 171–182.

[6] J.M.M. VAN ROOIJ, J. NEDERLOF, AND T.C. VAN DIJK, Inclusion/Exclusion Meets Measure and Conquer, *Proceedings of ESA 2009*, LNCS 5757, Springer, Berlin, 2009, pp. 554–565.

New Results for the Directed Network Diversion Problem

Christopher Cullenbine, [a] R. Kevin Wood, [b] Alexandra Newman [a]

[a] *Division of Economics and Business, Colorado School of Mines, Golden, Colorado, 80401 USA*
{ccullenb,newman}@mines.edu

[b] *Operations Research Department, Naval Postgraduate School, Monterey, California, 93943 USA*
kwood@nps.edu

Key words: network diversion, interdiction, s-t cut, NP-complete, vertex cover

1 Extended Abstract

The directed network-diversion problem (DND) seeks a minimum-weight, minimal s-t cut in a directed graph that contains a pre-specified "diversion edge," denoted here by e'. Although straightforward to describe, this is an interesting and difficult combinatorial optimization problem. We present an improved integer-programming formulation (IP) for DND, and new, more precise results on its computational complexity.

DND, and its undirected counterpart "UND," arise in intelligence-gathering and war-fighting scenarios. Curet [3] formulates DND as an IP and applies Lagrangian relaxation to solve that IP approximately; see also [2]. Erken [4] solves DND by enumerating near-minimum-weight s-t cuts. Yang and Park [9] create a tabu-search heuristic for DND, and Cho [1] applies an IP-based cut-enumeration algorithm to the formulation in [3].

2 A Stronger Integer-Programming Formulation

Section 3 presents new complexity results on DND, but the general problem is known to be NP-hard [3]. An IP formulation is therefore appropriate, and we begin with one; this also helps define the problems precisely.

Curet [3] describes an IP for DND, denoted here as **P1**, that consists of (a) a standard minimum-weight s-t cut model (i.e, the linear-programming dual to the maximum-flow problem), (b) a network-flow model that routes one unit of flow along a directed s-t path containing e', and (c) linking constraints that ensure that

the cut and path-flow intersect only at the diversion edge e'. If all edges in the minimum cut are interdicted (deleted or destroyed), except for the diversion edge, then all communication from s to t must pass through e' where it can be exploited. One might try to formulate DND without (b) and (c) and simply fix e' to be a forward edge of the minimum cut, but this may result in a non-minimal cut that disconnects s from t entirely.

P1's "single-commodity formulation" can be improved, however. In particular, the LP relaxation of **P1** allows flow around cycles containing e', which our "two-commodity formulation" **P2** precludes:

Indices and Index Sets:

$i \in V$	vertices; special vertices $s, t \in V, s \neq t$, are also defined
$e \in E$	directed edges; $e = (i, j)$ with tail vertex i and head vertex j
e', $\hat{E}$	$e' = (i', j')$ is the diversion edge, and $\hat{E} = E \setminus \{e'\}$
E_i^+ (E_i^-)	set of edges with tail (head) vertex i

Parameters:

w_e	interdiction weight for edge e, where $w_e > 0 \; \forall e \in \hat{E}$, $w_{e'} \equiv 0$
d_i ($\bar{d}_i$)	$d_s = 1$, $d_{i'} = -1$ and $d_i = 0$ for $i \in V - s - i'$ ($\bar{d}_{j'} = 1$, $\bar{d}_t = -1$ and $\bar{d}_i = 0$ for $i \in V - t - j'$)

Variables:

α_i	1 if vertex $i \in \bar{S}$ of an identified cutset $[S, \bar{S}]$, 0 otherwise
β_e	1 if edge e is interdicted (is a forward edge w.r.t. $[S, \bar{S}]$), 0 otherwise
y_e ($\bar{y}_e$)	1 if edge e lies on a path from s to i' (from j' to t), and 0 otherwise

Formulation (P2): (all variables in $\{0, 1\}$)

$$z^* = \min \sum_{e \in E} w_e \beta_e \tag{1}$$

$$\text{s.t. } \alpha_i - \alpha_j + \beta_e \geq 0 \;\; \forall \, e \in E \tag{2}$$

$$\sum_{e \in E_i^+} y_e - \sum_{e \in E_i^-} y_e = d_i \;\; \forall \, i \in V \tag{3}$$

$$\sum_{e \in E_i^+} \bar{y}_e - \sum_{e \in E_i^-} \bar{y}_e = \bar{d}_i \;\; \forall \, i \in V \tag{4}$$

$$y_e + \bar{y}_e + \beta_e \leq 1 \;\; \forall \, e \in \hat{E} \tag{5}$$

$$\sum_{e \in E_i^+} y_e + \alpha_i \leq 1 \;\; \forall \, i \in V \setminus \{t\} \tag{6}$$

$$\sum_{e \in E_i^+} \bar{y}_e - \alpha_i \leq 0 \;\; \forall \, i \in V \setminus \{s\} \tag{7}$$

$$\beta_{e'} \equiv 1, \alpha_s \equiv 0, \alpha_{i'} \equiv 0, \alpha_{j'} \equiv 1, \alpha_t \equiv 1 \tag{8}$$

The objective function (1) minimizes the total weight of the interdicted edges. (As in [3], we find that a small, symmetry-breaking path-length penalty, not shown, can also be helpful here.) Constraints (2) coupled with fixed variables in (8) require the identification of an s-t cut that contains e'. Constraints (3) require that one unit of flow be routed from s to i', while constraints (4) require that one unit of flow be routed from j' to t. Constraints (5) require, with mutual exclusivity, that an edge

$e \in \hat{E}$ may define (a) part of a "flow-preserving path" from s to i', (b) part of a flow-preserving path from j' to t, (c) a forward edge of the s-t cut, or (d) none of those three possibilities. Constraints (6) imply that flow on variables y_e, from s to i', can pass only through vertices in S, and constraints (7) are analogous for flow on variables $\bar{y}_e$ in $\bar{S}$. Constraints (6) and (7) are not strictly necessary, but help to tighten the formulation in practice. The following proposition is clear.

Theorem 2.1 *Formulation* **P2** *for DND is at least as strong as* **P1**.

Computational tests using CPLEX 12.1 on a fast workstation indicate that **P2** is much superior to **P1**. For instance, one planar model instance based on a rectangular grid with $|V|$ = 10,000, $|E|$ = 39,800 and $w_e \in \{1, \ldots, 5\}$ for $e \in \hat{E}$ solves in 14 seconds with **P2**, but does not solve in under 3,600 seconds with **P1**. Another instance of **P2** solves in 92 seconds while no integer solution for **P1** is found in 3,600 seconds. On a set of 10 test problems, **P2** closes the average duality gap observed for **P1** by 18%, with a range of 3%-52%.

3 Computational Complexity

This section describes new results on the theoretical computational complexity of DND, and to a lesser degree on UND. Proofs are omitted. We begin by noting that Curet [3] actually defines DND with respect to a *set* of diversion edges E'. That is, he seeks to interdict a minimum-weight set of edges such that all s-t paths must traverse *at least* one edge of E'.

Theorem 3.1 *DND defined for a set of diversion edges E' is polynomially equivalent to DND defined for a single diversion edge e'.*

Thus, no generality is lost by studying DND having only a single diversion edge; in fact, the improved IP formulation **P2** depends on this assumption.

Proposition 3.2 *(See [3].) DND is NP-complete.*

Curet [3] does not formally prove Proposition 3.2, but argues that DND is feasible if and only if there exists a simple directed s-t path containing e', and then applies the fact that this "edge-restricted feasible-path problem" is an NP-complete instance of the directed subgraph homeomorphism problem [5]. (See also [9].) The analogous path problem for undirected graphs is solvable in polynomial time, so this result implies nothing about UND's complexity.

Using a transformation from the vertex-cover problem [6], we can prove the following stronger result and a corollary:

Theorem 3.3 *DND is strongly NP-complete even when the diversion edge is incident out of s or into t.*

The proof involves interdicting directed edges, of course, but undirected edges suffice in the proof if diversion through e' is accomplished by interdicting vertices rather than edges. Thus, the following corollary results.

Corollary 1 *The vertex-interdiction version of both DND and UND is strongly NP-complete.*

We are still investigating general instances of UND, but have shown that special cases of both UND and DND are solvable in polynomial time.

Theorem 3.4 *Both UND and DND are solvable in polynomial time in s-t planar graphs.*

Theorem 3.4 is shown by (a) creating a single-commodity flow model in the s-t planar dual graph G^* that moves one unit of "dual flow" from each of the dual vertices in faces incident to e' to a dual vertex in the outer face of G, and (b) places a capacity of one on the flow through any dual vertex.

The related maximum-flow network-interdiction problem ([8], [7]) can be solved in polynomial or pseudo-polynomial time in general directed or undirected planar graphs, not just s-t planar ones. The solution algorithm involves finding an odd-parity cycle in G^* with respect to a simple s-t path in G. cycle in G^* crosses the path in G an odd number of times.) Interestingly, this technique seems to apply to neither UND nor DND.

References

[1] Cho, D., 2009, "An optimization algorithm for the network diversion problem using combinatorial Benders' cut," M.S. Thesis, Dept. of Industrial Engineering, Korea Advanced Institute of Science and Technology, Daejon, Korea.

[2] Cintron-Arias, A., Curet, N., Denogean, L., Ellis, R., Gonzalez, C., Oruganti, S. and Quillen, P., 2000, "A network diversion vulnerability problem," *IMA Preprint*.

[3] Curet, N.D., 2001, "The network diversion problem," *Military Operations Research*, **6**, 35–44.

[4] Erken, O., 2002, "A branch and bound algorithm for the network diversion problem," Master's thesis, Operations Research Department, Naval Postgraduate School, Monterey, California.

[5] Fortune, S., Hopcroft, J. and Wyllie, J., 1980, "The directed subgraph homeomorphism problem," *Theoretical Computer Science*, **10**, 111–121.

[6] Karp, R., 1972, "Reducibility among combinatorial problems," In *Complexity of Computer Computations*, R. Miller and J. Thatcher, eds., Plenum Press, 85–103.

[7] Phillips, C.A., 1993, "The network inhibition problem," In *STOCâ 93: Proc. of the 25th Annual ACM Symposium on Theory of Computing*, ACM Press, New York, NY, 776–785.

[8] Wood, R.K., 1993, "Deterministic network interdiction," *Mathematical and Computer Modelling*, **17**, 1–18.

[9] Yang, H. and Park, S., 2004, "A tabu search algorithm for the network diversion problem," *J. of the Military Operations Research Society of Korea*, **30**, 30–47.

A New Feasibility Pump-Like Heuristic for Mixed Integer Problems

M. De Santis, S. Lucidi, F. Rinaldi

Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma
Via Ariosto, 25 - 00185 Roma - Italy
{mdesantis,lucidi,rinaldi}@dis.uniroma1.it

Key words: Mixed integer programming, Concave penalty functions, Frank-Wolfe algorithm, Feasibility Pump.

Many real-world problems can be modeled as Mixed Integer Programming (MIP) problems, namely as minimization problems where some (or all) of the variables only assume integer values. Finding a first feasible solution quickly is crucial for solving this class of problems. In fact, many local-search approaches for MIP problems such as Local Branching [8], guide dives and RINS [6] can be used only if a feasible solution is available.
In the literature, several heuristics methods for finding a first feasible solution for a MIP problem have been proposed. Recently, Fischetti, Glover and Lodi [7] proposed a new heuristic, the well-known Feasibility Pump, that turned out to be very useful in finding a first feasible solution even when dealing with hard MIP instances. The FP heuristic is implemented in various MIP solvers such as BONMIN [5].
The basic idea of the FP is that of generating two sequences of points $\{\bar{x}^k\}$ and $\{\tilde{x}^k\}$ such that $\bar{x}^k$ is LP-feasible, but may not be integer feasible, and $\tilde{x}^k$ is integer, but not necessarily LP-feasible. To be more specific the algorithm starts with a solution of the LP relaxation $\bar{x}^0$ and sets $\tilde{x}^0$ equal to the rounding of $\bar{x}^0$. Then, at each iteration $\bar{x}^{k+1}$ is chosen as the nearest LP-feasible point in ℓ_1-norm to $\tilde{x}^k$, and $\tilde{x}^{k+1}$ is obtained as the rounding of $\bar{x}^{k+1}$. The aim of the algorithm is to reduce at each iteration the distance between the points of the two sequences, until the two points are the same and an integer feasible solution is found. Unfortunately, it can happen that the distance between $\bar{x}^{k+1}$ and $\tilde{x}^k$ is greater than zero and $\tilde{x}^{k+1} = \tilde{x}^k$, and the strategy can stall. In order to overcome this drawback, random perturbations and restart procedures are performed.
As the algorithm has proved to be effective in practice, various papers devoted to

its further improvements have been developed. Fischetti, Bertacco and Lodi [3] extended the ideas on which the FP is based in two different directions: handling MIP problems with both 0-1 and integer variables, and exploiting the FP information to drive a subsequent enumeration phase. In [1], in order to improve the quality of the feasible solution found, Achterberg and Berthold consider an alternative distance function which takes into account the original objective function. In [9], Fischetti and Salvagnin proposed a new rounding heuristic based on a diving-like procedure and constraint propagation. Here we report a brief outline of the Feasibility Pump basic scheme:

The Feasibility Pump (FP) - basic version

Initialization: Set $k = 0$, let $\bar{x}^0 := \arg\min\{c^T x : Ax \geq b\}$
While (not stopping condition) **do**
  Step 1 If ($\bar{x}^k$ is integer) return $\bar{x}^k$
  Step 2 Compute $\tilde{x}^k = round(\bar{x}^k)$
  Step 3 If (cycle detected) $perturb(\tilde{x}^k)$
  Step 4 Compute $\bar{x}^{k+1} := \arg\min\{\Delta(x, \tilde{x}^k) : Ax \geq b\}$
  Step 5 Update $k = k + 1$
End While

An interesting interpretation of the FP has been given by J.Eckstein and M.Nediak in [4]. In this work they noticed that the FP heuristic may be seen as a form of Frank-Wolfe procedure applied to a nonsmooth merit function which penalizes the violation of the 0-1 constraints:

$$\begin{aligned} \min \quad & \psi(x) \\ \text{s.t.} \quad & Ax \geq b \\ & 0 \leq x_j \leq 1 \ \forall j \in I \end{aligned}$$

where

$$\psi(x) = \sum_{i \in I} \phi(x_i) = \sum_{i \in I} \min\{x_i, 1 - x_i\}.$$

In this paper, taking inspiration from [9], we propose the following merit functions:
Logarithmic function

$$\phi(t) = \min\Big\{\lambda_{1,1}(t + \varepsilon), \lambda_{1,1}[(1 - t) + \varepsilon]\Big\};$$

Hyperbolic function

$$\phi(t) = \min\Big\{ -(t + \varepsilon)^{-p}, -[(1 - t) + \varepsilon]^{-p}\Big\};$$

Concave function

$$\phi(t) = \min\Big\{1 - \exp(-\alpha t), 1 - \exp(-\alpha(1 - t))\Big\};$$

Logistic function

$$\phi(t) = \min\left\{[1+\exp(-\alpha t)]^{-1}, [1+\exp(-\alpha(1-t))]^{-1}\right\}.$$

The use of these merit functions leads to a new FP scheme in which the ℓ_1-norm used for calculating the next LP-feasible point is replaced with a "weighted" ℓ_1-norm of the form

$$\Delta_W(x,\tilde{x}) = \sum_{j\in I} w_j|x_j - \tilde{x}_j| = \|W(x-\tilde{x})\|_1,$$

where

$$W = diag(w_1,\ldots,w_n)$$

and

$$w_j^k = |g_j^k|$$

with $g^k \in \partial\psi(\bar{x}^k)$, where $\partial\psi(x)$ belongs to the supergradient of the function ψ.
The main feature of the method is the use of an infeasibility measure that

- tries to discourage the optimal solution of the relaxation from being far from $\tilde{x}$ (similarly to the original FP algorithm);
- takes into account, in some way, the information carried by the LP-feasible points obtained at the previous iterations of the algorithm for speeding up the convergence to integer feasible points.

Here we report an outline of the algorithm:

Reweighted Feasibility Pump (RFP) - basic version

Initialization: Set $k = 0$, let $\bar{x}^0 := \arg\min\{c^T x : Ax \geq b\}$
While (not stopping condition) **do**
  Step 1 If ($\bar{x}^k$ is integer) return $\bar{x}^k$
  Step 2 Compute $\tilde{x}^k = round(\bar{x}^k)$
  Step 3 If (cycle detected) $perturb(\tilde{x}^k)$
  Step 4 Compute $\bar{x}^{k+1} := \arg\min\{\|W^k(x-\tilde{x}^k)\|_1 : Ax \geq b\}$
  Step 5 Update $k = k+1$
End While

This approach is extended to the case of general MIP problems where the use of suitable nonsmooth merit functions allows us to penalize the violation of the integer constraints.
Finally, a reported computational experience seems to indicate that the use of these new merit functions can improve the FP efficiency.

References

[1] T. ACHTERBERG AND T. BERTHOLD. *Improving the feasibility pump*. Discrete Optimization, 4, pp 77–86, 2007.

[2] E. BALAS, S. CERIA, M. DAWANDE, F. MARGOT, G. PATAKI. *OCTANE: A new heuristic for pure 0-1 programs*. Operations Research, 49(2), pp 207–225, 2001.

[3] L. BERTACCO, M. FISCHETTI, AND A. LODI. *A feasibility pump heuristic for general mixed-integer problems*. Discrete Optimization, 4, pp 63–76, 2007.

[4] J. ECKSTEIN AND M. NEDIAK. *Pivot, cut, and dive: a heuristic for 0-1 mixed integer programming*. Journal of Heuristics, 13, pp 471–503, 2007.

[5] P. BONAMI, L.T. BIEGLER, A.R. CONN, G. CORNUEJOLS, I.E. GROSSMANN, C.D. LAIRD, J. LEE, A. LODI, F. MARGOT, N.SAWAYA AND A. WAECHTER. *An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs*. IBM Research Report RC23771, To appear in Discrete Optimization, in press.

[6] E. DANNA, E. ROTHBERG, C. LE PAPE. *Exploring relation induced neighborhoods to improve MIP solution*. Mathematical Programming 102, 1, pp 71–90, 2005.

[7] M. FISCHETTI, F. GLOVER, A. LODI. *The Feasibility Pump*. Mathematical Programming, 104, pp 91–104, 2005.

[8] M. FISCHETTI, A. LODI. *Local Branching*. Mathematical Programming, 98(1-3), pp 23–47, 2003.

[9] M. FISCHETTI, D. SALVAGNIN. *Feasibility pump 2.0*. Mathematical Programming Computation, 1, pp 201–222, 2009.

[10] F. GLOVER, M. LAGUNA. *General purpose heuristics for integer programming - part I*. Journal of Heuristics, 3, 1997.

[11] F. GLOVER, M. LAGUNA. *General purpose heuristics for integer programming - part II*. Journal of Heuristics, 3, 1997.

[12] S. LUCIDI, F. RINALDI. *Exact penalty functions for nonlinear integer programming problems*. Journal of Optimization Theory and Applications Vol.145(3), 479-488, 2010.

Continuous Reformulations for Zero-one Programming Problems

M. De Santis, S. Lucidi, F. Rinaldi

Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma
Via Ariosto, 25 - 00185 Roma - Italy
{mdesantis,lucidi,rinaldi}@dis.uniroma1.it

Key words: Zero-one programming, Concave functions, Continuous Programming.

Several important problems arising in operations research, graph theory and mathematical programming are formulated as 0-1 programming problems:

$$\begin{aligned} &\min c^T x \\ &x \in C \\ &x \in \{0,1\}^n \end{aligned}$$

where C is a convex set.
A possible approach for solving this class of problems can be that of transforming the original problem into an equivalent continuous problem. Various transformations have been proposed in the literature (see e.g. [1]-[3], [10]-[12]). A well-known continuous reformulation comes out by relaxing the integrality constraints on the variables and by adding a penalty term $\psi(x)$ to the objective function:

$$\begin{aligned} &\min c^T x + \varepsilon\, \psi(x) \\ &x \in C \\ &x \in [0,1]^n \end{aligned}$$

where ε is a positive parameter. This approach has been first introduced by Raghavachari [13] to solve 0-1 linear programming problems, then extended by Giannessi and Niccolucci [6] to general nonlinear integer programming problems. Many other reformulations related to the one by Raghavachari have been proposed in the literature (see e.g. [4],[7]-[9], [14] and [16]).
An important theoretical result is that, under weak assumptions, there exists a threshold value $\bar{\varepsilon} > 0$ of the penalty parameter ε such that, for any $\varepsilon \in (0, \bar{\varepsilon}]$, any solution of the continuous problem is also a solution of the related binary problem

(see e.g. [9]).
In this paper, we consider convex sets C satisfying some specific assumptions and we show that:

- a value $\bar{\varepsilon} > 0$ that guarantees the equivalence between the original integer problem and its continuous reformulation can be explicitly calculated;

- a different continuous reformulation for solving 0-1 programming problems (obtained by relaxing the integrality constraints on the variables and by making a nonlinear transformation of the variables in the objective function) can be defined.

In particular, the new continuous reformulation assumes the following form:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in C \\ & 0 \leq x \leq e \end{aligned}$$

where

$$f(x) = \sum_{\substack{i=1 \\ c_i>0}}^{n} c_i \, g_i(x_i) + \sum_{\substack{i=1 \\ c_i<0}}^{n} |c_i| \, g_i(x_i) + \sum_{\substack{i=1 \\ c_i=0}}^{n} g_i(x_i) - \sum_{\substack{i=1 \\ c_i<0}}^{n} |c_i|,$$

and $g_i : [0,1] \rightarrow \mathbf{R}$, $i = 1, \ldots, n$ are continuous concave functions.
Starting from the ideas developed in [5], we propose various examples of functions g_i. First of all, we denote

$$\tilde{c} = \frac{(n+1) \, \max_i |c_i| + \sum_i |c_i|}{\min_i |c_i|}$$

and we define two values x_l and x_u as follows:

$$x_l = \inf_{x \in S} l(x),$$

$$x_u = \sup_{x \in S} u(x),$$

where $S \subset [0,1]^n$ is the set of extreme points of the feasible set of the continuous reformulation and

$$l(x) = \begin{cases} \min\{x_i : \, i = 1, \ldots, n; \, x_i \neq 0\} & \text{if } x \neq 0 \\ 1 & \text{if } x = 0; \end{cases} \tag{1}$$

$$u(x) = \begin{cases} \max\{x_i : \, i = 1, \ldots, n; \, x_i \neq 1\} & \text{if } x \neq e \\ 0 & \text{if } x = e. \end{cases} \tag{2}$$

Now we can define the functions g_i to be used in the continuous reformulation:

Case $c_i > 0$:

$$g_i(t) = \min\Big\{\gamma_{1_+}\,\phi(t), 1 + \gamma_{2_+}\,\phi(1-t)\Big\}, \tag{3}$$

$$\gamma_{1_+} > \frac{\tilde{c}}{\phi(x_l)}\,, \qquad \gamma_{2_+} > \frac{\tilde{c}-1}{\phi(1-x_u)}; \tag{4}$$

Case $c_i < 0$:

$$g_i(t) = \min\Big\{1 + \gamma_{1_-}\,\phi(t), \gamma_{2_-}\,\phi(1-t)\Big\}, \tag{5}$$

$$\gamma_{1_-} > \frac{\tilde{c}-1}{\phi(x_l)}\,, \qquad \gamma_{2_-} > \frac{\tilde{c}}{\phi(1-x_u)}; \tag{6}$$

Case $c_i = 0$:

$$g_i(t) = \min\Big\{\gamma_0\,\phi(t), \gamma_0\,\phi(1-t)\Big\}, \tag{7}$$

$$\gamma_0 > \max\left\{\frac{n\max_i|c_i| + \sum_i|c_i|}{\phi(x_l)}, \frac{n\max_i|c_i| + \sum_i|c_i|}{\phi(1-x_u)}\right\}; \tag{8}$$

with $\phi : \mathbf{R} \to \mathbf{R}$ a strictly increasing, concave function such that $\phi(0) = 0$.

By choosing functions g_i equal to (3), (5) and (7), we can prove that a binary problem and the continuous reformulation are equivalent.

Finally, we report a numerical comparison between our approach and the one based on the exact penalty reformulation.

References

[1] ABELLO, J., BUTENKO, S., PARDALOS, P.M., RESENDE, M.,*Finding independent sets in a graph using continuous multivariable polynomial formulations*. J. Glob. Optim. 21, 111-137, 2001.

[2] BALASUNDARAM, B., BUTENKO, S.,*Constructing test functions for global optimization using continuous formulations of graph problems*. Optim. Methods Softw. 20, 439-452, 2005.

[3] HORST, R., PARDALOS, P.M., THOAI, N.V.,*Introduction to Global Optimization* 2nd edn. Kluwer, Dordrecht, 2000.

[4] BORCHARDT M., *An Exact Penalty Approach for Solving a Class of Minimization Problems with Boolean Variables*. Optimization. 19(6), pp. 829-838, 1988.

[5] M. DE SANTIS, S. LUCIDI, F. RINALDI. *New Concave Penalty functions for improving the feasibility pump*. Department of Computer and System Sciences Antonio Ruberti Technical Reports, Vol.2(10), 2010.

[6] GIANNESSI F., NICCOLUCCI F., *Connections between nonlinear and integer programming problems*. Symposia Mathematica, Academic Press, New York , Vol. 19, pp. 161-176, 1976.

[7] KALANTARI B., ROSEN J.B., *Penalty Formulation for Zero-One Integer Equivalent Problem*. Mathematical Programming, Vol. 24, pp. 229-232, 1982.

[8] KALANTARI B., ROSEN J.B., *Penalty Formulation for Zero-One Nonlinear Programming*. Discrete Applied Mathematics, Vol. 16(2), pp. 179-182, 1987.

[9] S. LUCIDI, F. RINALDI. *Exact penalty functions for nonlinear integer programming problems*. J Optim Theory Appl Vol. 145, pp. 479-488, 2010.

[10] MANGASARIAN, O.L., *Knapsack Feasibility as an Absolute Value Equation Solvable by Successive Linear Programming*.Optim. Lett. Vol. 3(2), 2009.

[11] MURRAY W., NG K. M., *An algorithm for nonlinear optimization problems with binary variables*. Computational Optimization and Applications, Vol. 47(2), 257-288, 2010.

[12] PARDALOS P. M., PROKOPYEV O. A., BUSYGIN S., *Continuous Approaches for Solving Discrete Optimization Problems*. Handbook on Modelling for Discrete Optimization, Springer US, Vol. 88, pp. 39-60, 2006.

[13] RAGHAVACHARI M., *On Connections Between Zero-One Integer Programming and Concave Programming Under Linear Constraints*, Operation Research Vol. 17(4), pp. 680-684, 1969.

[14] RINALDI F. *New results on the equivalence between zero-one programming and continuous concave programming*, Optimization Letters, Vol. 3(3), 377–386, 2009.

[15] ROCKAFELLAR T., *Convex Analysis*, Princeton University Press, 1970.

[16] ZHU W. X., *Penalty Parameter for Linearly Constrained 0-1 Quadratic Programming*, Journal of Optimization Theory and Applications, Vol. 116(1), pp. 229-239, 2003.

The maximum labeled clique problem

Paolo Dell'Olmo, [a] Raffaele Cerulli, [b] Francesco Carrabs [b]

[a] *Department of Statistic Sciences, Sapienza University of Rome, Italy.*
paolo.dellolmo@uniroma1.it

[b] *Department of Mathematics, University of Salerno, Italy.*
{raffaele,fcarrabs}@unisa.it

Key words: Clique, Edge Labeled Graph, Budget Constraint

1 Introduction

The maximum clique problem may be called one of the most important combinatorial optimization problem, with application in many real world situations. In this paper we study a variant of the maximum clique problem, namely, the *Maximum Labeled Clique problem* (MLC). Given a graph G with a label (color) assigned to each edge (not necessarily properly) we look for a clique of G as large as possible but with the minimum number of different edge labels. Moreover, the number of usable labels is limited by a fixed constant (budget) as will be described later in the mathematical formulation.
The problem has several applications, among others, in telecommunication and social networks. For example, let us consider a telecommunication network where the connections belong to different companies, each one identified by a different label. Our aim is to localize the maximum number of nodes connected with each other where to place mirroring servers. These servers share the same information and the direct connection with each other guarantees that when a server falls down the others remain synchronized. Since the use of connections have to be paid to the owner company, a second aim is to minimize the number of labels used for the connections. However, there is even a budget that bounds the number of different labels usable. Then our problem is to find a maximum clique with the minimum number of labels without exceeding the budget available. A further application may be found in social network environment. Examples of application of graph analysis to social networks are given [14] where nodes represent persons and undirected edges represent a generic realtionship among individuals (communication, friendship, working cooperation and so on). In this context, cliques represent groups of individuals mutually connected by the mentioned relation. The label of edges may represent the specific topic or argument which motivated the communication. Hence, a (maximum) clique with a small number of labels corresponds to the largest group (mutually connected) with small etherogeneity on the excanged subjects, i.e. largest

"most focused" group on a single idea. The study of such structures on the web and organizational communities (e.g. large corporate groups) is also motivated by the need of identifing new business or management opportunities by analysing the people behaviour by means of the corresponding graphs.

In this paper we provide a formulation of the problem, analyze its complexity and introduce a heuristic approach that computes feasible solutions for the problem exploiting the information concerning edge colors and budget constraint. The exact solutions provided by model are used to verify the efficacy of our heuristic. Moreover, the performances of algorithms are experimentally evaluated on a wide set of instances.

A number of problems with labeled graphs have been studied in recent years. One such problem is the Minimum Label Spanning Tree (MLST) Problem, in which a spanning tree with the minimum number of labels is sought over a labeled graph. The problem was introduced in [4,5]. Heuristics for the problem have been discussed, for instance, in [10]. Other works involving classic combinatorial optimization problems defined on labeled graphs include the Minimum Label Steiner Tree Problem discussed in [11], the Minimum Label Generalized Forest Problem discussed in [4], the Minimum Label Path Problem studied in [12] and the Labeled Maximum Matching Problem in [13]. At the best of our knowledge, the MLC problem has not been studied before.

2 Definitions and Notations

Let $G = (V, E, L)$ a labeled, connected and undirected graph where V denotes the set of n vertices, E the set of m edges and L the set of labels associated to the edges. Let us define the function $c : E \rightarrow L$ that given an edge (i, j) returns its color. Given a subset $X \subseteq V$ of vertices, we define $\bar{X} = V \setminus X$ its complementary set. Moreover, we denote by $G[X]$ the subgraph of G induced by the set of vertices X. Formally, $G[X] = (X, E[X], L[X])$ where $E[X] = \{(i, j) \in E : i, j \in X\}$ and $L[X] = \{l \in L : \exists (i, j) \in E[X] \; with \; c(i, j) = l\}$. The Maximum Labeled Clique problems (MLC) consists to find a maximal complete subgraph $G' = (V', E[V'], L[V'])$ of G with the minimum number of different colors and such that $|L[V']| \leq b$, with $b \in Z^+$ and $b \leq L$. The MLC is a generalization of the Maximum Clique (MC) problem. Indeed, this last problem can be seen as a MLC where $|L| = 1$ and $b \geq 1$. However, it is interesting to notice that the polynomial cases of MC not necessary are polynomial for the MLC. For instance, the MC problem is solvable in polynomial time on perfect graphs and in particular on these graphs there are polynomial algorithms that enumerate all the maximal cliques. This means that in polynomial time it is possible to find the maximum clique with the minimum number of colors by these enumerating algorithms. However, because of constrains on budget b, this solution could be infeasible for MLC.

In the following are introduced the formulation and a brief description of heuristic. The set of variables for the MLC formulation are:

- binary variable x_i for each $i \in V$ that assumes value 1 if the vertex i is selected and 0 otherwise;
- binary variable y_l for each $l \in L$ that assumes value 1 if there is at least an edge which color is l in G' and 0 otherwise.

Moreover, let $a_{ij}^l = 1$ if $c(i,j) = l$. Then, the formulation of MLC is the following:

$$\max \quad |L| \sum_{v \in V} x_v - \sum_{l \in L} y_l \tag{1}$$

$$\sum_{l \in L} y_l \leq b \tag{2}$$

$$x_i + x_j \leq 1 \qquad \forall (i,j) \in \bar{E} \tag{3}$$

$$a_{ij}^l (x_i + x_j) \leq y_l + 1 \qquad \forall (i,j) \in E, l \in L \tag{4}$$

$$x_i \in \{0,1\} \qquad \forall i \in V \tag{5}$$

$$y_l \in \{0,1\} \qquad \forall l \in L \tag{6}$$

The objective function (1) of the formulation shown above requires to maximize the difference between the number of nodes selected and the number of colors inside the clique inducted by these nodes. The first sum is multiplied for number of colors because the primary objective of the problem is to maximize the number of nodes selected. This means that we prefer a clique with k nodes and $p > 1$ colors instead that a clique with $k-1$ nodes and 1 color. Constraint (2) guarantee that the number of colors inside the clique selected do not exceeed the budget b. Constraints (3) ensure that two nodes x_i and x_j can be selected only if exists the edge (x_i, x_j) in G. Constraints (4) ensure that y_l is equal to 1 if and only if in the clique there is an edge (x_i, x_j) such as $c(x_i, x_j) = l$. Finally, constraints (5) and (6) require that x_i and y_l are binary variables.

Regarding the heuristic approach it is derived from most efficient heuristics proposed in literature [1,2,3,6,7,8,9] for the MC problem. The heuristic is divided in two phases: expansion and improvement.
Into the first phase the aim is to maximize the number of vertices inside the clique obviously respecting the budget constraints. To this end, the algorithm sorts the vertices taking in account for each vertex: its degree, the degree of its neighbor and the number of colors incident to it. Later, it inserts the vertices inside the current clique (initially empty) according to previous sorting and checking that the new subgraph produced is again a clique. Once completed the insertion of vertices, it is carried out a local search to increase the cardinality of current clique.
Obtained the maximal clique from first phase, in the second phase is applied local search to reduce the number of colors present in it. During the local search the cardinality of clique never changes. However, alternating phases of plateau (same cardinality, same colors but at least a different node) and reduction (same cardinality,less colors) are carried out.

References

[1] R. Battiti and M. Protasi *Reactive Local Search for the Maximum Clique Problem*. Algorithmica, 29, 610-637, (2001).

[2] R.Battiti and F. Mascia *Reactive Local Search for the Maximum Clique Problem: a new implementation*. Technical Report, (2007).

[3] S. Busygin *A new trust region technique for the maximum clique problem*. Internal report, http://www.busygin.dp.ua.

[4] H. Broersma and X. Li. *Spanning trees with many or few colors in edge-colored graphs*. Discussiones Mathematicae Graph Theory, 17(2), (1997), 259-269.

[5] R.S. Chang and S.J. Leu. *The minimum labeling spanning trees*. Information Processing Letters, 63(5), (1997), 277-282.

[6] W. Pullan, H.H. Hoos *Dynamic Local Search for the Maximum Clique Problem*. Journal of Artificial Intelligence Research, 25, (2006), 159-185

[7] A.Grosso, M. Locatelli, F.D. Croce, *Combining swaps and node weights in an adaptive greedy approach for the maximum clique problem*. Journal of Heuristics, 10, (2004), 135-152.

[8] K. Katayama, A. Hamamoto, H. Narihisa *An effective local search for the maximum clique problem*, Information Processing Letters, 95 Issue 5, (2005)

[9] S. Fenet and C. Solnon *Searching for Maximum Cliques with Ant Colony Optimization* Lecture Notes in Computer Science, Volume 2611/2003, 291-302, (2003)

[10] R. Cerulli, A. Fink, M. Gentili, and S. Vos. *Metaheuristics comparison for the minimum labelling spanning tree problem*. In B. Golden, S. Raghavan, and E. Wasil, editors, The Next Wave in Computing, Optimization, and Decision Technologies, 93-106. Springer, 2005.

[11] R. Cerulli, A. Fink, M. Gentili, and S. Voss. *Extensions of the minimum labelling spanning tree problem*. Journal of Telecommunication and Information Technology, 4, 39-45, (2006).

[12] R.D. Carr, S. Doddi, G. Konjedov, and M. Marathe. *On the red-blue set cover problem*. In 11th ACM-SIAM Symposium on Discrete Algorithms, 345-353, (2000).

[13] F. Carrabs, R. Cerulli, M. Gentili. *The Labeled Maximum Matching Problem*. Computers & Operations Research, 36(6), 1859-1871, (2009).

[14] S. Wasserman, K. Faust. *Social network analysis: methods and applications*. Cambridge University Press, (1999).

Pickup and Delivery Problem with Incompatibility Constraints

Pablo Factorovich, Isabel Méndez-Díaz, Paula Zabala

Departamento de Computación - FCEyN, Universidad de Buenos Aires and CONICET
{pfactoro,imendez,pzabala}@dc.uba.ar

Key words: pickup and delivery problem, incompatibilities, routing

1 Introduction

In the Asymmetric Single Vehicle One-to-One Pickup and Delivery Problem Without Time Windows nor Capacity Constraints (PDP herein), there is a set of requests and a vehicle must follow a route that starts at a source depot,
accomplishes all the requests and finishes at an end depot. Each request consists in a transportation task from a pickup node to a delivery node, being each of those pairs potentially different for each task. As a consequence of this, the activities establish precedences between the nodes. The goal of the problem is to find a route of minimum cost satisfying the collection of precedences given by the requests.

Pickup and delivery problems have a large set of applications in routing problems, but we can emphasize their use in door-to-door courier service and transportation for handicapped people.

As far as we know, there are no articles on this particular version of the problem; however, for the symmetric version we found few articles (among which, the recent [2]). For a survey on similar problems see [1].

A different scenario arises for the transportation of certain incompatible goods like food and detergents or hazardous materials. This situation is considered in legislation all around the world. The proposed solutions for routing incompatible goods use different trucks or specialized vehicles with compartments to serve nodes in conflict. The former is inefficient and the latter requires more expensive vehicles (and it could still be dangerous in case of accident).

In this work, we propose to use the same vehicle for accomplish the tasks and simply forbid routes including incompatible goods on the vehicle at the same time. We will name this problem *Pickup and Delivery Problem with Incompatibilities*

⋆ This work is partially supported by UBACyT X143 and PICT 1600
[1] Work partially supported by YPF foundation and EADIC (Erasmus Mundus)

(PDPwI herein). We were not able to find this problem in the scientific literature (we just were able to find a patent [3]).

If we consider instances where every pair of goods is incompatible, we have an ATSP instance since every pick have to be followed by its delivery in a feasible solution. In addition, we have also proved that PDPwI generalizes Vertex Coloring problem (VC). Since PDP, ATSP and VC can be polynomially reduced to our problem, PDPwI is NP-hard.

2 Mathematical formulations

First, we give a formal definition of PDP. Let n be the number of requests and $G = (V_G, E_G)$ a directed graph where $V_G = \{0, ..., 2n+1\}$ is the node set and E_G the arc set, each arc having a positive cost. Nodes $2n$ and $2n+1$ represent source and target depots respectively. The sets $P = \{0, ..., n-1\}$ and $D = \{n, ..., 2n-1\}$ denote the pickup and delivery sets respectively. Requests are pairs $(i, i+n)$, where node $i \in P$ and $i+n \in D$.

The goal of the problem is to find the minimum cost Hamiltonian path among those starting from $2n$, finishing at $2n+1$ and following the pickup-delivery precedences.

In PDPwI we add an incompatibility undirected graph $I = (V_I, E_I)$ to the instances, where $V_I = P$ and $(i, j) \in E_I$ iff goods picked at i and j cannot be on the vehicle at the same time.

We have implemented 4 models for this problem. The first one (*2iFM*) is a variation of the model used in [2] obtained by adding the following constraints to forbid paths with incompatibilities:

$$X(T) \leq |T| - 2 \quad \forall T \subset P \cup D : \exists i, j \in T \cap P : (i, j) \in E_I, (i+n), (j+n) \notin T$$

The second and third models (*PV1* and *PV2*) use $b_{i,j}$ binary variables which take value 1 iff node i is before node j in the solution and are based on models proposed in [5] and [4] respectively. Clearly we need to set $b_{i,i+n} = 1$ in a pickup and delivery context and we have to add to the model the following equalities to account the incompatibilities

$$\begin{aligned} b_{i,j} &= b_{i+n,j} \quad && \forall (i, j) \in E_I \\ b_{i,j} &= b_{i+n,j+n} \quad && \forall (i, j) \in E_I, i < j \end{aligned}$$

Finally, we have developed a model based on new binary variables $z_{i,j}$, being 1 iff $j \in \{i, i+n\}$ or j is in the path from i to its delivery $i+n$. To consider incompatibilities, we just have to set $z_{i,j} = 0$ for $(i, j) \in E_I$. The proposed model (*ITVM*) is obteined adding to an ATSP formulation –based on the classical one for TSP introduced by Dantzig, Fulkerson and Jhonson (1954)– the following

$$\begin{aligned} x_{i,j} + z_{k,i} &\leq z_{k,j} + 1 \quad && \forall i, j \in V_G, k \in P; j \neq i, i+n, k, k+n; i \neq k+n \\ x_{i,j} + z_{k,j} &\leq z_{k,i} + 1 \quad && \forall i, j \in V_G, k \in P; j \neq i, i+n, k; i \neq k, k+n \end{aligned}$$

To test the above 4 formulations we took PDP instances in "Set 1" of [2]. From this set, we used the 10 instances with 10 and 15 tasks (22 and 32 nodes respectively). For every PDP instance and every percentage K in the set$\{10\%, 25\%, 50\%, 70\%\}$, we generated 5 different instances adding random incompatibility graphs with $K\%$ of density. In conclusion, we have built 200 instances.

Our models have been tested in the context of a *Branch and Cut* approach using CPLEX 10.1 on a 1Ghz processor. Results show that ITVM clearly outperforms the other models on instances with high incompatibility and, disregarding the density of I, the larger the $|P|$, the better it performs in comparison to PV1, PV2 and 2iFM. Regarding these results, we decided to study the ITVM polyhedron.

3 Polyhedral analysis

For PDPwI we have found the equalities below:

$$\begin{aligned}
z_{i,j} + z_{j,i} &= z_{i,j+n} + z_{j,i+n} && \forall (i,j) \notin E_I \\
x_{i,j} &= z_{i,j} && \forall (i,j) \notin E_I, d_I(i) = n-2 \\
x_{j+n,i+n} &= z_{i,j+n} && \forall (i,j) \notin E_I, d_I(i) = n-2 \\
x_{j,i+n} + z_{i,j+n} &= z_{i,j} + x_{i,j+n} && \forall (i,j) \notin E_I, i<j, d_I(i) < n-2, d_I(j) < n-2, \\
& && P - \{i,j\} = \Gamma(i)_I \cup \Gamma(j)_I \\
z_{i,j} + x_{k+n,i+n} &= x_{i,j} + z_{i,k+n} && \forall i : d_I(i) = n-3, \\
& && (i,j),(i,k) \notin E_I, j<k, (j,k) \in E_I
\end{aligned}$$

We have proved that when $E_I = \emptyset$ (PDP) the first equation and the constraints of the assignment problem constitute the minimal equation system (the remaining sets are empty). Therefore, in this case the dimension of the polyhedron is $\frac{11}{2}n^2 - \frac{13}{2}n - 1$. We have also proved that, for the general case, equalities listed above form the minimal equation system.

Finally, we have found 3 families of valid cuts whose size is $O(n)$, 12 with size $O(n^2)$ and 8 with size $O(n^3)$. As an example, we provide these families:

$$\begin{aligned}
z_{i,j} + z_{j,i} + x_{j+n,i} + x_{i+n,j} &\leq 1 && \forall i,j \in P, i \neq j, (i,j) \notin E_I \\
z_{i,j} + z_{j,k} + z_{k,i} + x_{i,k} + x_{i+n,j} + x_{j+n,k} + x_{k+n,i} &\leq 2 && \forall i,j,k \in P, i \neq j, k \neq i, k \neq j, \\
& && (i,j),(j,k),(k,i) \notin E_I \\
z_{i,k} + z_{j,k} + x_{k,j} + x_{k,i} &\leq 1 && \forall i,j \in P, i \neq j, \\
& && k \in P/\{i,j\} \cup D/\{i+n, j+n\}, \\
& && (i,j) \in E_I, (j,k),(k,i) \notin E_I
\end{aligned}$$

4 Branch and Cut

A subset of 8 families of cuts with size $O(n^2)$, the 8 with size $O(n^3)$ and all the equalites, proved to be useful in the context of an exact algorithm. We added the

quadratic families to the model and the cubic ones through a separation procedure; results can be seen in the second column of table 1.

We have also developed a primal heuristic consisting in a greedy constructive phase followed by a local search. First phase builds a feasible solution "close" to the linear relaxation vector. Neighborhood for local search is defined by moving every possible node to every feasible position. The first phase has been implemented in $O(n^3)$ and local search iterations in $O(n^2)$. The heuristic found the optimum in the large majority of the instances (performing also very well in the remaining instances) and the best solution was found in the first 10% of nodes of the decision tree in average. The improvement on the dual bounds using the cutting planes, combined with primal bounds found with the heuristic, proved to be useful to reduce the number of explored nodes in the decision tree. Results can be seen in the third column of table 1.

	Without cuts	nor heur.	With cuts and	without heur.	With cuts	and heur.
Inc.(%)	Time(s)	#Nodes	Time(s)	#Nodes	Time(s)	#Nodes
10	1010,6	20423,64	863,36	16598,6	402,64	3195,84
25	339,08	10153,52	271,92	7519,84	129,44	1844,64
50	16,96	1157,08	19,12	979,24	11,96	407,24
70	0,6	121,94	1,04	108,92	0,52	43,18

Table 7. Average nodes and solution times using cuts and primal heuristic on instances of 10 tasks

References

[1] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia and G. Laporte, Static Pickup and Delivery Problems: A Classification Scheme and Survey, *TOP*, **15**, 1–31, (2007).

[2] I. Dumitrescu, S. Ropke, J.-F. Cordeau and G. Laporte, The Traveling Salesman Problem with Pickup and Delivery: Polyhedral Results and a Branch-and-Cut Algorithm, *Math. Program. Series A*, **121**, 269–305 (2010).

[3] X. Feng, Formal sequential lagrangian algorithm for large scale resource scheduling optimization, US Patent 2006/0089863 A1 (2006)

[4] L. Gouveia and J.M. Pires, The asymmetric travelling salesman problem and a reformulation of the Miller-Tucker-Zemlin constraints, *European J. Oper. Res.*, **112**, 134–146, (1999).

[5] S.C. Sarin, H.D. Sherali and A. Bhootra, New tighter polynomial length formulations for the asymmetric travelling salesman problem with and without precedence constraints, *Oper. Res. Lett.*, **33**, 62–70, (2005).

Representations of Power Series over Word Algebras

Ulrich Faigle, [a] Alexander Schönhuth [b]

[a] *Mathematical Institute, University of Cologne, Weyertal 80, 50931 Köln, Germany*

[b] *Centrum Wiskunde & Informatica, Science Park 123, 1098 XG Amsterdam, Netherlands*

Key words: power series, representation, equivalence, automata, Markov chain

1 Introduction

We introduce a combinatorial model of power series that admit linear representations in a general sense. In particular: we establish a generic approach to obtain efficient tests for equivalence for power series with finite-dimensional representations and show how the equivalence problems for hidden Markov chains and nondeterministic finite automata of degree $\leq k$ may be solved, for example. The latter have been extensively discussed in the literature, but were tackled by different approaches [4,1]. Our contribution can be seen as part of a general effort to analyze stochastic/statistical models with algebraic techniques (*cf.* [2]).

2 Word algebras and power series

Think of the non-empty set X as an *alphabet* with letters $x \in X$. A finite string $w = x_1 \dots x_t$ of (not necessarily pairwise different) letters $x_i \in X$ is a *word of length* $|w| = t$. X^t denotes the set of all words of length t with $X^0 = \{\Box\}$ in particular. The collection

$$X^* := \bigcup_{t \geq 0} X^t$$

of all words is a semigroup (with neutral element $\Box$) under the concatenation operation $(v, w) \mapsto vw$. A *power series* (over the field $\mathbb{K}$) is an element $f \in \mathbb{K}^{X^*}$, which may be denoted as a formal sum

$$f = \sum_{w \in X^*} f_w w \quad (f_w \in \mathbb{K}).$$

For notational convenience, we also express the coefficients of a power series f as $f(w) := f_w$ (*i.e.*, we take the view of a function $f : X^* \to \mathbb{K}$).

3 Representations

Let V be a $\mathbb{K}$-vector space $\mathcal{L}(V)$ the vector space of linear operators on V. A V-*representation* of the word algebra X^* arises from a map $\sigma : X \to \mathcal{L}(V)$ that may be extended to all of X^*. Denoting the image of $x \in X$ by σ^x, we define the induced *covariant* representation $\sigma : X^* \to \mathcal{L}(V)$ by

(1) $\sigma^{\square} := id$.

(2) $\sigma^w = \sigma^{x_1 x_2 \ldots x_t} := \sigma^{x_1} \circ \sigma^{x_2} \ldots \circ \sigma^{x_t}$ for all words $w = x_1 \ldots x_t \in X^*$.

Notice that the covariant representation σ is a semigroup homomorphism:

$$\sigma^{uv} = \sigma^u \circ \sigma^v \quad (u, v \in X^*).$$

The *dimension* of the V-representation σ is defined as the dimension of V:

$$\dim \sigma := \dim_{\mathbb{K}} V$$

In the finite-dimensional case $\dim \sigma = n < \infty$, there is no loss of generality to think of V as the n-dimensional parameter space $\mathbb{K}^n$ and of the operators σ^w as $(n \times n)$-matrices $[\sigma^w_{ij}]$ with coefficients $\sigma^w_{ij} \in \mathbb{K}$.

3.1 Representations of power series

Let $\sigma : X^* \to \mathcal{L}(V)$ be a representation on some $\mathbb{K}$-vector space V. Choose furthermore a vector $\pi \in V$ and a linear functional $\gamma : V \to \mathbb{K}$. Then (σ, γ, π) defines a power series $f(\sigma, \gamma, \pi) \in \mathbb{K}[X]$ with coefficients

$$f_w = \gamma(\sigma^w(\pi)) \quad (w \in X^*). \tag{1}$$

We say that (σ, γ, π) is a *representation* of the the power series $f = f(\sigma, \gamma, \pi)$ and we refer to $\dim \sigma$ as the *dimension* of the representation (σ, γ, π).

4 Examples

We give two examples of relevant power series representations.

Ex. 1 (Hidden Markov chains) *Let M be an $(n \times n)$-matrix with real coefficients $m_{ij} \geq 0$ and column sums $\sum_i m_{ij} = 1$. Let $\pi \in \mathbb{R}^n$ be a non-negative vector with component sum 1. (M, π) describes a random walk on $N = \{1, \ldots, n\}$ that starts with the empty word $\square$ at time $t = 0$, then moves to j_1 with probability π_{j_1} at time $t = 1$ and subsequently to j_2 with probability $m_{j_1 j_2}$ at time $t = 2$ and so on.*

Let $\varphi : N \to X$ be an arbitrary (but fixed) function. Then the random walk induces a stochastic process (X_t), where $X_t = \varphi(j)$ if the random walk has reached $j \in N$

at time t. So the elements $j \in N$ can be viewed as the "hidden" states of the process (X_t) as they can be observed only indirectly through the symbol $\varphi(j)$ they produce.

Let $S^x = [s^x_{ij}]$ be the $(n \times n)$-matrix given by

$$s^x_{ij} := \begin{cases} m_{ij} \text{ if } \varphi(j) = x \\ 0 \quad \text{otherwise.} \end{cases}$$

With $\mathbf{1} := [1, 1, \ldots, 1]^T$ being the sum of the unit vectors $\mathbf{e}_i \in \mathbb{R}^n$,

$$P = (\{S^x \mid x \in X\}, \mathbf{1}^T, \pi)$$

is an n-dimensional representation of the probability function p of the stochastic process (X_t) by virtue of

$$p(x_1 x_2 \ldots x_t) = \Pr\{X_1 = x_1, X_2 = x_2, \ldots, X_t = x_t\} = \mathbf{1}^T (S^{x_t} \ldots S^{x_1})\pi.$$

Ex. 2 (Non-deterministic Finite Automata) *Let $A = (S, X, \delta, s_0, F)$ be a non-deterministic finite automaton (NDFA) (see e.g. [4] for detailed definitions).*
We write $i = 1, ..., |S|$. Given a NDFA, consider the matrices $T_x \in \{0, 1\}^{S \times S}$ for all $x \in X$, defined by

$$(T_x)_{ij} = \begin{cases} 1 & j \in \delta(i, x) \\ 0 & \text{else} \end{cases}$$

Without loss of generality, let $s_0 = 1$. Let $e_j \in \mathbb{R}^S, j = 1, ..., |S|$ be the canonical basis vectors of $\mathbb{R}^S$ and

$$e_F := \sum_{i \in F} e_i.$$

It then follows from a straighforward proof by induction on the word length $|v|$ that

$$e_1^T T_{x_1} \cdot \ldots \cdot T_{x_n} e_F$$

is the number of accepting paths for $v = x_1 ... x_n$. That is

$$(\{T_x, x \in X\}, e_1^T, e_F) \quad \textit{represents} \quad \sum_{u \in X^*} f_u u$$

where f_u the number of accepting paths for u in A.

5 Equivalence of representations

It is quite possible that two different representations $F = (\sigma, \gamma, \pi)$ and $F' = (\sigma', \gamma', \pi')$ determine the same power series $f \in \mathbb{K}[X]$. The *equivalence problem* is:

(**EP**) Decide whether the representations F and F' determine the same power series.

We show that the equivalence problem can be solved efficiently under the following assumptions:

(1) X is finite and the representations F and F' are finite-dimensional with $\dim F = n$ and $\dim F' = n'$.
(2) F is presented by an explicit set $\{\sigma^x \mid x \in X\}$ of $(n \times n)$-matrices σ^x over the field $\mathbb{K}$ and parameter vectors $\gamma, \pi \in \mathbb{K}^n$.
(3) F' is presented by an explicit set $\{(\sigma')^x \mid x \in X\}$ of $(n' \times n')$-matrices $(\sigma')^x$ over the field $\mathbb{K}$ and parameter vectors $\gamma', \pi' \in \mathbb{K}^{n'}$.

In particular, we can show:

Theorem 5.1 *The equivalence problem for presentations F and F' can be solved by an algorithm whose number of arithmetic operations is bounded by a polynomial in $\max\{\dim(F), \dim(F')\}$ and $|X|$.*

References

[1] U. FAIGLE AND A. SCHÖNHUTH, "Efficient tests for equivalence of hidden Markov processes and quantum random walks", *IEEE Transactions on Information Theory*, to appear, 2011.

[2] L. PACHTER AND B. STURMFELS, *Tropical geometry of statistical models*, Proc. Proc. Nat. Academy of Sciences 101 (2004), 16132.16137.

[3] M. DRTON, B. STURMFELS AND S. SULLIVANT, *Lectures on Algebraic Statistics*, Oberwolfach Seminar Series 39, Birkhäuser 2009.

[4] R.E. STEARNS AND H.B. HUNT III, "On the Equivalence and Containment Problems for Unambiguous Regular Expressions, Regular Grammars and Finite Automata", *SIAM J. Comp.*, 14(3), p. 598–611, 1985.

The Boat and Barge Routing Problem

Mette Gamst, [a] Niels Kjeldsen [b]

[a] *University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark*

[b] *University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark*
DONG Energy, Kraftvaerksvej 53, DK-7000 Fredericia, Denmark

Key words: BBRP, VRPTW, TTRP, column generation

1 Introduction

The thermal power plants in Denmark need to be refueled at regular intervals to ensure stable and reliable production of electricity. Fuel (typically coal) for the power plants is delivered from overseas to two central depots in Denmark, from here the fuel is distributed to the power plants by an internal fleet of tug boats and barges. The fuel is loaded into a barge which is then pulled by a tug boat to a power plant harbor where the barge is unloaded. During the loading and unloading period the tug boat is not needed and can service other deliveries. Barges cannot move on their own. The internal fleet does not always have the capacity to service all power plants, in these cases an external delivery can be made at a significantly higher cost.

The Barge and Boat Routing Problem (BBRP) shares similarities with the Vehicle Routing Problem with Time Windows (VRPTW), see e.g. Toth and Vigo [6], and with the Truck and Trailer Routing Problem (TTRP), see e.g. Chao [2].

As in the VRPTW each delivery is also associated with a time window. The key difference from the TTRP are the autonomous vehicles in the BBRP, i.e., the possibility to change which tug boat pulls which barge — in the TTRP trailers are always moved by the same truck. Drexl considers a variant of the Vehicle Routing Problem in [3] called the Vehicle Routing Problem with Trailers and Transshipments (VRPTT). The VRPTT also includes autonomous vehicles along with other complicating constraints. Drexl solves a simplified core part of the problem with a branch-and-cut algorithm, but only for a very small instance. He concludes that the addition of autonomous vehicles greatly increases the complexity of the problem.

The contribution of this work is the introduction of the BBRP, a MIP formulation (which we do not state here) and a branch-and-price algorithm for the BBRP.

⋆ This project was supported in part by DONG Energy and by the Villum-Kann-Rasmussen Foundation.

The mathematical formulation is based on formulations for the VRPTW, see e.g. Kallehauge et al. [4]. We propose a branch-and-price algorithm with two pricing problems: one for generating barge paths and one for generating tug boat paths. Both the mathematical formulation and the branch-and-price algorithm have been tested on a number of real-life test instances based on data from the Danish utility company DONG Energy. Early results indicate that the branch-and-price algorithm clearly outperforms the mathematical formulation.

2 Branch-and-price algorithm

The BBRP is Dantzig-Wolfe decomposed and solved to optimality using a branch-and-price (BP) algorithm. Two different pricing problems are generated; one for creating paths for tug boats and one for creating paths for barges. The master problem merges the paths into an overall feasible solution. The BBRP works on a graph $G = (N, A)$. The nodes N is the union of depots (denoted D) and deliveries at power plants (denoted L); arcs A connect the nodes. Time is discretized into a set of time stamps T. Let k_B be the set of barges, k_T be the set of tug boats and let the set of paths for vehicles of type $k \in \{k_B, k_T\}$ be denoted P_k. The binary variable z_i is an expensive, external delivery to power plant i, with cost c_i. Let y_p^k be a binary variable denoting whether or not a vehicle k travels on path p, let $c_p^k \geq 0$ be the cost for using a vehicle of type k on path p. The master problem is:

$$\min \quad \sum_{p \in P_{k_T}} c_p^{k_T} y_p^{k_T} + \sum_{i \in L} c_i z_i \tag{1}$$

$$\text{s.t.} \quad \sum_{p \in P_{k_B}} \delta_p^i y_p^{k_B} + z_i = 1 \qquad \forall i \in L \tag{2}$$

$$\sum_{p \in P_{k_B}} \delta_{ij\tau}^p y_p^{k_B} - \sum_{p \in P_{k_T}} \delta_{ij\tau}^p y_p^{k_T} \leq 0 \qquad \forall (i,j) \in A, \forall \tau \in T \tag{3}$$

$$\sum_{p \in P_k} y_p^k \leq |k| \qquad k \in \{k_B, k_T\} \tag{4}$$

$$y_p^k \in \{0,1\} \qquad \forall p \in P_k, \forall k \in K \tag{5}$$

$$z_i \in \{0,1\} \qquad \forall i \in L \tag{6}$$

The objective function (1) minimizes the total cost of transporting fuel. The first constraints (2) ensure that all deliveries are satisfied: constant δ_p^i denotes whether or not path p performs delivery i. Constraints (3) ensure that barges sail with tug boats: constant $\delta_{ij\tau}^p$ denotes whether or not path p sails on arc (i,j) at time τ. The number of used barges resp. tug boats is limited in constraints (4), where $|k|$ denotes the number of vehicles of type k. Finally, bounds (5)-(6) force variables to take on feasible values.

Pricing problem for tug boats
Let $\pi_i \in \mathbb{R},\ \lambda_{ij\tau} \leq 0$ and $\omega_{k_T} \leq 0$ be the dual variables of constraints (2), (3) and

(4), respectively. The reduced cost for a tug boat path p is:

$$\bar{c}_p^{k_T} = c_p^{k_T} + \sum_{(i,j)\in A} \sum_{\tau \in T} \lambda_{ij\tau}^{k_T} - \omega_{k_T} \leq 0 \Rightarrow \sum_{(i,j)\in A} \left(c_{ij}^{k_T} + \sum_{\tau \in T} \lambda_{ij\tau}^{k_T} \right) \leq \omega_{k_T} \quad (7)$$

A tug boat must satisfy constraints on time windows and must travel from one depot or power plant to the next such that the reduced cost (7) is minimized. Cycles are allowed, i.e., a path can visit the same depot or power plant several times. The reduced costs can be interpreted as arc weights, hence the pricing problem becomes the *Shortest Path Problem with Resource Constraints* and is solved by a pseudo-polynomial labeling algorithm [1].

Pricing problem for barges
The reduced cost for a barge path p is:

$$\bar{c}_p^{k_B} = -\sum_{i \in L} \pi_i - \sum_{(i,j)\in A} \sum_{\tau \in T} \lambda_{ij\tau} \leq \omega_{k_B} \quad (8)$$

Again, the reduced cost (8) must be minimized according to constraints on time windows and path connectivity. By subtracting π_i from all edges into i, the reduced costs can be viewed as arc weights and a barge path can be seen as pairs of depots and deliveries, where each pair must add to the negative reduced cost. Enumerating all pairs of depots and deliveries gives $\mathcal{O}(|D||L||T|^2)$ pairs. Combining these pairs into a barge route can be done with a dynamic programming approach, and this gives a pseudo-polynomial algorithm for the pricing problem.

Branching
Branching is necessary, when the solution in the current branch-and-bound node is fractional. A solution with binary barge path variables can be transformed into being integer using Proposition 4 in [7], hence branching on barge usage of arcs at specific times eventually ensures a feasible solution and is a finite strategy.

3 Computational Evaluation

The proposed BP-price algorithm is implemented and compared to solving the mathematical formulation using a standard MIP-solver. The solution methods are tested on instances based on real-life data provided by DONG Energy, Denmark. A test instance consists of a number of deliveries at a number of power plants. The start time of a delivery is defined by the time at which the plant has enough capacity to stock a delivery of fuel. The end time of a delivery is defined by the time at which the plant reaches the least allowed amount of available fuel. Time is discretized into hours, i.e., each time stamp represents an hour. The instance covering a full year's fuel consumption consists of 122 deliveries, which all have large time windows.

The proposed BP-algorithm is implemented using the COIN Bcp framework [5]. CPLEX 12.1 is used for solving the MIP formulation. The presented work is still in progress, hence a full computational evaluation has not yet been performed. Early results suggest that the BP-algorithm outperforms solving the original formulation. Current performance of the BP-algorithm indicates the necessity of a good primal heuristic. Without such a heuristic, the BP-algorithm converges slowly and generates a very large number of columns; the far majority of these columns does not improve the solution. The many columns are caused by large time windows giving rise to many different arrival and departure times. Also, the reduced costs only indicate that expensive arcs should not be used at certain times instead of indicating that the arcs should *never* be used. Finally, ensuring that barges and tug boats sail together adds to the number of columns.

References

[1] J. E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989.

[2] I.-M. Chao. A tabu search method for the truck and trailer routing problem. *Computers and Operations Research*, 29(1):33–51, 2002.

[3] M. Drexl. *On Some Generalized Routing Problems*. PhD thesis, RWTH Aachen University, 2007.

[4] B. Kallehauge, J. Larsen, O. Madsen, and M. Solomon. Vehicle routing problem with time windows. *Column Generation*, pages 67–98, 2005.

[5] R. Lougee-Heimer. The common optimization interface for operations research. *IBM Journal of Research and Development*, 47:57–66, 2003.

[6] P. Toth and D. Vigo. *The Vehicle Routing Problem*. SIAM - Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

[7] F. Vanderbeck. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1):111–128, 2000.

A Probabilistic Cellular Automata algorithm for the clique problem

A. Gaudilliére, [a] Antonio Iovanella, [b] B. Scoppola, [c] E. Scoppola, [a] M. Viale [d]

[a] *Dipartimento di Matematica, University of Rome "Roma Tre"*
Largo San Murialdo, 1 - 00146 Rome, Italy
{gaudilli,scoppola}@mat.uniroma3.it

[b] *Dipartimento di Ingegneria dell'Impresa, University of Rome "Tor Vergata"*
Via del Politecnico, 1 - 00133 Rome, Italy.
iovanella@disp.uniroma2.it

[c] *Dipartimento di Matematica, University of Rome "Tor Vergata"*
Via della Ricerca Scientifica - 00133 Rome, Italy
scoppola@mat.uniroma2.it

[d] *Dipartimento di Fisica, University of Rome "Roma Tre"*
Via della Vasca Navale, 84 - 00146 Rome, Italy
viale@fis.uniroma3.it

---

1 Extended Abstract

In this paper we want to present a randomized algorithm for the metaheuristic search of the maximum clique of a graph. The method is substantially new because it exploits the featureas of the so-called *Probabilistic cellular automata* to sample a probability measure that is mainly concentrated on the large cliques of the graph. This techniques is presented here just applied to the maximum clique problem, but it can be easily generalized.

Let $G = (V, E)$ be a graph, and consider the set of *configurations* σ on V as $\sigma \in \{0,1\}^V := \mathcal{X}$.

Consider the following function on $V \times V$

$$J_{ij} = \begin{cases} 0 & \text{if} (i,j) \in E \\ 1 & \text{if} (i,j) \notin E \end{cases} \tag{1}$$

and

$$J_{i,i} = -h \quad \forall i \in V \tag{2}$$

For each site $i \in V$ and each configuration σ we define the *cavity field*:

$$h_i(\sigma) = \sum_j J_{ij}\sigma_j \equiv \sum_{j \neq i} J_{ij}\sigma_j - h\sigma_i \tag{3}$$

Finally, for each pairs of configurations σ, σ' we can define the *Hamiltonian*

$$H(\sigma, \sigma') = 2\sum_{(i,j)} J_{ij}\sigma_i\sigma'_j - h\sum_i \sigma_i\sigma'_i = \sum_i h_i(\sigma)\sigma'_i \tag{4}$$

The hamiltonian $H(\sigma, \sigma')$ is symmetric in the exchange σ, σ' and its value is minimal only on pairs of configurations σ, σ' such that $\sigma = \sigma'$ with σ a maximum clique.

For any integer $k \leq n = |V|$ we define

$$\mathcal{X}_k := \{\sigma \in \mathcal{X} : \sum_{i \in V} \sigma_i = k\} \tag{5}$$

We consider the canonical case, i.e., we define a MCMC on the configuration space $\mathcal{X}_k$; we define the partition function depending on σ:

$$Z_\sigma = \sum_{\tau \in \mathcal{X}_k} e^{-\beta H(\sigma,\tau)} \tag{6}$$

and the transition probabilities:

$$P(\sigma, \sigma') = \frac{e^{-\beta H(\sigma,\sigma')}}{\sum_{\tau \in \mathcal{X}_k} e^{-\beta H(\sigma,\tau)}} = \frac{e^{-\beta H(\sigma,\sigma')}}{Z_\sigma} \tag{7}$$

By an immediate computation we can conclude that the detailed balance condition w.r.t. the invariant measure

$$\Pi_\sigma = \frac{\sum_\tau e^{-\beta H(\sigma,\tau)}}{\sum_{\tau,\sigma} e^{-\beta H(\sigma,\tau)}} = \frac{Z_\sigma}{Z} \tag{8}$$

is verified for large β, the stationary measure Π_σ is exponentially concentrated on cliques.

Note that at each step all the sites are updated; it could be considered a canonical version of probabilistic cellular automata (PCA).

The dynamics we just presented has some very interesting features.

First, changing suitably β and h we can see that their typical paths tend to have very different behaviors: for large h the system tends to "freeze" on fixed configurations, while for small h it tends to flip all its component in a single step. From a detailed numerical study we realized that for suitable intermediate values of h the system tend to reach quite quickly cliques of size k, if any. The algorithm has been tested on Erdos random graphs of size up 8000 vertices, finding in a reasonable time a maximum clique

References

[1] B. BOLLOBAS *Random graph*, 2nd ed.,Cambridge University Press, 2001.

[2] M. CARAMIA, G. FELICI *Mining relevant information on the Web: a clique based approach*, International Journal of Production Research, special issue on Data Mining, accepted 2006.

[3] M. MÉZARD, G. PARISI, *The cavity method at zero temperature*, J. Stat. Phys 111 (2003) 1.

[4] M. MÉZARD, G. PARISI, R. ZECCHINA, *Analytic and algorithmic solution of random satisfiability problems*, Science 297, 812-815 (2002).

Odd cyclic surface separators in planar graphs

Petr A. Golovach, [a] Marcin Kamiński, [b 1] Dimitrios M. Thilikos [c]

[a] *School of Engineering and Computing Sciences, University of Durham, UK*

[b] *Département d'Informatique, Université Libre de Bruxelles, Belgium*

[c] *Department of Mathematics, National and Kapodistrian University of Athens, Greece*

Key words: planar graphs, odd cycles, irrelevant vertex

Given a plane graph G and two vertices s and t in G, an *s-t surface separator* is a closed curve C such that s belongs to the other region of $\mathbb{R}^2 \setminus C$ than t. A cycle C is called a *cyclic surface separator* if C is an s-t surface separator, and an *odd cyclic surface separator* if C is an odd cycle. We consider the problem of finding odd cyclic surface separators in planar graphs and show that the problem can be solved in polynomial time.

1 Introduction

Given two vertices $s, t \in V(G)$, we are interested in finding a cycle in G such that one of the two regions of $\mathbb{R}^2 \setminus C$ contains s and the other contains t. We show that the problem is solvable in polynomial time, even when we impose a parity condition on the length of the cycle.

Our proof is based on the *irrelevant vertex technique* introduced by Robertson and Seymour in Graph Minors: we prove that either the treewidth of the input graph G is small, and then the problem can be solved by dynamic programming, or when it is large, the G contains a vertex v such that $G-v$ is a yes-instance if and only if G is a yes-instance. Moreover, such an irrelevant vertex can be found in polynomial time. We recursively remove irrelevant vertices, each time reducing the size of the graph. Finally, the treewidth is small and we apply the dynamic programming approach.

For a cycle C in a plane graph G, there are two open regions of $\mathbb{R}^2 \setminus C$. The one containing the outerface of G is the *exterior* of C. The other one if the *interior* of C and we denote it by $\mathrm{int}(C)$. If a path contains a vertex from the interior and a vertex from the exterior of a cycle, we say that is *crosses* the cycle.

[1] Chargé de Recherches du FRS-FNRS.

2 Results

2.1 The bipartite case

Lemma 3 *Let G be a plane graph, $s, t \in V(G)$, C be a cycle in G and G' be the graph induced by the vertices in the interior of C. If $G[V(G')]$ is bipartite, $s, t \notin V(G' \cup C)$ and exists a vertex $v \in \text{int}(C)$, then if G has an odd cyclic surface separator, so does $G - v$.*

Proof. Let us consider an odd cyclic surface separator in G. If it does not cross C, it contains no vertex of $V(G')$. (Notice that s and t lie in the exterior of C.) Assume that every odd cyclic surface separator crosses C and let C^* be one which crosses C the minimum number of times. Exactly one of the terminals, without loss of generality we assume it is s, lies in the interior of C^*.

Let P be the set of edges of C^* that lie in the interior of C, and Q be the set of edges of C that lie in the interior of C^*. (P and Q are disjoint.) Then, $C \cup P - Q$ is a collection of cycles. Terminal s lies in the interior of one of the cycles in the collection; let this cycle be called C_s.

Notice that C_s does not cross C. If C_s was an odd cycle, it would be an odd cyclic surface separator; hence, C_s is even. Hence, there must be another cycle in $C \cup P - Q$ which is of odd parity, since C^* was odd. Let us call the odd cycle C_{odd}.

Now we will investigate how C_s and C_{odd} are related. Suppose there is a subpath $v_L P v_R$ of C_s such that $v_L, v_R \in C$ but no vertex of P belongs to C. Now, C can be split into two disjoint paths between vertices v_L and v_R. Exactly one of those two paths has the property that the cycle formed by $v_L P v_R$ and this path does not contain the other path inside. Let us call that path P'. The cycle formed by $v_L P v_R$ and P' will be called a *pouch*.

Clearly, C_{odd} lies in one of the pouches of C_s; we will look at this pouch now. Let v_L, v_R, P and P' be as in the definition of the pouch. Vertices v_L and v_R split C_s into two paths; one is P and let the other be called Q. We investigate two cases:

CASE 1. First let us assume that the other terminal t **does not lie in the interior of the pouch**. Let v'_L and v'_R be the first and last vertex along P' that belong to C_{odd}. Notice that since C_{odd} is odd, vertices v'_L and v'_R split it into two paths of different parity P_{even} and P_{odd}. Consider the two cycles formed by the path $v'_L P' v_L Q v_R P' v'_R$ and either P_{even} or P_{odd}. They both are s, t-separators and they do not cross C. However, exactly one of them is odd; a contradiction.

CASE 2. Now let us assume that t **lies in the interior of the pouch** of C_s in which C_odd also lies. We define pouches for C_{odd} analogously. If t does not lie in the interior of the pouch of C_{odd} that contains C_s, then the same argument as in CASE 1 can be applied to show a contradiction.

Let us assume that t lies in the interior of the pouch of C_{odd} that contains C_s. Also let v'_L, v'_R, P_{odd}, and P_{even} be as in CASE 1. Consider the two cycles formed by the path $v'_L P' v_L P v_R P' v'_R$ and either P_{even} or P_{odd}. They both are s, t-separators and

they do not cross C. However, exactly one of them is odd; a contradiction. ∎

2.2 Odd cycle

Lemma 4 *Let G be a 2-connected, plane graph, $s, t \in V(G)$, C be a cycle in G and G' be the graph induced by the vertices in the interior of C. If $G[V(G')]$ is non-bipartite and contains a odd cycle C' such that exists a vertex $v \in \mathrm{int}(C')$, $s, t \notin V(G' \cup C)$, then if G has an odd surface separator, so does $G - v$.*

Proof. As in the proof of Lemma 3, we can assume that there is a cycle C_s whose interior contains s. Also, $|C \cap C_s| \geq 2$. From 2-connectivity, there are two paths P_1 and P_2 between $C \cap C_s$ and C'. The endpoints of P_1 and P_2 on C' split C' into two paths P_1' and P_2'. Similarly, the endpoints of P_1 and P_2 split C_s into two paths; let P_s be one of these paths, the one that contains vertices from the exterior of C. Then, both $P_s P_1 P_1' P_2$ and $P_s P_1 P_2' P_2$ are cyclic separators and they are of different parity.
Clearly, if G has an odd surface separator, so does $G - v$. ∎

2.3 Algorithm

Let G be a plane input graph. The problem easily reduces to 2-connected graphs so we can assume that G is 2-connected. There exists a constant c such that if the treewidth of G is at least c, then (1) it contains a cycle C with a vertex in its interior and (2) if the graph induced by the vertices in the interior of C is not bipartite, the interior of C contains an odd cycle whose interior contains a vertex. Therefore, if the treewidth of G is at least c, then we can find using Lemmas 3 and 4 an irrelevant vertex v. Removing v reduces the size of the graph and we recurse. If the treewidth of G is at most c, then the problem can be solved efficiently by dynamic programming.

Theorem 18 *There exists a polynomial-time algorithm deciding given a plane graph G and two of its vertices s, t, whether G contains an odd cyclic surface separator.*

Computing Derivatives via Compression : An Exact Scheme

Shahadat Hossain [1]

Department of Mathematics and Computer Science, University of Lethbridge, Canada

Abstract

We describe a branch-and-bound approach for determining a sparse Jacobian matrix $A \in \mathbb{R}^{m\times n}$ using an extended notion of *structural independence* or *structural orthogonality* of segments of columns of A or A^T. We assume that the sparsity pattern of the sparse Jacobian is a priori known. Our branching strategy is based on Zykov contraction algorithm [8].

Key words: Sparsity, Automatic Differentiation, optimal direct methods, column partitioning.

1 Introduction

We consider the problem of determining the Jacobian matrix $F'(x)$ of a mapping $F : \mathbb{R}^n \mapsto \mathbb{R}^m$. Using differences, the jth column of the Jacobian matrix may be approximated as

$$\left.\frac{\partial F(x+ts)}{\partial t}\right|_{t=0} = F'(x)s \approx As = \frac{1}{\varepsilon}[F(x+\varepsilon s) - F(x)] \equiv b \qquad (1.1)$$

with one extra function evaluation, where $\varepsilon > 0$ is a small increment. Also Algorithmic (or Automatic) Differentiation (AD) [3] forward mode gives $b = F'(x)s$ accurate up to the machine round-off, at a cost which is a small multiple of the cost of one function evaluation. The Jacobian matrix determination problem (JMDP) can be stated as below.

Problem 2 (JMDP) *Obtain vectors $s_i \in \mathbb{R}^n, i = 1,\ldots,p$ and $w_j \in \mathbb{R}^m, j = 1,\ldots,q$ with $p+q$ minimized such that the products $b_k = As_k,\ j = 1,\cdots,p$ or $B = AS$ and $c_k^T = w_k^T A,\ k = 1,\cdots,q$ or $C^T = W^T A$ determine the matrix A uniquely.*

[1] This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

In absence of any sparsity information, one may use the Cartesian basis vectors $e_i, i = 1, \ldots, n$ in (1.1) using n extra function evaluations, or in the case of AD, m reverse mode calculations or n forward mode calculations, whichever is smaller. Specifically, for a scalar function the gradient is obtained with just one reverse calculation. On the other hand, for example, if a group of columns of matrix A, say columns j and l, are *structurally orthogonal* i.e. no two columns have nonzero entries in the same row position only one extra function evaluation

$$F'_j + F'_l \approx A(:, j) + A(:, l) = \tfrac{1}{\varepsilon}[F(x + \varepsilon(e_j + e_l)) - F(x)],$$

is sufficient to read-off the nonzero entries from the product $b = As$ where $s \equiv (e_j + e_l)$, an observation due to [1]. A group of nonzero unknowns are said to be *directly determined* or read-off from the product $b = As$ if no floating point arithmetic operation is needed to recover them. Thus, if the columns (rows) can be partitioned into p (q) structurally orthogonal groups then the Jacobian matrix is directly determined from the *row compressed (column-compressed)* matrix $B = AS$ $(C^T = W^T A)$. In this case we have a *one-sided compression* of matrix A. Specifically, with $q = 0$ we have row compression and with $p = 0$ we have column compression.

In this paper, we consider exact methods for one-sided direct determination of sparse Jacobian matrices. We show that a branching scheme originally proposed in [8] for exact coloring of undirected graphs can be applied to the Jacobian matrix determination problem via *merging* of structurally orthogonal column segments. Although the matrix problem can be formulated as a graph coloring problem [2,6], we believe that the matrix formulation is more natural and yields efficient implementation since, for a sparse matrix A its column intersection graph $G(A)$ is likely to be dense. Further, we avoid forming a large graph [4] corresponding to column segments.

2 Compression and Branching

Let $\mathcal{A} \in \{0, 1\}^{m \times n}$ be obtained from the Jacobian A where $\mathcal{A}(i, j) = 1$ if $A(i, j) \neq 0$ and $\mathcal{A}(i, j) = 0$, otherwise. Without loss of generality, we assume that indices j and l satisfy $j < l$. For structurally orthogonal columns $\mathcal{A}(:, j)$ and $\mathcal{A}(:, l)$ the sequence of operations

$$\mathcal{A}(:, j) \leftarrow \mathcal{A}(:, j) + \mathcal{A}(:, l); \;\; \mathcal{A}(:, l) \leftarrow \mathbf{0}$$

represents a *row compression* of matrix $\mathcal{A}$ where the columns j and l are merged together and can be identified as column j of the compressed matrix while column l is now replaced with a vector of all zeros $\mathbf{0}$. With each column j of $\mathcal{A}$we associate a set of column indices f, the *forbidden set* of columns that were initially structurally orthogonal to column j but are now forbidden to be merged with j. We denote the matrix resulting from the merging of columns j and l by $\mathcal{A}^{(j \leftarrow l)}$ and the matrix

where columns j and l have been made forbidden for each other by $\mathcal{A}^{(j \not\leftarrow l)}$ (in this case indices l and j are added to the respective forbidden sets of columns j and l). When a pair of columns are merged we take the union of their respective forbidden sets and assign it to the resulting column. Denote by $g(\mathcal{A})$ the minimum number of row compressions needed to determine A directly using column merging. Note that a column consisting entirely of zero does not play any further role in the compression process and hence can be ignored. The following result follows from [8].

Theorem 19 *For structurally orthogonal columns $\mathcal{A}(:, j)$ and $\mathcal{A}(:, l)$,*

$$g(\mathcal{A}) = \min\left\{g\left(\mathcal{A}^{(j \leftarrow l)}\right),\ g\left(\mathcal{A}^{(j \not\leftarrow l)}\right)\right\}.$$

The above theorem can be applied repeatedly until there are no pair of columns that are structurally orthogonal and not mutually forbidden. The merging process can be described by a binary tree of matrices where a leaf node corresponds to a compressed matrix that cannot be further processed via Theorem 19 and thereby yielding an upper bound on the number of matrix-vector products needed to determine A. The notion of structural orthogonality can be extended to segments of columns, and for certain types of matrix sparsity pattern the minimum value of p is achieved only by exploiting structural orthogonality of column segments [6].

A *row κ-partitioning* Π is a partition of $\{1, 2, \ldots, m\}$ yielding $w_1, w_2, \ldots, w_{\tilde{i}}, \ldots, w_\kappa$ where $w_{\tilde{i}}$ contains the row indices that constitute the *block* $\tilde{i}$, denoted by

$$A(w_{\tilde{i}}, :) \in R^{m_{\tilde{i}} \times n},\ \tilde{i} = 1, 2, \ldots, \kappa$$

where $m_{\tilde{i}} = |w_{\tilde{i}}|$. A segment of column j in block $\tilde{i}$ of A, denoted by $A(w_{\tilde{i}}, j), \tilde{i} = 1, 2, \ldots, \kappa$, is called a *column segment*. Unless explicitly stated the column segments in the following are not identically zero.

Definition 1 (Structurally Orthogonal Column Segments.)

- (`Same Column`)
  Column segments $A(w_{\tilde{i}}, j)$ and $A(w_{\tilde{k}}, j)$, $\tilde{i} \neq \tilde{k}$ are *structurally orthogonal*.
- (`Same Row Block`)
  Column segments $A(w_{\tilde{i}}, j)$ and $A(w_{\tilde{i}}, l)$, $j \neq l$ are *structurally orthogonal* if there does not exist $i \in w_{\tilde{i}}$ such that $a_{ij} \neq 0$ and $a_{il} \neq 0$.
- (`Different`)
  Column segments $A(w_{\tilde{i}}, j)$ and $A(w_{\tilde{k}}, l)$, $\tilde{i} \neq \tilde{k}$ and $j \neq l$ are *structurally orthogonal* if
  - $A(w_{\tilde{i}}, j)$ and $A(w_{\tilde{i}}, l)$ are *structurally orthogonal* and
  - $A(w_{\tilde{k}}, j)$ and $A(w_{\tilde{k}}, l)$ are *structurally orthogonal*.

With $\kappa = m$ the structurally orthogonal column segments $A(i, j)$ and $A(k, l)$ are said to be *isolated* [7]. We can now extend compressions of structurally orthogonal columns to structurally orthogonal column segments and an analogous result can be derived for column segments. Clearly, for structurally orthogonal column segments $(w_{\tilde{i}}, j)$ and $(w_{\tilde{k}}, l)$, $g(\mathcal{A}) = \min\{g(\mathcal{A}^{(w_{\tilde{i}}, j \leftarrow l)}), g(\mathcal{A}^{(w_{\tilde{i}}, j \not\leftarrow l)})\}$. Denote by $\gamma(A)$ the minimum number of matrix-vector products needed to determine A using a direct

method. Then the branching scheme for column segments compression where $\kappa = m$ we have

Theorem 20 $\gamma(\mathcal{A}) = \min\{\gamma(\mathcal{A}^{(w_i^z, j \leftarrow l)}), \gamma(\mathcal{A}^{(w_i^z, j \not\leftarrow l)})\}$

This result is also derived in [5] using extended graph coloring. The example below illustrates the compression algorithm with column segments (for clarity of identification of the compressions we use matrix A),

$$\begin{pmatrix} a_{11} & 0 & 0 & a_{14} & 0 & 0 \\ 0 & a_{22} & 0 & 0 & a_{25} & 0 \\ 0 & 0 & a_{33} & 0 & 0 & a_{36} \\ a_{41} & a_{42} & a_{43} & 0 & 0 & 0 \\ a_{51} & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & a_{62} & 0 & a_{64} & 0 & a_{64} \\ 0 & 0 & a_{73} & a_{74} & a_{75} & 0 \end{pmatrix} \overset{(1:2,1\leftarrow 2)}{\Rightarrow} \begin{pmatrix} a_{11} & 0 & 0 & a_{14} & 0 & 0 \\ a_{22} & 0 & 0 & 0 & a_{25} & 0 \\ 0 & 0 & a_{33} & 0 & 0 & a_{36} \\ a_{41} & a_{42} & a_{43} & 0 & 0 & 0 \\ a_{51} & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & a_{62} & 0 & a_{64} & 0 & a_{64} \\ 0 & 0 & a_{73} & a_{74} & a_{75} & 0 \end{pmatrix} \overset{(1:3,1\leftarrow 3)}{\Rightarrow} \begin{pmatrix} a_{11} & 0 & 0 & a_{14} & 0 & 0 \\ a_{22} & 0 & 0 & 0 & a_{25} & 0 \\ a_{33} & 0 & 0 & 0 & 0 & a_{36} \\ a_{41} & a_{42} & a_{43} & 0 & 0 & 0 \\ a_{51} & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & a_{62} & 0 & a_{64} & 0 & a_{64} \\ 0 & 0 & a_{73} & a_{74} & a_{75} & 0 \end{pmatrix} \overset{*}{\Rightarrow} \dots$$

$$\overset{*}{\Rightarrow} \begin{pmatrix} a_{11} & a_{11} & a_{14} & a_{14} \\ a_{22} & a_{25} & a_{22} & a_{25} \\ a_{33} & a_{36} & a_{36} & a_{33} \\ - & a_{41} & a_{42} & a_{43} \\ a_{51} & - & a_{56} & a_{55} \\ a_{62} & a_{66} & - & a_{64} \\ a_{73} & a_{75} & a_{74} & - \end{pmatrix}$$

where the first compression performed is $A^{(1:2,1\leftarrow 2)}$ and after a sequence of compressions we arrive at a matrix containing four columns. The entries marked $-$ can be ignored. It can be shown that $\gamma(A) = 4$ while using Theorem 1 a trivial upper bound on γ is 6 as no pair of columns of matrix A is structurally orthogonal.

3 Concluding Remarks

We have proposed a branching scheme for direct determination of sparse Jacobian matrices based on Zykov contraction algorithms for vertex coloring. The matrix formulation offers a more natural representation of the Jacobian determination as matrix-vector product calculation. We are currently exploring new heuristics for column segments partitioning and extension to two-sided compressions.

References

[1] A. R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *J. Inst. Math. Appl.*, 13:117–119, 1974.

[2] A. H. Gebremedhin, F. Manne, and A. Pothen. What color is your Jacobian? graph coloring for computing derivatives. *SIAM Rev.*, 47(4):629–705, 2005.

[3] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[4] S. Hossain. Cseggraph: a graph colouring instance generator. *International Journal of Computer Mathematics*, 86(10/11):1956 – 1967, 2009.

[5] S. Hossain and T. Steihaug. Optimal direct determination of sparse Jacobian matrices. Technical Report 254, Department of Informatics, University of Bergen, Norway, oct 2003. Revised version to appear in *Optimization Methods and Software*.

[6] S. Hossain and T. Steihaug. Graph coloring in the estimation of sparse derivative matrices: Instances and applications. *Discrete Appl. Math.*, 156(2):280–288, 2008.

[7] G. N. Newsam and J. D. Ramsdell. Estimation of sparse Jacobian matrices. *SIAM J. Alg. Disc. Meth.*, 4(3):404–417, 1983.

[8] A. A. Zykov. On some properties of linear complexes. *Amer. Math. Soc. Translation*, 1952(79):33, 1952.

Improved Taxation Rate for Bin Packing Games

Walter Kern, Xian Qiu

Department of Applied Mathematics, University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands

Key words: cooperative games, core, taxation rate, bin packing

1 Introduction

A *cooperative game* is defined by a tuple $\langle N, v\rangle$, where N is a set of *players* and $v : 2^N \to \mathbb{R}$ is a *value function* satisfying $v(\emptyset) = 0$. A subset $S \subseteq N$ is called a *coalition* and N itself is the *grand coalition*. The usual goal in cooperative games is to 'fairly' allocate the total gain $v(N)$ of the grand coalition N among the individual players. A well known concept is the *core* of a cooperative game (see von Neumann, Morgenstern [5]), defined by

(i) $x(N) \leq v(N)$,
(ii) $x(S) \geq v(S)$ for all $S \subseteq N$.

As usual, we abbreviate $x(S) = \sum_{i\in S} x_i$.

When the core is empty, one may want to relax the condition (ii) above in such a way that the modified core becomes nonempty. Faigle and Kern [1] introduced the multiplicative ϵ-core as follows. Given $\epsilon > 0$, the ϵ-core consists of all vectors $x \in \mathbb{R}^N$ satisfying condition (i) above together with

(ii') $x(S) \geq (1-\epsilon)v(S)$ for all subsets $S \subseteq N$.

A *bin packing game* is defined by a set of k bins, of capacity 1 each, and n items $1, 2, \cdots, n$ of sizes $a_1, a_2, \cdots, a_n$, where we assume, w.l.o.g, $0 \leq a_i \leq 1$.

Let A be the set of items and B be the set of bins. A *feasible packing* of an item set $A' \subseteq A$ into a set of bins $B' \subseteq B$ is an assignment of some (or all) elements in A' to the bins in B' such that the total size of items assigned to any bin does not exceed the bin capacity one.

The player set N consists of all items and all bins. The value $v(S)$ of a coalition $S \subseteq N$, where $S = A_S \cup B_S$ with $A_S \subseteq A$ and $B_S \subseteq B$, is the maximum value of all feasible packings of A_S into B_S. A corresponding feasible packing is called an *optimum packing*.

An intriguing problem is to find the 'minimal' taxation rate $\epsilon_{\min}$ such that the $\epsilon_{\min}$-core is nonempty for all bin packing games. It was shown in Faigle and Kern [1] that $1/7 \leq \epsilon_{\min} \leq 1/2$. Woeginger [6] improved this result to $\epsilon_{\min} \leq 1/3$. Kuipers

[3] showed that $\epsilon_{\min} = 1/7$ if all item sizes are strictly larger than $1/3$. We present an alternative proof for the result $\epsilon_{\min} \leq 1/3$ based on a straightforward greedy packing heuristic. After that, we apply the same greedy heuristic w.r.t. modified ("virtual") item sizes to derive a slightly better bound. Finally, we conjecture that $\epsilon_{min} = 1/7$ and draw the reader's attention to the connection with the well-known 3-PARTITION problem.

2 Fractional packings

A set F of items is called a *feasible set* if its total size does not exceed 1. Denote by $\mathcal{F}$ the set of all feasible sets. Let σ_F be the value of a feasible set and let $\sigma = (\sigma_F) \in \mathbb{R}^{\mathcal{F}}$ for all $F \in \mathcal{F}$, then the total earning $v(N)$ of the grand coalition N equals

$$\begin{aligned} & \max\ \sigma^T y \\ \text{s.t. } & \sum_{F \in \mathcal{F}} y_F \leq k, \\ & \sum_{F \ni i} y_F \leq 1 \quad (i = 1, 2, \cdots, n), \\ & y \in \{0,1\}^{\mathcal{F}}. \end{aligned} \tag{2.1}$$span

The value $v'(N)$ of an *optimum fractional packing* is defined by the relaxation of (2.1), i.e.,

$$\begin{aligned} & \max\ \sigma^T y \\ \text{s.t. } & \sum_{F \in \mathcal{F}} y_F \leq k, \\ & \sum_{F \ni i} y_F \leq 1 \quad (i = 1, 2, \cdots, n), \\ & y \in [0,1]^{\mathcal{F}}. \end{aligned} \tag{2.2}$$span

A *fractional packing* of our bin packing problem is a vector y satisfying the constraints of the linear program (2.2). Faigle and Kern [2] have given a sufficient and necessary condition for the non-emptiness of the ϵ-core of a bin packing game.

Lemma 5 ([2]) *The ϵ-core is nonempty if and only if $\epsilon \geq 1 - v/v'$.*

If all items are packed in a feasible integer packing, we obviously have $v' = v$, thus the core is nonempty. For convenience of description, we always ignore this trivial case. As a consequence, $v > v'/2$ can always be achieved by filling each bin to $1/2$. So the $1/2$-core is nonempty for all bin packing games. Denote by $\epsilon_N = 1 - v/v'$ the minimal taxation rate of a bin packing game N. We thus seek for good lower bounds on v/v'.

The first step in [6] is to reduce the analysis to item sizes $a_i > 1/3$. Similarly, if we aim for a bound $\epsilon_N \leq \epsilon$ with $\epsilon \in [1/4, 1/3)$, it suffices to investigate instances with item sizes $a_i > 1/4$, as can be seen from the following two lemmas:

Lemma 6 *Let A be a set of items disjoint from N and $v(N) + \sigma(A) \leq v(N \cup A)$. Then $\epsilon_{N \cup A} \leq \epsilon_N$.*

For $\delta \in (0, 1)$, let N_δ denote the restriction of N to items of size $a_i > \delta$.

Lemma 7 *If $\delta, \epsilon_{N_\delta} \leq \epsilon$, then $\epsilon_N \leq \epsilon$.*

Thus in what follows, when seeking for an upper bound $\epsilon_N \leq \epsilon$ with $\epsilon \in [1/4, 1/3)$, we may assume that all item sizes are at least $a_i > 1/4$. (This is actually a rather interesting class anyway, as it contains all instances of $3 - PARTITION$, *c.f.* section 4).

3 Greedy heuristic and modified greedy selection

Consider any bin packing game with k bins and item sizes $a_1, \cdots, a_n$ with all $a_i > 1/4$. Let $y = (y_F)$ be an optimal fractional packing. We order the support $\mathcal{F} = \{F \mid y_F > 0\}$ according to non-decreasing values of σ_F: Assume that, say, $\mathcal{F} = \{F_1, \cdots, F_m\}$ and

$$\sigma_{F_1} \geq \sigma_{F_2} \geq \cdots \geq \sigma_{F_m}.$$

Note that the number of fully packed items is at most $3k$ (3 items per bin), so that $m \leq 3k+1$. The basic idea is to construct an integral solution "greedily" as follows: Let $F_{i_1} := F_1$ and $\mathcal{F}_{i_1} := \{F \in \mathcal{F} \mid F \cap \mathcal{F}_{i_1} \neq 0\}$. Then choose the largest size feasible set F_{i_2} in $\mathcal{F} \backslash \mathcal{F}_{i_1}$, let $\mathcal{F}_{i_2} = \{F \in \mathcal{F} \backslash \mathcal{F}_{i_1} \mid F \cap F_{i_2} \neq \emptyset\}$ etc. Each F_{i_s} contains at most 3 items. Hence in each step, when removing $\mathcal{F}_{i_s}$, we remove at most 3 from the total sum $\sum_F y_F = k$, so that our construction yields $F_{i_1}, \cdots, F_{i_r}$ with $r \geq k/3$. Assuming $\sigma_F \geq 2/3$ for each F and extending our greedy selection by $k - r$ bins (each at least filled to $1/2$), we arrive at an integer packing whose value can be shown to satisfy

$$v \geq \frac{1}{3}v' + (r - \frac{k}{3})\frac{2}{3} + (k - r)\frac{1}{2} \geq \frac{2}{3}v', \tag{3.1}$$

so the $1/3$-core is nonempty. The estimate can be (slightly) improved by modifying the greedy selection so as to give higher priority to feasible sets $F \in \mathcal{F}$ with comparatively large y_F. Consider the modified ("virtual") size $\tilde{\sigma}_F := \sigma_F + \frac{1}{9}y_F$. We order the $F \in \mathcal{F}$ according to non-increasing $\tilde{\sigma}$-values, i.e.,

$$\sigma_{F_1} + \frac{1}{9}y_{F_1} \geq \sigma_{F_2} + \frac{1}{9}y_{F_2} \cdots \geq \sigma_{F_m} + \frac{1}{9}y_{F_m}$$

and apply greedy selection w.r.t the modified size. This results in a slightly improved lower bound of $\epsilon \leq 1/3 - 1/108 = 35/108$.

4 Remarks and open problems

Clearly the most straightforward problem is to determine the smallest ϵ such that all bin packing games have non-empty ϵ -core. We conjecture that $1/7$ is best possible

(*c.f.* [1] for an example showing that $\epsilon < 1/7$ is impossible and a proof that the ϵ-core is non-empty for any sufficiently large (in terms of k) bin packing game).

A further challenging conjecture due to Woeginger states that $v' - v$ is bounded by a universal constant.

We finally would like to draw the attention of the reader to the well-known $3 - PARTITION$ problem: Given a set of items of sizes $a_1, \cdots, a_{3k}$ with $1/4 < a_i < 1/2$ and k bins, can we pack all items? If the fractional optimum is less than k, the answer is clearly "no". Note that the fractional optimum can be computed efficiently as there are only $O(k^3)$ feasible sets. Thus if $P \neq NP$, then there must be instances with fractional optimum equal to k and integral optimum $< k$. Although we tried hard, we could not exhibit a single such instance.

References

[1] U. Faigle and W. Kern. *On some approximately balanced combinatorial cooperative games*. *Mathods and Models of Operation Research, 38:141-152, 1993*.

[2] U. Faigle and W. Kern. *Approximate core allocation for binpacking games*. *SIAM J. Discrete Math, 11:387-399, 1998*.

[3] J. Kuipers. Bin packing games. *Mathematical Methods of Operations Research, 47:499-510, 1998*.

[4] L. Lovász and M. Plummer. *Matching Theory. North-Holland Math, Amsterdam, 1986*.

[5] J. V. Neumann and O. Morgenstern. *Theory of Games and Economic Behavior. Princeton University, Princeton, 1947*.

[6] G. J. Woeginger. *On the rate of taxation in a cooperative bin packing game*. *Mathematical Methods of Operations Research, 42:313-324, 1995*.

Decomposition of Tutte Polynomial

Martin Kochol

MÚ SAV, Stefánikova 49, 814 73 Bratislava, Slovakia,
martin.kochol@mat.savba.sk

Key words: Tutte polynomial, determinant, partition, edge cut

1 Introduction

Tutte polynomial is an important invariant encoding many properties of graphs (see [1,5]). Computing of the polynomial is a $\sharp P$-complete problem (see [2,3]) but can be evaluated in polynomial time for some special classes of graphs [4]. We show that if an n-edge cut divides a graph into two subgraphs, then the Tutte polynomial of the original graph can be fully expressed by Tutte polynomials of these parts. We give a formula consisting from a determinant of size $1 + b_n$ (where b_n is the nth Bell number). The edge cut induces a bipartite graph with partition of vertices of size p and p'. Our second formula uses determinat of size $1 + \min\{p, p'\}$. This improvement can be significant if one of p or p' is much smaller then n. Thus, from computational point of view, we can effectively evaluate the Tutte polynomial of a graph with small vertex cut if we know these polynomials on the two parts.

2 A general formula

If $G = (E, G)$ is a graph and $A \subseteq E$, then $r(A) = |V| - c(A)$, where $c(A)$ is the number of components of the graph (V, A). The Tutte polynomial of G is $T_G(x, y) = \sum_{A \subseteq E}(x-1)^{r(E)-r(A)}(y-1)^{|A|-r(A)}$. We need to consider its rational version $T'_G(x, y) = (x-1)^{-r(E)}T_G(x, y)$.

Denote by B_n the set of partitions of $\{1, \ldots, n\}$, $|B_n| = b_n$. If $P = \{Q_1, \ldots, Q_p\}$, $P' = \{Q'_1, \ldots, Q'_{p'}\} \in B_n$ and for each Q_k there exists $Q'_{k'}$ such that $Q_k \subseteq Q'_{k'}$, then $P \leq P'$. Denote by $B_{P,n} = \{P' \in B_n; P \leq P'\}$.

Let $C = \{e_1, \ldots, e_n\}$ be an edge cut of a graph G and $G - C$ has components G_1 and G'_1. Let $e_i = v_i v'_i$, $v_i \in V(G_1)$, $v'_i \in V(G'_1)$ for $i = 1, \ldots, n$. Denote by P_C (P'_C) the permutation from B_n so that $i, j \in \{1, \ldots, n\}$ belong to one set in P_C (P'_C) if and only if $v_i = v_j$ ($v'_i = v'_j$).

⋆ Partially supported by grant VEGA 2/0118/10

Let H be the graph arising from G_1 after adding n new vertices $u_1, \ldots, u_n$ and edges $u_1v_1, \ldots, u_nv_n$. Consider $P = \{Q_1, \ldots, Q_p\} \in B_n$. For $k = 1, \ldots, p$, identify the set of vertices $\{u_q; q \in Q_k\}$ into a new vertex x_k and denote the resulting graph by $[H, P]$. In a similar way construct $[H', P]$ from G'_1.

Take n isolated edges $w_1w'_1, \ldots, w_nw'_n$ and partitions $P = \{Q_1, \ldots, Q_p\}$, $P' = \{Q'_1, \ldots, Q'_{p'}\} \in B_n$. For $k = 1, \ldots, p$ and $k' = 1, \ldots, p'$, identify the sets of vertices $\{w_q; q \in Q_k\}$ and $\{w'_{q'}; q' \in Q'_{k'}\}$ into new vertices y_k and $y'_{k'}$, respectively, and denote the resulting graph by $[P, P']$.

Let $B_n = \{P_{n,1}, \ldots, P_{n,b_n}\}$ and $M_n(x, y)$ be the symmetric $b_n \times b_n$ matrix with the (i, j)-entry $T'_{[P_{n,i},P_{n,j}]}(x, y)$. Let $M_n^{H,H'}(x, y)$ arise from $M_n(x, y)$ after adding a row and column so that the $(b_n + 1, b_n + 1)$-entry is $T'_G(x, y)$, the $(i, b_n + 1)$-entry is $T'_{[H',P_{n,i}]}(x, y)$ and the $(b_n + 1, i)$-entry is $T'_{[H,P_{n,i}]}(x, y)$. If $B, B' \subseteq B_n$, then $M_{B,B'}(x, y)$ ($M_{B,B'}^{H,H'}(x, y)$) denotes the submatrix of $M_n(x, y)$ ($M_n^{H,H'}(x, y)$) arising after removing the rows and columns corresponding to the permutations not belonging to B and B', respectively. Let $M_n'^{H,H'}(x, y)$ and $M_{B,B'}'^{H,H'}(x, y)$ arise from $M_n^{H,H'}(x, y)$ and $M_{B,B'}^{H,H'}(x, y)$), respectively, after replacing $T'_G(x, y)$ by 0 in the entry of the last row and the last column.

Theorem 21 *Let G be a graph with an n-edge cut C, and $B_{P_C,n} \subseteq B \subseteq B_n$, $B_{P'_C,n} \subseteq B' \subseteq B_n$. Let $\tilde{B} \subseteq B$, $\tilde{B}' \subseteq B'$ so that $M_{\tilde{B},\tilde{B}'}(x, y)$ is a maximal regular submatrix of $M_{B,B'}(x, y)$. Then $|M_{\tilde{B},\tilde{B}'}^{H,H'}(x, y)| = 0$, i.e., $T'_G(x, y) = -|M_{\tilde{B},\tilde{B}'}'^{H,H'}(x, y)| \cdot |M_{\tilde{B},\tilde{B}'}(x, y)|^{-1}$.*

3 Two specific formulas

We can prove that $M_n(x, y)$ is regular, whence we have the following.

Theorem 22 *Let C be an n-edge cut of a graph G. Then $|M_n^{H,H'}(x, y)| = 0$, i.e., $T'_G(x, y) = -|M_n'^{H,H'}(x, y)| \cdot |M_n(x, y)|^{-1}$.*

Let $\tilde{P} \in B_n$, $\tilde{P} \leq P_C$. Then $P_C = \{Q_1, \ldots, Q_p\}$, and $\tilde{P} = \{Q_{1,1}, \ldots, Q_{1,k_1}, \ldots, Q_{p,1}, \ldots, Q_{p,k_p}\}$, such that $Q_{i,1}, \ldots, Q_{i,k_i}$ are subsets of Q_i for $i = 1, \ldots, p$. Denote by $B_{\tilde{P},C}$ the sets of permutations from B_n arising from $\tilde{P}$ after unifying some sets from $\{Q_{1,1}, \ldots, Q_{1,p}\}$. Then $|B_{\tilde{P},C}| = |B_{P_C,n}| = b_p$ and we can prove that $M_{B_{P_C,n},B_{\tilde{P},C}}(x, y)$ is regular. This implies the following.

Theorem 23 *Let C be an n-edge cut of a graph G and $\tilde{P} \in B_n$, $\tilde{P} \leq P_C, P'_C$. Then $|M_{B_{P_C,n},B_{\tilde{P},C}}^{H,H'}(x, y)| = 0$, in other words $T'_G(x, y) = -|M_{B_{P_C,n},B_{\tilde{P},C}}'^{H,H'}(x, y)| \cdot |M_{B_{P_C,n},B_{\tilde{P},C}}(x, y)|^{-1}$.*

References

[1] T. Brylawski and J. Oxley, *The Tutte polynomial and its applications*, in: N. White (Ed.), Matroid Applications, Cambridge University Press, Cambridge (1992), pp. 123-225.

[2] L.A. Goldberg and M. Jerrum, *Inapproximability of the Tutte polynomial*, in: STOC'07: Proceedings of the 39th anual ACM symposium on Theory of computing, ACM, New York, (2007), pp. 459–468.
[3] F. Jaeger, D.L. Vertigan, and D.J.A. Welsh, *On the computational complexity of the Jones and Tutte polynomials*, Math. Proc. Cambridge Philos. Soc. **108** (1990) 35–53.
[4] S.D. Noble, *Evaluating the Tutte polynomial for graphs of bounded tree-width*, Combin. Prob. Comp. **7** (1998) 307-321.
[5] D.J.A. Welsh, *Complexity: Knots, Colourings and Counting*, Cambridge University Press, Cambridge (1993).

Approximability of the Two-Stage Knapsack problem with discretely distributed weights

Stefanie Kosuch

Institutionen för datavetenskap (IDA), Linköpings Universitet, Sweden
stefanie.kosuch@liu.se

Key words: stochastic knapsack problem, multidimensional knapsack, two-stage, non-approximability, approximation algorithms

1 Introduction and Mathematical Formulation

The knapsack problem is a widely studied combinatorial optimization problem. It consists in choosing among a set of items a subset such that the total weight of the chosen items respects a given weight restriction (the capacity of the knapsack) while the total reward of the chosen items is maximized. The most common applications arise in fields where some capacity has to be respected (storage, transport, packing, network optimization...) or where the decision maker has to handle limited resources (recourse allocation, cutting stock problems...). However, knapsack problems also serve as subproblems in less obvious fields of application such as cryptography or finance.

As in many applications the decision maker has to face uncertainty in the involved parameters, more and more studies are made on various settings of the *Stochastic Knapsack problem*, where some of the parameters are assumed to be random (i.e. not exactly known in the moment the (pre-)decision has to be made). In this paper we restrict our study to the case where the weights are assumed to be random. Moreover, we assume that the decision can be made in two stages: A *pre*-decision is made while the item weights are still unknown, i.e. the decision maker assigns some items to the knapsack without knowing their exact weights. In this *first stage* we obtain a certain reward for the added items. Then, once the weights of all items have come to be known, we can make a *corrective* decision (*second stage*): If additional items are added, the reward obtained for these items is smaller than it would have been in the first stage. And if items are removed, a penalty has to be paid that is naturally strictly greater than the received first-stage reward. The objective is to maximize the first-stage reward plus the expected second-stage gain, where the latter is composed of the reward obtained for added items minus the penalty paid for removed ones. We call the resulting problem *Two-Stage Knapsack problem*. Its mathematical formulation is as follows:

$$(TSKP) \quad \max_{x\in\{0,1\}^n} \quad \sum_{i=1}^{n} r_i x_i + \mathbb{E}[\mathcal{Q}(x,\chi)] \tag{1.1a}$$

$$\text{s.t.} \quad \mathcal{Q}(x,\chi) = \max_{y^+,y^-\in\{0,1\}^n} \sum_{i=1}^{n} \overline{r}_i y_i^+ - \sum_{i=1}^{n} d_i y_i^- \tag{1.1b}$$

$$\text{s.t. } y_i^+ \leq 1 - x_i, \quad \forall\, i = 1,\dots,n, \tag{1.1c}$$

$$y_i^- \leq x_i, \quad \forall\, i = 1,\dots,n, \tag{1.1d}$$

$$\sum_{i=1}^{n} (x_i + y_i^+ - y_i^-)\chi_i \leq c. \tag{1.1e}$$

where x is the first-stage decision vector and $r > 0$ the first-stage reward vector, both of dimension n. The weight of item i is represented by the random variable χ_i. The second-stage binary decision vector y^+ models the adding of items while the decision vector y^- models their removal. If item i is added in the second stage we receive a second-stage reward $\overline{r}_i < r_i$, and if it is removed we have to pay a penalty $d_i > r_i$. An item can only be added if it had not been added in the first stage (constraint (1.1c)) and removed if it has been added previously (constraint (1.1d)). In the end, the items (remaining) in the knapsack need to respect the knapsack capacity $c > 0$ (constraint (1.1e)).

To the best of our knowledge there have only been two previous studies of Two-Stage Knapsack problems: In [4] the authors study a Two-Stage Knapsack problem with probability constraint in the first stage where the item weights are assumed to be normally distributed. The main difficulty in this case arises from the question of how to evaluate the objective function exactly. The authors therefore propose upper and lower bounds and apply a branch-and-bound algorithm to search the first-stage solution space for the best such lower bound. In [2] the authors study Static as well as Two-Stage Quadratic Knapsack problems with chance-constraint. The authors assume a finite distribution for the weight vector which allows them to reformulate the studied problems in a deterministic equivalent form. The authors propose semi-definite relaxations to obtain good upper bounds in reasonable time.

As in [2] we assume in this paper that the weight vector only admits a finite number of outcomes (*scenarios*) with non-zero probabilities. This allows to reformulate the $TSKP$ deterministically (see e.g. [2]). In fact, in [3] it has been shown that a stochastic combinatorial optimization problem can, under some mild assumptions, be approximated to any desired precision by replacing the underlying distribution by a finite random sample. However, to obtain a good approximation the used set of random samples needs generally to be rather large. Moreover, if the weights are e.g. independently, discretely distributed, the number of scenarios might grow exponentially with the number of items. Solving the obtained deterministic equivalent problem to optimality is thus generally not tractable. That is why we are interested in the approximability of the $TSKP$ with discretely distributed weights. Note that, like its deterministic counterpart, the Two-Stage Knapsack problem is $\mathcal{NP}$-hard. Moreover, it has been shown in [1] that two-stage stochastic integer programming problems with discretely distributed parameters are even $\sharp\mathcal{P}$-hard. In the second section we state a non-approximability result for the $TSKP$ and give a sketch of

the proof. This is followed by three positive approximation results.

2 (Non)-Approximation results

Theorem 2.1 *For any $\epsilon > 0$, there exists no polynomial-time $K^{-\frac{1}{2}+\epsilon}$ - approximation algorithm for the $TSKP$, unless $\mathcal{P} = \mathcal{NP}$.*

Sketch of the proof: The idea of the proof is as follows: Basically we do a reduction from the multi-dimensional knapsack problem ($MDKP$). In [5] the authors prove that, for all $\epsilon > 0$, the $MDKP$ does not admit a polynomial-time $m^{-\frac{1}{4}+\epsilon}$-approximation algorithm (where m is the number of constraints) unless $\mathcal{P} = \mathcal{NP}$. This is proven by a reduction from the maximum clique problem. In their paper the authors use that the maximum clique problem cannot be approximated within a factor $n^{-\frac{1}{2}+\epsilon}$, for any $\epsilon > 0$, where n stands for the number of vertices. A newer result however states that it is even $\mathcal{NP}$-hard to approximate the maximum clique problem within a factor $n^{-1+\epsilon}$.

Instead of giving a direct reduction from the $MDKP$ to the $TSKP$, we first show how the optimal solution value of the $MDKP$ can be obtained by solving a special variant of the $TSKP$ where items can only be added in the second-stage (called $AddTSKP$). Note that this is not done by an equivalent reformulation: In fact, the reduction is such that the integer part of the solution value of the $AddTSKP$ instance gives us the optimal solution value of the initial $MDKP$ instance. However, the optimal first-stage solution of the former is optimal solution for the latter. The number of scenarios in the obtained $AddTSKP$ instance equals the number of constraints of the $MDKP$. In the second step we show that for any instance of the $AddTSKP$ there exists an instance of the $TSKP$ with same number of scenarios, identical optimal solution value and such that an optimal solution of the $TSKP$ instance is optimal solution of the $AddTSKP$ instance, and vice versa. The last step consists in proving that these polynomial reductions preserve the non-approximability result for the $MDKP$. □

Proposition 2.2 *For an instance of the $TSKP$ define $\alpha := \min_{i=1\ldots,n} \frac{\overline{r}_i}{r_i}$. Then adding no items in the first stage yields a solution whose solution value is at least an α-fraction of the optimal solution value.*

Idea of the proof: First of all note that $\alpha < 1$. The idea is to first replace, for all i, the second-stage reward $\overline{r}_i$ by $\alpha \cdot r_i$. The optimal solution value of the new instance is thus a lower bound on the optimal solution value of the initial instance. Then adding no item at all in the first stage yields a solution for the obtained instance with approximation ratio at most α. This approximation ratio is thus also valid for the initial instance.

Proposition 2.3 *Under the assumption of a polynomial scenario model, there exists a polynomial-time $\frac{1}{n}$-approximation algorithm for the $TSKP$.*

Underlying algorithm: For all $i = 1, \ldots, n$ let R_i denote the maximum ex-

pected reward we can obtain for item i. Let $\mathcal{K}_i = \{k \in \{1,\dots,K\} : \chi_i^k \leq c\}$ where K is the number of scenarios and χ_i^k ($k = 1,\dots,n$) are the outcomes of the random variable χ_i. Let $p^k > 0$ be the probability of scenario k. It follows that $R_i = \max\{r_i - \sum_{k\notin\mathcal{K}_i} p^k d_i, \sum_{k\in\mathcal{K}_i} p^k \overline{r}_i\}$. Let $j = \arg\max_{i=1,\dots,n} R_i$. If $R_j = r_j - \sum_{k\notin\mathcal{K}_j} p^k d_j$, set $x_j = 1$ and $x_i = 0$ for all $i \neq j$, otherwise set $x_i = 0$ for all $i = 1,\dots,n$. This clearly yields a solution with approximation ratio $\frac{1}{n}$. However, in order to determine j in polynomial time, K needs to be polynomial in n.

Proposition 2.4 *Let K-$AddTSKP$ (K-$MDKP$) denote the variant of the $AddTSKP$ ($MDKP$) where the number of scenarios (constraints) is fixed to be K. Then, for a given $\epsilon > 0$, there exists a polynomial-time approximation algorithm for the K-$AddTSKP$ with approximation-ratio $\frac{1}{2} - \epsilon$.*

Idea of the underlying algorithm: First, solve the first-stage problem as a K-$MCKP$ (i.e. the solution has to respect the K second-stage capacity constraints) using a $PTAS$. Then, solve independently the K second-stage knapsack problems using the well known $FTPAS$ and compute the expectation of the obtained solution values (based on the corresponding probabilities of the scenarios). The associated solution is to add no item at all in the first stage. Compare the two solutions and output the better.

References

[1] Dyer, M., Stougie, L.: *Computational complexity of stochastic programming problems*. Mathematical Programming 106(3), 423–432 (2006)

[2] Gaivoronski, A.A., Lisser, A., Lopez, R., Hu, X.: *Knapsack problem with probability constraints*. Journal of Global Optimization 49(3), 397–413 (2010)

[3] Kleywegt, A.J., Shapiro A., Homem-de-Mello, T.: *The sample average approximation method for stochastic discrete optimization*. SIAM Journal on Optimization 12(2), 479–502 (2002)

[4] Kosuch, S., Lisser, A.: *On two-stage stochastic knapsack problems*. Discrete Applied Mathematics (In Press, Corrected Proof) (2010)

[5] Li'ang, Z., Yin, Z.: *Approximation for knapsack problems with multiple constraints*. Journal of Computer Science and Technology 14(4), 289–297 (1999)

Computing the Grothendieck constant of some graph classes

M. Laurent, [a] A. Varvitsiotis [b]

[a] *Centrum Wiskunde & Informatica (CWI), Amsterdam, and Tilburg University.*

[b] *Centrum Wiskunde & Informatica (CWI), Amsterdam.*

1 Introduction

Given a simple loopless graph $G = ([n], E)$, define $\mathcal{M}(G)$ to be the set of symmetric $n \times n$ matrices with zero diagonal that are supported by G, i.e., $A_{ii} = 0$ for $i \in [n]$ and $A_{ij} = 0$ for $(i, j) \notin E$. For $A \in \mathcal{M}(G)$, consider the integer quadratic program over the discrete hypercube

$$\text{ip}(G, A) := \max_{x \in \{\pm 1\}^n} \frac{1}{2} x^T A x = \max_{x \in \{\pm 1\}^n} \sum_{ij \in E} A_{ij} x_i x_j, \tag{1.1}$$

whose canonical semidefinite programming relaxation is

$$\text{sdp}(G, A) := \max_{u_1, \ldots, u_n \in S^{n-1}} \sum_{ij \in E} A_{ij} u_i^T u_j, \tag{1.2}$$

where S^{n-1} denotes the n-dimensional unit sphere. Let $\kappa(G)$ denote the integrality gap of relaxation (1.2), defined by

$$\kappa(G) = \sup_{A \in \mathcal{M}(G)} \frac{\text{sdp}(G, A)}{\text{ip}(G, A)}. \tag{1.3}$$

In other words, $\kappa(G)$ is the smallest constant $K > 0$ for which the inequality

$$\max_{u_1, \ldots, u_n \in S^{n-1}} \sum_{ij \in E} A_{ij} u_i^T u_j \leq K \max_{x \in \{\pm 1\}^n} \sum_{ij \in E} A_{ij} x_i x_j \tag{1.4}$$

holds for all matrices $A \in \mathcal{M}(G)$. Following Alon et al. [1], $\kappa(G)$ is called the *Grothendieck constant* of G. It is shown in [1] that $\Omega(\log \omega(G)) = \kappa(G) = O(\log \vartheta(\bar{G}))$ for any graph G. Here, $\omega(G)$ denotes the maximum size of a clique in G and $\vartheta(\bar{G})$ is the Lovász theta function of $\bar{G}$, shown in [9] to lie between $\omega(G)$ and the chromatic number $\chi(G)$ of G.

On the other hand, an important inequality of A. Grothendieck states that there exists a constant $K > 0$ such that

$$\max_{\substack{u_1,\dots,u_m\in S^{m+n-1}\\ v_1,\dots,v_n\in S^{m+n-1}}} \sum_{i=1}^{m}\sum_{j=1}^{n} B_{ij}u_i^T v_j \le K \max_{\substack{x\in\{\pm1\}^n\\ y\in\{\pm1\}^n}} \sum_{i=1}^{m}\sum_{j=1}^{n} B_{ij}x_i y_j, \tag{1.5}$$

holds for all matrices $B \in \mathbb{R}^{m\times n}$ [7]. The smallest such constant is known as *Grothendieck's constant* and is denoted by K_G. Note that this fits into the general framework of (1.1), since (1.5) is equivalent to

$$\text{sdp}(K_{m,n}, A) \le K \text{ ip}(K_{m,n}, A),$$

where $K_{m,n}$ is the complete bipartite graph on $n+m$ nodes and $A = \begin{pmatrix} 0 & \frac{1}{2}B \\ \frac{1}{2}B^T & 0 \end{pmatrix}$.

In particular, Grothendieck's inequality states that the integrality gap of the canonical semidefinite relaxation of (1.1), when restricted to bipartite graphs, is constant. It is known that $K_G \le \pi[2\lambda_{1,1}(1+\sqrt{2})]^{-1} \sim 1.782$ [8], and that $K_G \ge 1.6769...$ [10], but its exact value remains unknown.

We point out some tight connections between problem (1.1) and the *max-cut problem* which, given edge weights $w \in \mathbb{R}^E$, asks for the maximum weight $\text{mc}(G,w)$ of a cut in G. Then, $\text{mc}(G,w) = \max_{x\in\{\pm1\}^n} \frac{1}{2}(w(E) - x^T A_w x) = \max_{x\in\{\pm1\}^n} \frac{1}{4}x^T L_w x$, where A_w is the weighted adjacency matrix of G and L_w is its Laplacian matrix. Hence, $\text{mc}(G,w) = \frac{1}{2}(w(E)+\text{ip}(G,-A_w))$ and $\text{sdp}_{GW}(G,w) = \frac{1}{2}(w(E) + \text{sdp}(G,-A_w))$ is the canonical semidefinite programming relaxation of max-cut considered in [6]. The Grothendieck constant of G can be used to bound the integrality gap for max-cut since, assuming $w(E) \ge 0$, we have that $\text{sdp}_{GW}(G,w) \le \kappa(G)\cdot\text{mc}(G,w)$. If, moreover, $w \ge 0$, then L_G is positive semidefinite and, as observed in [1], $\text{mc}(G,w) = \text{ip}(G,A)$, $\text{sdp}_{GW}(G,w) = \text{sdp}(G,A)$, where A is the matrix with bipartite pattern $K_{n,n}$ and off-diagonal block $B = L_w/4$. Thus the Grothendieck constant K_G (for bipartite graphs) bounds the integrality gap for max-cut, although it provides a bound less tight than the celebrated bound $1/0.878 \sim 1.139$ of [6].

During the last couple of years there has been increased interest in Grothendieck type inequalities, since they proved to be extremely useful for an abundance of applications, most notably in approximation algorithms, optimization and quantum information theory (see e.g. [1,4]).

In this paper we study the Grothendieck constant $\kappa(G)$ for some specific graph classes. It turns out that for K_5-minor-free graphs, we can compute it exactly, whereas for some other graph classes we can obtain tight upper bounds.

2 Geometric reformulation and behavior under graph operations

We begin with a geometric reformulation for the Grothendieck constant $\kappa(G)$. For this, define the matrix sets $\mathcal{E}_n := \{X \in \mathcal{S}_n^+ \mid X_{ii} = 1 \ \forall i \in [n]\}$ and $\text{CUT}_n := \text{conv}(X \in \mathcal{E}_n \mid \text{rank}X = 1\}$, where $\mathcal{S}_n^+$ is the cone of positive semidefinite matrices. Moreover, if π_E denotes the projection from $\mathbb{R}^{n\times n}$ onto the subspace

$\mathbb{R}^E$ indexed by the edge set of G, define $\mathcal{E}(G) := \pi_E(\mathcal{E}_n)$, $\mathrm{CUT}(G) := \pi_E(\mathrm{CUT}_n)$, known respectively as the *elliptope* and the *cut polytope* of G (see e.g. [5] for a detailed study of these objects). Clearly, $\mathrm{CUT}(G) \subseteq \mathcal{E}(G)$. The following lemma shows that that $\kappa(G)$ is the smallest dilation of the cut polytope containing the elliptope.

Lemma 2.1 *For a graph G, $\kappa(G)$ is equal to the smallest constant $K > 0$ for which*

$$\mathcal{E}(G) \subseteq K \cdot \mathrm{CUT}(G).$$

As an easy application we can see that $\kappa(G) = 1$ iff G is a forest. Moreover, it is easy to verify that $\kappa(G) = \max_H K(H)$, where the maximum is taken over all subgraphs H of G supporting a facet defining inequality of $\mathrm{CUT}(G)$.

We mention some results concerning the behavior of $\kappa(G)$ under graph operations. Clearly, $\kappa(G)$ is monotone nondecreasing with respect to edge deletion. On the other hand, $\kappa(G)$ is not monotone with respect to edge contraction, since $\kappa(K_2) = 1 < \kappa(C_3) = 3/2$, while $\kappa(C_4) = \sqrt{2} < \kappa(C_3) = 3/2$, where C_n denotes the circuit graph on n nodes.

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ for which $V_1 \cap V_2$ is a clique of size k in both G_1 and G_2, the graph $G = (V_1 \cup V_2, E_1 \cup E_2)$ is called the *clique k-sum* of G_1 and G_2. The next lemma uses a result from [2], that gives the description of $\mathrm{CUT}(G)$, when G is a clique k-sum with $k \leq 3$ in terms of the descriptions of the cut polytopes of the component graphs. As an easy application, we obtain $\kappa(K_{3,n}) \leq 3/2$ and $\kappa(C_n) \leq 3/2$.

Lemma 2.2 *Assume G is the clique k-sum of G_1 and G_2 and $k \leq 3$. Then, $\kappa(G) = \max(\kappa(G_1), \kappa(G_2))$.*

3 Exact computation and bounds for $\kappa(G)$ for some graph classes

We start with graphs with no K_5-minor, for which we can give a formula for their Grothendieck constant as a function of their girth, i.e., the minimum length of a circuit in the graph. As a consequence, in this case, $\kappa(G)$ can be computed in polynomial time.

A well known result from [2] states that if G has no K_5-minor, then its cut polytope $\mathrm{CUT}(G)$ is defined by the inequalities $|x_e| \leq 1$ for $e \in E$, and $-x(F)+x(C \setminus F) \leq |C| - 2$ for all cycles C in G and all odd subsets $F \subseteq C$. Any vector $x \in [-1, 1]^E$ can be parametrized as $x = \cos(\pi a)$ where $a \in [0, 1]^E$; [3] shows that $x \in \mathcal{E}(C_n)$ iff a satisfies the inequalities $a(F) - a(C_n \setminus F) \leq |F| - 1$ for all odd subsets F of C_n. The above descriptions for $\mathrm{CUT}(G)$ (when G has no K_5-minor) and for $\mathcal{E}(C_n)$ are the basic ingredients for the following results.

Theorem 3.1 *The Grothendieck constant of a circuit $C_n, (n \geq 3)$ is equal to*

$$\kappa(C_n) = \frac{n}{n-2} \cos\left(\frac{\pi}{n}\right).$$

Corollary 2 *If G is a graph with no K_5-minor (and G is not a forest), then*

$$\kappa(G) = \frac{g}{g-2}\cos\left(\frac{\pi}{g}\right),$$

where g is the minimum length of a circuit in G.

As an application, $\kappa(K_{2,2}) = \kappa(K_{2,n}) = \kappa(K_{3,3}) = \kappa(K_{3,n}) = \sqrt{2}$; $\kappa(C_{2n})$ was computed in [11] in the context of quantum information theory.

Moreover, we can bound the Grothendieck constant for graphs whose cut polytope is defined by inequalities supported by at most k points. Let $\mathcal{R}_k(K_n)$ denote the polyhedron in $\mathbb{R}^{E_n}$ defined by all the valid inequalities for $\mathrm{CUT}(K_n)$ that are supported by at most k points, and define its projection $\mathcal{R}_k(G) := \pi_E(\mathcal{R}_k(K_n))$ for any graph G on n vertices.

Theorem 3.2 *For any graph G, we have that $\mathcal{E}(G) \subseteq \kappa(K_k) \cdot \mathcal{R}_k(G)$.*

Define $\mathcal{G}_k$ to be the class of graphs for which $\mathrm{CUT}(G) = \mathcal{R}_k(G)$. For instance, $\mathcal{G}_2$ is the class of graphs with no K_3-minor (all forests), while $\mathcal{G}_3 = \mathcal{G}_4$ is the class of graphs with no K_5-minor. Clearly, for $G \in \mathcal{G}_2$, $\kappa(G) = \kappa(K_2) = 1$; we saw above that, for $G \in \mathcal{G}_3$, $\kappa(G) \leq \kappa(K_3) = 3/2$. More generally, we have:

Corollary 3 *For any graph $G \in \mathcal{G}_k$, we have that $\kappa(G) \leq \kappa(K_k)$. This bound is tight since $K_k \in \mathcal{G}_k$.*

References

[1] N. Alon, K. Makarychev, Y. Makarychev, and A. Naor. Quadratic forms on graphs. *Invent. Math.*, 163(3): 499–522, 2006.

[2] F. Barahona. The max-cut problem on graphs not contractible to K_5. *Oper. Res. Lett.*, 2: 107–111, 1983.

[3] W.W. Barrett, C.R. Johnson, and P. Tarazaga. The real positive definite completion problem for a simple cycle. *Linear Algebra Appl.*, 192:3–31, 1993.

[4] J. Briët, H. Buhrman and B. Toner. A generalized Grothendieck inequality and entangled XOR games. Preprint, arXiv:0901.2009 [quant-ph].

[5] M.M. Deza and M. Laurent. *Geometry of Cuts and Metrics*, Springer, 1997.

[6] M.X. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42:1115–1145, 1995.

[7] A. Grothendieck. Résumé de la théorie métrique des produits tensoriels topologiques. *Bol. Soc. Mat. Sao Paolo*, 8:1–79, 1953.

[8] J. Krivine. Sur la constante de Grothendieck. *Comptes Rendus de l'Académie des Sciences de Paris Series A-A*, 284:445–446, 1977.

[9] L. Lovász. On the Shannon capacity of a graph. *IEEE Trans. Inf. Theory*, 25(1):1–7, 1979.

[10] J.A. Reeds. A new lower bound on the real Grothendieck constant. Unpublished note, 1991. Avalaible at `http://www.dtc.umn.edu/~reedsj/bound2.dvi`

[11] S. Wehner. Tsirelson bounds for generalized CHSH inequalities. *Physical Review A*, 73:022110, 2006.

Branch-and-Prune trees with bounded width

Leo Liberti, [a] Benoît Masson, [b] Carlile Lavor, [c] Antonio Mucherino [b]

[a] *LIX, École Polytechnique, 91128 Palaiseau, France*
liberti@lix.polytechnique.fr

[b] *IRISA, INRIA, Campus de Beaulieu, 35042 Rennes, France*
{benoit.masson, antonio.mucherino}@inria.fr

[c] *Dept. of Applied Maths (IMECC-UNICAMP), State Univ. of Campinas, C.P. 6065, 13081-970, Campinas - SP, Brazil*
clavor@ime.unicamp.br

Key words: DMDGP, distance geometry, order, reflection, symmetry.

1 Introduction

The MOLECULAR DISTANCE GEOMETRY PROBLEM, which asks to find the embedding in $\mathbb{R}^3$ of a given weighted undirected graph, is a good model for determining the structure of proteins given a set of inter-atomic distances [6,4]. Its generalization to $\mathbb{R}^K$ is called DISTANCE GEOMETRY PROBLEM (DGP), which has applications in wireless sensor networks [2] and graph drawing. In general, the MDGP and DGP implicitly require a search in a continuous Euclidean space. Proteins, however, have further structural properties that can be exploited to define subclasses of instances of the MDGP and DGP whose solution set is finite [5]. These instances can be solved with an algorithmic framework called Branch-and-Prune (BP) [3,5]: this is an iterative algorithm where the i-th atom of the protein can be embedded in $\mathbb{R}^3$ using distances to at least three preceding atoms. Since the intersection of three 3D spheres contains in general two points, the BP gives rise to a binary search tree. In the worst case, the BP is an exponential time algorithm, which is fitting because the MDGP and DGP are **NP**-hard [9].

Compared to continuous search algorithms, the performance of the BP algorithm is impressive from the point of view of both efficiency and reliability. In this paper we try to explain why the BP algorithm is so much faster than other approaches notwithstanding its worst-case exponential running time. Specifically, using the particular structure of the protein graph, we argue that it is reasonable to expect that the BP will yield a search tree of bounded width.

2 Discretizable instances and the BP algorithm

For all integers $n > 0$, we let $[n] = \{1, \ldots, n\}$. Given an undirected graph $G = (V, E)$ with $|V| = n$, for all $v \in V$ we let $N(v) = \{u \in V \mid \{u, v\} \in E\}$ be the set of vertices *adjacent* to v. Given a positive integer K, an *embedding* of G in $\mathbb{R}^K$ is a function $x : V \to \mathbb{R}^K$. If $d : E \to \mathbb{R}_+$ is a given edge weight function on $G = (V, E, d)$, an embedding is *valid* for G if $\forall \{u, v\} \in E \; \|x_u - x_v\| = d_{uv}$, where $x_v = x(v)$ for all $v \in V$ and $d_{uv} = d(\{u, v\})$ for all $\{u, v\} \in E$. For any $U \subseteq V$, an embedding of $G[U]$ (i.e. the subgraph of G induced by U) is a *partial embedding* of G. If x is a partial embedding of G and y is an embedding of G such that $\forall u \in U \, (x_u = y_u)$ then y is an *extension* of x. For a total order $<$ on V and for each $v \in V$, let $\rho(v) = |\{u \in V \mid u \leq v\}|$ be the *rank* of v in V with respect to $<$. The rank is a bijection between V and $[n]$, so we can identify v with its rank and extend arithmetic notation to V so that for $i \in \mathbb{Z}$, $v + i$ denotes the vertex $u \in V$ with $\rho(u) = \rho(v) + i$. For all $v \in V$ and $\ell < \rho(v)$ we denote by $\gamma_\ell(v)$ the set of ℓ immediate predecessors of v. If $U \subseteq V$ with $|U| = h$ such that $G[U]$ is a clique, let $D'(U)$ be the symmetric matrix whose (u, v)-th component is d_{uv}^2 for $u, v \in U$, and let $D(U)$ be $D'(U)$ bordered by a left $(0, 1, \ldots, 1)^\top$ column and a top $(0, 1 \ldots, 1)$ row (both of size $h + 1$). Then the Cayley-Menger formula [1] states that the volume in $\mathbb{R}^{h-1}$ of the h-simplex defined by $G[U]$ is given by $\Delta_{h-1}(U) = \sqrt{\frac{(-1)^h}{2^{h-1}((h-1)!)^2}|D(U)|}$.

Generalized DISCRETIZABLE MOLECULAR DISTANCE GEOMETRY PROBLEM ($^{\mathsf{K}}$DMDGP). Given an integer $K > 0$, a weighted undirected graph $G = (V, E, d)$ with $d : E \to \mathbb{Q}_+$, a total order $<$ on V and an embedding $x' : [K] \to \mathbb{R}^K$ such that:
(1) x' is a valid partial embedding of $G[[K]]$ (START)
(2) G contains all $(K + 1)$-cliques of $<$-consecutive vertices as induced subgraphs (DISCRETIZATION)
(3) $\forall v \in V$ with $v > K$, $\Delta_{K-1}(\gamma_K(v)) > 0$ (STRICT SIMPLEX INEQUALITIES),
is there a valid embedding x of G in $\mathbb{R}^K$ extending x'?

We denote by X the set of embeddings solving a $^{\mathsf{K}}$DMDGP instance. The $^{\mathsf{K}}$DMDGP generalizes the DISCRETIZABLE MOLECULAR DISTANCE GEOMETRY PROBLEM (DMDGP) [3] from $\mathbb{R}^3$ to $\mathbb{R}^K$. Furthermore, it is a subclass of the DISCRETIZABLE DISTANCE GEOMETRY PROBLEM (DDGP) [3] given by all DDGP instances where the K adjacent predecessors used to determine the two positions for the current vertex are immediate. Since for the DDGP $|X|$ is finite [3], this also holds for the $^{\mathsf{K}}$DMDGP; and since the DMDGP is **NP**-hard [3], the same is true for the $^{\mathsf{K}}$DMDGP. For a partial embedding x of G and $\{u, v\} \in E$ let S_{uv}^x be the sphere centered at x_u with radius d_{uv}. The BP algorithm, used for solving the DDGP and its restrictions, is BP($K + 1, x', \emptyset$) (see Alg. 2). By STRICT SIMPLEX INEQUALITIES, $|P| \leq 2$. At termination, X contains all embeddings extending x' [3,5].

Algorithm 1 BP(v, $\bar{x}$, X)

Require: A vtx. $v \in V \setminus [K]$, a partial emb. $\bar{x} = (x_1, \ldots, x_{v-1})$, a set X.

1: $P = \bigcap_{\substack{u \in N(v) \\ u < v}} S_{uv}^{\bar{x}}$;

2: $\forall p \in P$ ($(x \leftarrow (\bar{x}, p))$; **if** $(\rho(v) = n)$ $X \leftarrow X \cup \{x\}$ **else** BP$(v+1, x, X)$).

3 BP tree geometry

Since the definition of the $^{\mathsf{K}}$DMDGP requires G to have at least those edges used to satisfy the DISCRETIZATION axiom, we partition E into the sets $E_D = \{\{u, v\} \mid |\rho(v) - \rho(u)| \leq K\}$ and $E_P = E \setminus E_D$. With a slight abuse of notation we call E_D the *discretization distances* and E_P the *pruning distances*. Discretization distances guarantee that a DGP instance is in the $^{\mathsf{K}}$DMDGP. Pruning distances are used to reduce the BP search space by pruning its tree. In practice, pruning distances might make the set P in Alg. 2 have cardinality 0 or 1 instead of 2. We assume G is a feasible instance of the $^{\mathsf{K}}$DMDGP.

Let $G_D = (V, E_D, d)$ and X_D be the set of embeddings of G_D; since G_D has no pruning distances, the BP search tree for G_D is a full binary tree and $|X_D| = 2^{n-K}$. The discretization distances arrange the embeddings so that, at level ℓ, there

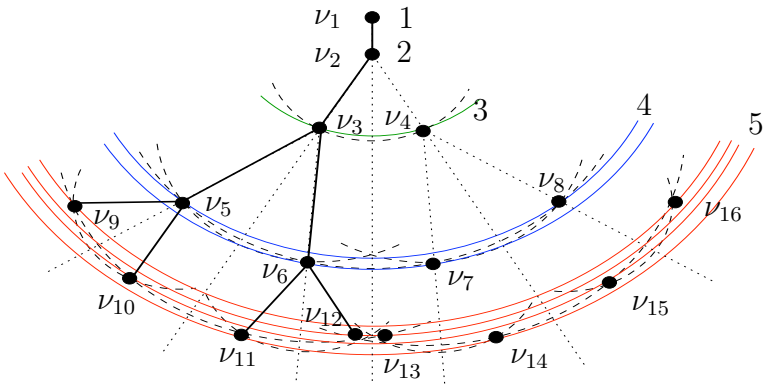


Fig. 1. A pruning distance $\{1, 4\}$ prunes either ν_6, ν_7 or ν_5, ν_8.

are $2^{\ell-K}$ possible embeddings x_v for the vertex v with rank ℓ. Furthermore, when $P = \{x_v, x'_v\}$ and the discretization distances to v only involve the K immediate predecessors of v, we have that $x'_v = R_x^v(x_v)$ [7], the reflection of x_v w.r.t. the hyperplane through $x_{v-K}, \ldots, x_{v-1}$. This also implies that the partial embeddings encoded in two BP subtrees rooted at reflected nodes ν, ν' are reflections of each other.

Theorem 24 ([7]) *With probability 1:* $\forall v > K, u < v-K$ $\exists\, H^{uv} \subseteq \mathbb{R}$ *s.t.* $|H^{uv}| = 2^{v-u-K}$ *and* $\forall x \in X$ $\|x_v - x_u\| \in H^{uv}$*; also* $\forall x \in X$ $\|x_v - x_u\| = \|R_x^{u+K}(x_v) - x_u\|$ *and* $\forall x' \in X$ $(x'_v \notin \{x_v, R_x^{u+K}(x_v)\} \rightarrow \|x_v - x_u\| \neq \|x'_v - x_u\|)$.

Proof (sketch) Sketched in Fig. 1; the circles mark distances to vertex 1. ∎

4 BP search trees with bounded width

Consider the BP tree for G_D and assume that there is a pruning distance $\{u, v\} \in E_P$; at level u there are $\max(2^{u-K}, 1)$ nodes, each of which is the root of a subtree with $2^{v-\max(u,K)}$ nodes at level v. By Thm. 24, for each such subtree only two nodes will encode a valid embedding for v (we call such nodes *valid*). Thus the number of valid nodes at level $v > K$ is $2^{\max(u-K+1,1)}$.

Fig. 2 shows a Directed Acyclic Graph (DAG) $\mathcal{D}_{uv}$ that we use to compute the number of valid nodes in function of pruning distances between two vertices $u, v \in V$ such that $v > K$ and $u < v - K$. The first line shows different values for

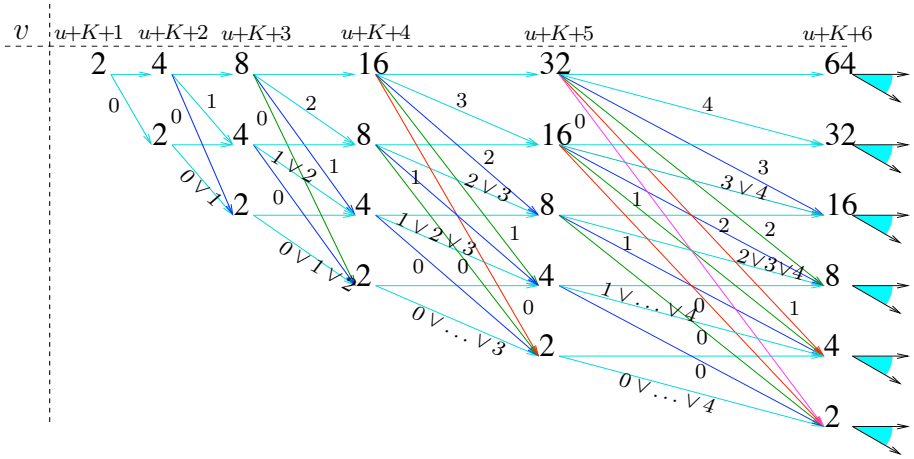


Fig. 2. Number of valid nodes in function of the pruning distances.

the rank of v w.r.t. u; an arc labelled with an integer i implies the existence of a pruning distance $\{u + i, v\}$ (arcs with $\vee$-expressions replace parallel arcs with different labels). An arc is unlabelled if there is no pruning distance $\{w, v\}$ for any $w \in \{u, \ldots, v-K-1\}$. The nodes of the DAG are arranged vertically by BP search tree level. A path p in this DAG represents the set of pruning distances between u and v: p_ℓ is the number of valid nodes in the BP search tree at level ℓ. For example, following unlabelled arcs corresponds to no pruning distance between u and v and leads to a full binary BP search tree with 2^{v-K} nodes at level v.

Each E_P corresponds to a longest path in $\mathcal{D}_{1n}$; BP trees have bounded width when these paths are below a diagonal with constant node labels.

Proposition 1 *If $\exists v_0 \in V \setminus [K]$ s.t. $\forall v > v_0\ \exists! u < v - K$ with $\{u, v\} \in E_P$ then the BP search tree width is bounded by 2^{v_0-K}.*

Proof (sketch) This corresponds to a path $\mathsf{p}_0 = (2, 4, \ldots, 2^{v_0-K}, \ldots, 2^{v_0-K})$ that follows unlabelled arcs up to level v_0 and then arcs labelled $v_0 - K - 1$, $v_0 - K - 1 \vee v_0 - K$, and so on, leading to nodes that are all labelled with 2^{v_0-K} (see Fig. 3, left). ∎

Proposition 2 *If $\exists v_0 \in V \setminus [K]$ such that every subsequence s of consecutive vertices $>v_0$ with no incident pruning distance is preceded by a vertex v_s such that $\exists u_s < v_s\ (\rho(v_s) - \rho(u_s) \geq |s| \wedge \{u_s, v_s\} \in E_P)$, then the BP search tree width is bounded by 2^{v_0-K}.*

Proof (sketch) Such instances yield paths that are below the path p_0 described in the proof of Prop. 1 (Fig. 3, right). ∎

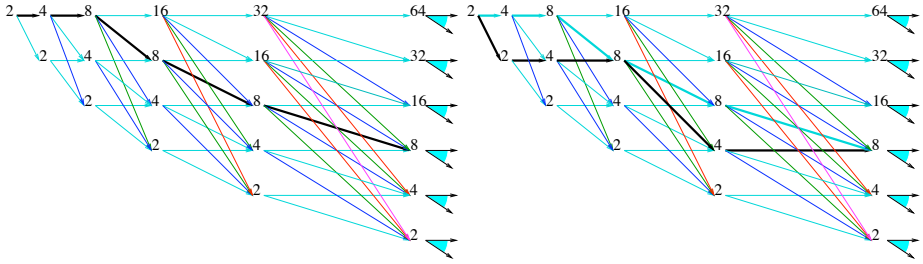


Fig. 3. A path p_0 with treewidth 8 (left) and another path below p_0 (right).

Moreover, For those instances for which the BP search tree width has a $O(\log n)$ bound, the BP has a polynomial worst-case running time $O(L2^{\log n}) = O(Ln)$, where L is the complexity of computing P.

On a set of 16 protein instances from the Protein Data Bank (PDB), twelve satisfy Prop. 4.1, and four Prop. 4.2, all with $v_0 = 4$. This validates the expectation that BP has polynomial complexity on real proteins.

References

[1] L. Blumenthal. *Theory and Applications of Distance Geometry*. Oxford University Press, Oxford, 1953.

[2] T. Eren, D.K. Goldenberg, W. Whiteley, Y.R. Yang, A.S. Morse, B.D.O. Anderson, and P.N. Belhumeur. Rigidity, computation, and randomization in network localization. *IEEE Infocom Proceedings*, pages 2673–2684, 2004.

[3] C. Lavor, L. Liberti, and N. Maculan. The discretizable molecular distance geometry problem. Technical Report q-bio/0608012, arXiv, 2006.

[4] C. Lavor, A. Mucherino, L. Liberti, and N. Maculan. On the computation of protein backbones by using artificial backbones of hydrogens. *Journal of Global Optimization*, accepted.

[5] L. Liberti, C. Lavor, and N. Maculan. A branch-and-prune algorithm for the molecular distance geometry problem. *International Transactions in Operational Research*, 15:1–17, 2008.

[6] L. Liberti, C. Lavor, A. Mucherino, and N. Maculan. Molecular distance geometry methods: from continuous to discrete. *International Transactions in Operational Research*, 18:33–51, 2010.

[7] L. Liberti, B. Masson, C. Lavor, J. Lee, and A. Mucherino. On the number of solutions of the discretizable molecular distance geometry problem. Technical Report 1010.1834v1[cs.DM], arXiv, 2010.

[8] A. Mucherino, C. Lavor, and L. Liberti. The discretizable distance geometry problem. *Optimization Letters*, in revision.

[9] J.B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. *Proceedings of 17th Allerton Conference in Communications, Control and Computing*, pages 480–489, 1979.

Discounted Markov Decision Processes and Algorithms for Solving Stochastic Control Problem on Networks

Dmitrii Lozovanu, [a] Stefan Pickl [b]

[a] *Institute of Mathematics and Computer Science, Academy of Sciences, Academy str., 5, Chisinau, MD–2028, Moldova*
lozovanu@usm.md

[b] *Institut für Theoretische Informatik, Mathematik und Operations Research, Fakultät fur Informatik, Universität der Bundeswehr, München*
stefan.pickl@unibw.de

Abstract

A class of discounted Markov decision processes with stopping states is considered and the problem of determining the optimal policy in such processes is studied. The linear programming approach for determining the optimal solution of the considered problem is proposed and its application for solving the discounted stochastic control problem on networks is described. Furthermore a polynomial time algorithm for determining the optimal stationary strategies of the control problem on networks with fixed starting and final states is developed.

Key words: Discounted Markov Decision Processes , Stochastic Control Problem on Network, Optimal Stationary Strategies, Polynomial Time Algorithm.

1 Introduction and Problem Formulation

In this paper we apply the concept of discounted Markov decision processes to stochastic optimal control problem on networks and develop the linear programming approach for determining the optimal stationary strategies of the problem with fixed starting and final states [1,2]. Based on this approach we develop a polynomial time algorithm for solving the discounted stochastic control problem on network. The statement of this problem is the following.

Let L be a time-discrete system with finite set of states X. At every discrete moment of time $t = 0, 1, 2, \ldots$ the state of the dynamical system is $x(t) \in X$. Two states $x_0, x_f \in X$ are given where $x_0 = x(0)$ is the staring state of L and x_f is the state in which the dynamical system must be brought. The dynamics of system L is

described by a directed graph $G = (X, E)$ where the set of vertices X corresponds to the set of states of system L and a directed edge $e = (x, y) \in E$ signifies the possibility of system L to pass from the state $x = x(t) \in X$ to the state $y = x(t + 1) \in X$ for arbitrary discrete moment of time $t = 0, 1, 2, \ldots$. Thus the set of edges $E(x) = \{e = (x, y) \in E | y \in X\}$ corresponds to the set of feasible controls in the state $x = x(t)$. On edge set E is defined a function $c : E \to R$ which gives to each directed edge $e = (x, y)$ a cost $c_e = c_{x,y}$, $i.e.$ $c_{x,y}$ expresses the cost of system's transition from the state $x = x(t)$ to the state $y = x(t)$ for arbitrary $t = 0, 1, 2, \ldots$. The transitions costs of the dynamical system we consider with given discount factor γ which means that if the system at the moment of time t passes from the state $x = x(t)$ to the state $y = x(t + 1)$ then the the cost $c_{x,y}$ is multiplied by γ^t, i.e. the cost of system's transition from the state $x = x(t)$ at the moment of time t to the state $y = x(t + 1)$ is equal to $\gamma^t c_{x,y}$.

For the control problem on network we have the following behavior of the dynamical system. If $x(0) \in X_1$ then the decision maker select a transition to the state $x(1) = x_1$ and we obtain the state x_1 at the moment of time $t = 1$; in the case $x(0) \in X_2$ the system passes from $x(0)$ to a state $x(1)$ in the random way. If at the moment of time $t = 1$ the state x_f is reached ($x_1 = x_f$), then the dynamical system stops transitions. In general case, if at the moment of time t the system is in the state $x(t) \in X_1$ and $x(t) \neq x_f$ then the decision maker select a transition to the next state $x(t + 1)$; if $x(t) \in X_2$ then the system passes to to a state $x(t + 1)$ in the random way. The dynamical system stops transitions if $x(t) = x_f$. It is evident that the final states x_f in this process can be reached with the probability that depends on probability distribution functions in uncontrollable states as well on control in controllable states. We assume that G possesses the property that the final state x_f can be reached with the probability equal to 1 for an arbitrary control and consider the problem of determining the control with minimal expected total discounted cost. For general case of the problem the expected total discounted cost can be estimated if the final state have been reached. In the considered problem we assume that the decision maker in the control process uses the stationary the stationary strategies of state's transition for the dynamical system. A stationary strategy we define as a map

$$s : x \to y \in X_1(x) \; for \; x \in X_1 \setminus \{x_f\},$$

which uniquely determine the transitions from the states $x = x(t) \in X_1 \setminus \{x_f\}$ to the states $y = s(x) \in X$ for arbitrary discrete moment of time $t = 0, 1, 2, \ldots$. In the terms of stationary strategies the discounted stochastic control problem on network can be formulated in the following way.

Let s be an arbitrary stationary strategy. We define the graph $G_s = (X, E_s \cup E_2)$, where $E_s = \{e = (x, y) \in E \mid x \in X_1, y = s(x)\}$, $E_2 = \{e = (x, y) \mid x \in X_2, y \in X\}$. This graph corresponds to a Markov process with the probability

matrix $P^s = (p^s_{x,y}),$ where

$$p^s_{x,y} = \begin{cases} p_{x,y}, & \text{if } x \in X_2 \text{ and } y = X; \\ 1, & \text{if } x \in X_1 \text{ and } y = s(x); \\ 0, & \text{if } x \in X_1 \text{ and } y \neq s(x). \end{cases}$$

For this Markov process with transition costs $c_e, \quad e \in E$ we define the expected total discounted cost $\sigma^\gamma_{x_0}(s)$ [1,2]. We consider the problem of determining the strategy s^* for which

$$\sigma^\gamma_{x_0}(s^*) = \min_s \sigma^\gamma_{x_0}(s).$$

We show that if x_f is a sink vertex in G and c_e for $e \in E$ are positive then the optimal strategy s^* for the considered problem exists and it can be found using the linear programming approach, i.e the problem can be solved by using a polynomial time algorithm.

2 The Main Results

To study and solve the considered stochastic control problem on network we use the framework of discounted Markov decision process (X, A, c, p) with discount factor γ $(0 < \gamma < 1)$, where X is the set of states of the system, A is the set of actions, $c : X \times X \to R$ is the transition cost function that gives the costs $c_{x,y}$ of system's transitions for arbitrary $x, y \in X$ and $p : X \times X \times A \to [0, 1]$ is the transition probability function that satisfy the condition $\sum_{y \in X} p^a_{x,y} = 1, \forall x \in X, a \in A(x)$ (here $A(x)$ is the set of actions in the state $x \in X$). A stationary strategy (a policy) in this Markov decision process we define as a map $s : X \to A$ that determines for each $x \in X$ an action $a \in A(x)$. We consider a Markov decision processes with given absorbing state x_f,where $c_{x_f,x_f} = 0$, i.e. we assume that the process stops as soon x_f is reached. For such Markov processes we consider the problem of determining the stationary strategy s^* that provides the minimal expected total discounted costs for an arbitrary starting state. We sow that optimal solution of the problem can be found by solving the following linear programming problem:
Minimize

$$\phi(\alpha, \beta) = \sum_{x \in X} \sum_{a \in A(x)} (\sum_{y \in X} c_{x,y} p_{x,y}) \alpha_{x,a} \tag{2.1}$$

subject to

$$\begin{cases} \beta_y - \gamma \sum_{x \in X} \sum_{a \in A(x)} p^a_{x,y} \, \alpha_{x,a} \geq 1, \ \ y \in X; \\ \qquad \sum_{a \in A(x)} \alpha_{x,a} = \beta_x, \ \ \forall x \in X \setminus \{x_f\}; \\ \beta_y \geq 0, \ \ \forall y \in X \setminus \{x_f\}; \quad \alpha_{x,a} \geq 0, \ \ \forall x \in X \setminus \{x_f\}, \, a \in A(x). \end{cases} \tag{2.2}$$

The optimal stationary strategy s^* for the problem with stopping state $x_f \in X$ is determined as follows: for $x \in X \setminus \{x_f\}$ fix $a = s^*(x)$ if $\alpha^*_{x,a} \neq 0$. Using the dual lineal programming model for problem (2.1),(2.2) we show that for determining the optimal stationary strategies in the discounted stochastic control problem can be used the following linear programming problem:
Maximize

$$\varphi(\sigma) = \sum_{x \in X} \sigma_x \tag{2.3}$$

subject to

$$\begin{cases} \sigma_x - \gamma\sigma_y \leq c_{x,y}, \ \forall x \in X_1, \ y \in X(x); \\ \sigma_x - \gamma \sum_{y \in X} p_{x,y}\sigma_y \leq \sum_{y \in X(x)} c_{x,y}p_{x,y}, \quad \forall x \in X_2 \setminus \{x_f\} \end{cases} \tag{2.4}$$

The optimal stationary strategy for the problem on network can be determined by fixing $s^* : X \setminus \{x_f\} \to A$ such that $s^*(x) = y \in X^*(x) \ \forall x \in X_1 \setminus \{x_f\}$, where $X^*(x) = \{y \in X(x) | \ \sigma_x - \gamma\sigma_y = c_{x,y}\}$. Note that the considered model is valid also for $\gamma = 1$. It is easy to observe that the linear programming model (2.3),(2.4) in the case $X_2 = \emptyset, \gamma = 1$ is transformed in the linear programming model for minimum cost problem in a weighted directed graph with fixed sink vertex x_f.

References

[1] Lozovanu, D, Pickl, S., Determining optimal stationary strategies for discounted stochastic optimal control problem on networks. 9th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, Cologne 115-118, (2010)

[2] Puterman, M., Markov Decision Processes:Stochastic Dynamic Programming. John Wiley, New Jersey (2005)

Infeasible path formulations for the time-dependent TSP with time windows

I. Méndez-Díaz, [a] J.J. Miranda-Bront, [a] P. Toth, [b] P. Zabala [a]

[a] *Departamento de Computación, FCEyN, Universidad de Buenos Aires, Av. Cantilo s.n., Pabellón I, Ciudad Universitaria, Buenos Aires, Argentina*
{imendez, jmiranda, pzabala}@dc.uba.ar

[b] *DEIS, University of Bologna, Bologna, Italy*
paolo.toth@unibo.it

Key words: Branch and Cut, Infeasible path, TDTSP, Time Windows

1 Introduction

The *Time-Dependent TSP with Time Windows* (TDTSP-TW) is a generalization of the classical ATSP-TW in which the cost of the travel between two cities depends on the time of the day the arc is travelled.

The problem can be stated as follows: Consider a complete digraph $D = (V, A)$, with $V = \{0, 1, \ldots, n, n+1\}$ the set of vertices and A the set of arcs. Vertices 0 and $n + 1$ represent the depot. There is a discrete time horizon $0, 1, \ldots, T$ in which the vehicle moves along the network. We assume that, for each arc $(i, j) \in A$, the travel time function is discretized into M different time periods, and that it is constant (and integer). Therefore, the problem is formulated on an expanded network where we replace each arc $(i, j) \in A$ by M parallel links going from i to j, one for each possible time period. Time period m for arc $(i, j) \in A$ is bounded by $(T_{ij}^{m-1}, T_{ij}^{m}]$, for $m = 1, \ldots, M$, where $T_{ij}^{0} = 0$. We name c_{ij}^{m} and θ_{ij}^{m} the travel time and cost from i to j in time period m, respectively. For each vertex $i \in V$, p_i represents its processing time and $W_i = [r_i, d_i]$ the corresponding time window, where r_i and d_i are the realease and deadline times, respectively. The TDTSP-TW involves finding a minimum cost tour, starting from vertex 0 and ending at vertex $n + 1$, that visits each vertex within its time window exactly once. It is important to note that waiting times at both the arrival and the departure of each vertex are allowed.

Exact approaches for the TDTSP can be found in [3,4] and the version considering time windows in [1], although the latter does not allow waiting times at departure

[1] Partially supported by an ERASMUS Mundus Grant from the EU.

of a vertex. In this paper, we propose two new formulations for the TDTSP-TW based on the idea of *infeasible paths*, that can be adapted for the case without time windows. In addition, we also present preliminary computational results for a B&C algorithm on instances with up to 60 vertices.

2 Models

We start by generalizing the definition of infeasible paths for the ATSP-TW (see, e.g. [2]) to the time-dependent case. Let $P = (v_1, v_2, \ldots, v_k)$ be a simple path and $T = (m_1, m_2, \ldots, m_{k-1})$ the corresponding sequence of time periods. We define a *time dependent simple path* as a combination between P and T, $[P, T]$, such that arc (v_i, v_{i+1}) is travelled in time period m_i, for $i = 1, \ldots, k-1$. Let t_{v_i} and s_{v_i} be, respectively, the *earliest* departure and arrival time for vertex v_i given $[P, T]$, defined as: (a) $t_{v_1} = T_{v_1 v_2}^{m_1 - 1} + 1$, (b) $s_{v_i} = \max\{t_{v_{i-1}} + c_{v_{i-1} v_i}^{m_{i-1}}, r_{v_i}\},\ i = 2, \ldots, k$ and (c) $t_{v_i} = \max\{s_{v_i} + p_{v_i}, T_{v_i v_{i+1}}^{m_i - 1} + 1\},\ i = 2, \ldots, k-1$.

These formulae allow waiting times, both at the arrival and the departure of a vertex. A Hamiltonian time dependent path $H = (0, v_1, \ldots, v_n)$ is *feasible* if $T_{v_i v_{i+1}}^{m_{i-1}} < t_{v_i} \leq \min\{T_{v_i v_{i+1}}^{m_i}, d_{v_i} + p_{v_i}\}$ and $r_{v_i} \leq s_{v_i} \leq d_{v_i}$ for $v_i \in H$. A time dependent simple path is *infeasible* if it cannot be a subpath in any feasible Hamiltonian path.

We let binary variable x_{ij}^m take value 1 iff we are travelling from vertex i to vertex j in time period $(T_{ij}^{m-1}, T_{ij}^m]$, and $P = (v_1, v_2, \ldots, v_k)$ and $T = (m_1, m_2, \ldots, m_{k-1})$ be an infeasible time dependent simple path. The generalization of well-known *Tournament constraints* (TDTOURs) reads

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} x_{v_i v_j}^{m_i} \leq k - 2. \tag{2.1}$$

Traditional x_{ij} variables from the ATSP can be easily defined in terms of x_{ij}^m. By considering the assignment equations from [3], the SEC over x_{ij} variables, inequalities (2.1) and the objective function minimizing the cost of the tour, we obtain a formulation for the TDTSP-TW, which we name TDIPF.

2.1 A Refined Formulation

Aiming to obtain stronger LP relaxations, we generalize the ideas presented in [2]. Following their notation, for each vertex $i \in V$ its corresponding time window W_i is divided into a set of L_i buckets $B_i = \{b_1^i, \ldots, b_{L_i}^i\}$ in such a way that each bucket $b_l^i = [r_{b_l^i}, d_{b_l^i}]$ satifies the following conditions: (i) $r_{b_1^i} = r_i$, (ii) $d_{b_{L_i}^i} = d_i + p_i$, (iii) $r_{b_{l+1}^i} > d_{b_l^i}$, for $l = 1, \ldots, L_i - 1$ and (iv) If $T_{ij}^m \in W_i$ for $j \in V$ and $1 \leq m \leq M$, then for some $1 \leq l \leq L_i$, $d_{b_l^i} = T_{ij}^m$.

Condition (iv) is new with respect to the original ones. To account for time-dependency, we force time period changes taking place during the vertex's time window to match the end of one of the buckets defined. By definition, given an arc

$(i,j) \in A$, a bucket $b \in B_i$ belongs to exactly one time period. Therefore, we denote by $c_{ij}^b = c_{ij}^m$ the time-dependent travel time when travelling arc (i,j) starting at vertex i during bucket b. Conversely, for $(i,j) \in A$ a time period $1 \leq m \leq M$ is covered by a unique set of buckets $b \in B_i$.

We define, for each vertex i and bucket $b_l \in B_i$, $I_k(i,b_l) = \{b \in B_k \ : d_{b_{l-1}} < r_b + p_k + c_{ki}^b \leq d_{b_l}\}$ as the collection of possible starting buckets at vertex k given that we select arc (k,i) and the starting bucket at vertex i is b_l, with $d_{b_0} = -\infty$. As in [2], we let binary variables x_{ij}^b take value 1 iff the vehicle travels from vertex i to vertex j starting from i in $b \in B_i$, and z_i^b take value 1 iff the vehicle arrives at i in bucket b. To account for waiting times at the departure of a vertex, we introduce binary variables y_i^b which take value 1 iff the vehicle leaves vertex i in bucket b. Variables x_{ij}^m, and therefore x_{ij}, can be defined correctly in terms of x_{ij}^b.

By considering (2.2) - (2.8) together with SEC and inequalities (2.1) for all time dependent infeasible pahts we obtain another model for the TDTSP-TW, that we name TDTBF. The objective function (2.2) minimizes the cost of the tour. Equations (2.3) and (2.4) establish that each vertex should be visited, and (2.5) account for waiting times at the departure. Finally, equations (2.6) and (2.7) relate x_{ij}^b variables with z_i^b and y_i^b, respectively.

$$\min z_{\text{cost}} = \sum_{(i,j)\in A} \sum_{m=1}^{M} \theta_{ij}^m x_{ij}^m \tag{2.2}$$

$$\text{s.t.} \quad \sum_{b\in B_i} z_i^b = 1 \ \ \forall\, i \in V\backslash\{0\} \tag{2.3}$$

$$\sum_{b\in B_i} y_i^b = 1 \ \ \forall\, i \in V\backslash\{n+1\} \tag{2.4}$$

$$\sum_{\beta\geq b,\ \beta\in B_i} y_i^\beta \geq z_i^b \ \ \forall\, i \in V\backslash\{0,n+1\},\ \beta \in B_i \tag{2.5}$$

$$\sum_{j=1,j\neq i}^{n} x_{ij}^b = y_i^b \ \ \forall\, i \in V\backslash\{n+1\},\ \forall\, b \in B_i \tag{2.6}$$

$$\sum_{k=1,k\neq i}^{n} \sum_{\beta\in I_k(i,b)} x_{ki}^\beta = z_i^b \ \ \forall\, i \in V\backslash\{0\},\ \forall\, b \in B_i \tag{2.7}$$

$$x_{ij}^b, z_i^b, y_i^b \in \{0,1\} \ \ \forall\,(i,j) \in A,\ \forall\, i \in V,\ \forall\, b \in B_i \tag{2.8}$$

3 Preliminary computational results

We conducted experiments considering the input files provided in [1] for 1 minute discretization and values of $n = 10, 20, \ldots, 60$. The maximum values for the size of the time windows are $20, 40, 60$, with five instances for each value (named pnwX, with n the number of vertices). We consider also instances with time windows of maximum size 80 and 120, obtained by doubling the size of the ones with 40 and

60, respectively (named pnwXA). We implemented the modified version of the formulation in [3] proposed in [4] (MD), the final formulation from [4] (SCM) as well as formulations TDIPF and TDTBF.

Algorithms are coded using CPLEX 12 and tests were run on a laptop with an Intel i3-350 CPU and 4 Gb of RAM. For MD and SCM, we consider the default CPLEX 12 algorithm. For TDIPF and TDTBF, we develop for each model a B&C algorithm including the SEC and TDTOURs as cuts. All CPLEX general purpose features are disabled. The table below shows the average computational results for the % GAP of the lower bounds at the root node corresponding to the LP relaxations for MD and SCM and to the cutting plane procedures for TDIPF and TDTBF, as well as the running time (expressed in seconds) of the overall exact algorithm for all methods. A cell filled with $(***)$ means that the algorithm cannot solve any of the corresponding instances to optimality within 1800 seconds. A number between parenthesis represents the number of instances that were solved to optimality. Due to space limitations, we report the results only for instances with $n \geq 30$. For MD and TDTBF, the average times are calculated over instances solved by both algorithms.

Inst.	n	MD		SCM		TDIPF		TDTBF	
		%GAP	Time	%GAP	Time	%GAP	Time	%GAP	Time
p30wX	30	13.46	0.06	13.46	1.55	0.94	0.03	0.36	0.02
p40wX	40	17.00	2.44	17.00	87.01 (14)	3.92	104.15	2.80	28.88
p50wX	50	15.89	3.07	15.89	160.28 (13)	4.31	26.08 (14)	2.92	4.29
p60wX	60	15.67	11.96	15.67	186.23 (8)	4.82	21.22 (14)	3.47	10.34
p30wXA	30	24.30	2.41	24.30	212.08	6.79	3.14	5.20	1.06
p40wXA	40	23.36	43.55	23.36	1044.50 (2)	8.78	96.77 (9)	6.11	6.80
p50wXA	50	27.46	354.01 (7)	27.46	***	11.81	849.32 (4)	8.49	98.70 (9)
p60wXA	60	25.98	415.62 (5)	25.98	***	12.36	441.22 (2)	8.91	83.28 (5)

The TDTBF appears as the best approach, solving more instances than the other methods with less computational effort. The only exception is for instances in p40wX, but the difference is produced by only one of the 15 instances in the group. The corresponding lower bounds are tight, obtaining also the best results. TDIPF lower bounds are also reasonable, but not as good as the previous ones. This is due to the quality of the initial LP relaxations. Model MD produces reasonable results, but tend to get worse as the time windows are wider. SCM shows that when minimizing the cost of the tour (instead of the makespan, as in [4]) both running times and quality of the LP relaxations are not good.

As future research we will continue working on deriving new families of valid inequalities as well as other aspects related with the B&C algorithm.

References

[1] J. Albiach, J. M. Sanchis, and D. Soler. An asymmetric tsp with time windows and with time-dependent travel times and costs: An exact solution through a graph transformation. *Eur. J. Oper. Res.*, 189(3):789–802, 2008.

[2] S. Dash, O. Günlük, A. Lodi, and A. Tramontani. A time bucket formulation for the tsp with time windows. *Forthcoming in INFORMS J. Comput.*, 2010.

[3] C. Malandraki and M. S. Daskin. Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Sci.*, 26(3):185–200, 1992.

[4] G. Stecco, J.-F. Cordeau, and E. Moretti. A branch-and-cut algorithm for a production scheduling problem with sequence-dependent and time-dependent setup times. *Comput. Oper. Res.*, 35(8):2635–2655, 2008.

A hereditary view on efficient domination

Martin Milanic

University of Primorska
UP FAMNIT, Glagoljaska 8, 6000 Koper, Slovenia
UP PINT, Muzejski trg 2, 6000 Koper, Slovenia
martin.milanic@upr.si

Key words: perfect code, efficient domination, perfect domination, hereditary efficiently dominatable graph, forbidden induced subgraph characterization, polynomial time algorithm

1 Introduction

The concept of an efficient dominating set in a graph (also known as perfect code, 1-perfect code, perfect dominating set, or perfect independent dominating set) was introduced by Biggs [2] as a generalization of the notion of a perfect error-correcting code in coding theory. Given a (simple, finite, undirected) graph $G = (V, E)$, we say that a vertex *dominates* itself and each of its neighbors. An *efficient dominating set* in G is a subset of vertices $S \subseteq V$ such that every vertex $v \in V$ is dominated by precisely one vertex from S. Efficient domination has several interesting applications in coding theory and resource allocation of parallel processing systems.

We say that a graph is *efficiently dominatable* (ED) if it contains an efficient dominating set. The problem of recognizing ED graphs is NP-complete even for restricted graph classes such as planar cubic graphs, planar bipartite graphs, chordal bipartite graphs and chordal graphs, but solvable in polynomial time for trees, interval graphs, series-parallel graphs, split graphs, block graphs, circular-arc graphs, permutation graphs, trapezoid graphs, cocomparability graphs, bipartite permutation graphs and distance-hereditary graphs.

Some small graphs such as the bull, the fork, or the 4-cycle C_4 are not efficiently dominatable. (The *bull* is the graph with vertex set $\{a, b, c, d, e\}$ and edge set $\{ab, bc, bd, cd, de\}$, and the *fork* is the graph obtained from the the *claw* $K_{1,3}$ by subdividing one of its edges.) All paths are efficiently dominatable, and a cycle C_k on k vertices is efficiently dominatable if and only if k is a multiple of 3.

**This work was supported in part by "Agencija za raziskovalno dejavnost Republike Slovenije", research program P1-0285.

The difficulty of characterizing ED graphs is perhaps related to the fact that they do not form a hereditary graph class (that is, a class of graphs closed under vertex deletions): adding a dominating vetrtex to *any* graph results in an ED graph. In this paper we investigate *hereditary efficiently dominatable* graphs, that is, graphs every induced subgraph of which is efficiently dominatable.

For a set of graphs $\mathcal{F} = \{F_1, \ldots, F_k\}$, we say that a graph is $(F_1, \ldots, F_k)$-free if no induced subgraph of it is isomorphic to a member of $\mathcal{F}$. We will also say that a graph G is (bull, fork, C_{3k+1}, C_{3k+2})-free if no induced subgraph of G is isomorphic to a graph from the set $\{\text{bull, fork}\} \cup \bigcup_{k\geq 1}\{C_{3k+1}, C_{3k+2}\}$. Clearly, the set of hereditary efficiently dominatable graphs forms a hereditary class contained in the class of (bull, fork, C_{3k+1}, C_{3k+2})-free graphs. We show in the present paper that the converse inclusion holds as well. This result is obtained as a consequence of a decomposition theorem for (bull, fork, C_4)-free graphs. We also outline a polynomial time algorithm for a problem generalizing the recognition of ED graphs in a class of graphs properly extending the class of (bull, fork, C_4)-free graphs, by reducing the problem to the maximum weight independent set problem in claw-free graphs.

2 Preliminary definitions

The structure theorem for (bull, fork, C_4)-free graphs (Theorem 1.7 in Section 3) will rely on certain operations that build larger graphs from smaller ones. We introduce the necessary definitions in this section.

Duplicating a vertex v in a graph G means adding to G a new vertex and making it adjacent to v and to all neighbors of v.

For $n \geq 1$, a *raft* R_n is a graph consisting of two disjoint cliques on $n+1$ vertices each, say $X = \{x_0, x_1, ..., x_n\}$ and $Y = \{y_0, y_1, ..., y_n\}$ together with additional edges between $\{x_1, ..., x_n\}$ and $\{y_1, ..., y_n\}$ such that for every $i = 1, ..., n$, vertex x_i is adjacent precisely to $y_i, y_{i+1}, ..., y_n$. We say that X and Y are the *parts* of the raft, and x_0 and y_0 are the *tips* of the raft. See Fig. 1 for examples of rafts.

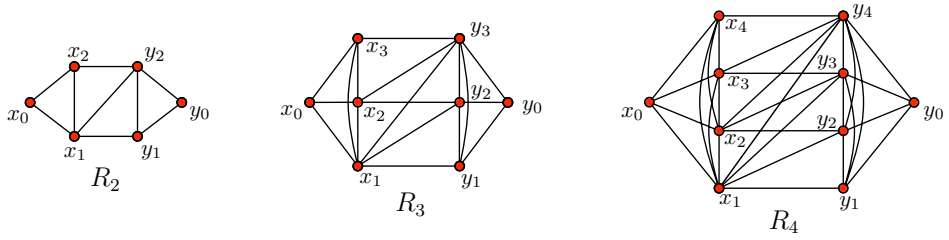


Fig. 1. Rafts R_2, R_3 and R_4

For $n \geq 1$, a *semi-raft* S_n is a graph obtained from a raft R_n (as above) by deleting its tips. We say that $X \setminus \{x_0\}$ and $Y \setminus \{y_0\}$ are the *parts* of the semi-raft. See Fig. 2 for examples of semi-rafts.

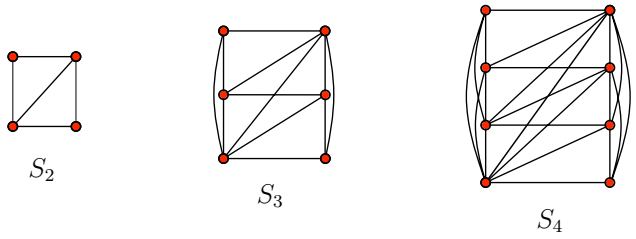


Fig. 2. Semi-rafts S_2, S_3 and S_4

For a graph G and a set $X \subseteq V(G)$, we denote by $G[X]$ the subgraph of G induced by X. Given a graph G and a pair x, y of non-adjacent vertices in G, a *raft expansion* (with respect to x, y) is the operation that replaces G with the graph G' where:

- $V(G') = (V(G) \setminus \{x, y\}) \cup (X \cup Y)$ such that
- $G'[X \cup Y]$ is a raft with parts X and Y and
- $E(G') = \{uv \in E(G) : \{u, v\} \cap \{x, y\} = \emptyset\} \cup \{ux' : x' \in X,\ ux \in E(G)\} \cup \{uy' : y' \in Y,\ uy \in E(G)\} \cup E(G'[X \cup Y])$, where $E(G'[X \cup Y])$ is the edge set of $G'[X \cup Y]$.

Given a graph G and a pair x, y of adjacent vertices in G, a *semi-raft expansion* (with respect to x, y) is the operation that replaces G with the graph G' where:

- $V(G') = (V(G) \setminus \{x, y\}) \cup (X \cup Y)$ such that
- $G'[X \cup Y]$ is a semi-raft with parts X and Y and
- $E(G') = \{uv \in E(G) : \{u, v\} \cap \{x, y\} = \emptyset\} \cup \{ux' : x' \in X,\ ux \in E(G)\} \cup \{uy' : y' \in Y,\ uy \in E(G)\} \cup E(G'[X \cup Y])$.

3 Results

We give the following decomposition theorem for (bull, fork, C_4)-free graphs:

Theorem 25 *Let G be a (bull, fork, C_4)-free graph. Then, G can be built from paths and cycles of order at least 5 by applying a sequence of the following operations:*

- *disjoint union of two graphs,*
- *vertex duplication,*
- *addition of a dominating vertex,*
- *raft expansion,*
- *semi-raft expansion.*

It can be shown that the set of ED graphs is closed under each of the above operations. This implies the following characterization of hereditary efficiently dominatable graphs in terms of forbidden induced subgraphs.

Theorem 26 *The class of hereditary efficiently dominatable graphs equals the class of (bull, fork, C_{3k+1}, C_{3k+2})-free graphs.*

We now turn to some algorithmic questions. Bange et al. [1] introduced the *efficient domination number* of a graph, denoted $F(G)$, as the maximum number of vertices

that can be dominated by a set S that dominates each vertex at most once. Clearly, a graph $G = (V, E)$ is efficiently dominatable if and only if $F(G) = |V|$. The *efficient domination problem* is the problem of computing in a given graph G a set S that dominates $F(G)$ vertices and dominates each vertex at most once. It is not hard to see that the efficient domination problem in a graph G is a special case of the maximum weight independent set problem in the square of G, which is the graph $G^2 = (V, E^2)$ such that $uv \in E^2$ if and only if either $uv \in E$ or u and v have a common neighbor in G. More precisely, denoting by $\alpha_\omega(G)$ the maximum weight of an independent set in G with respect to vertex weights ω, we have the following result:

Proposition 1 *Let $G = (V, E)$ be a graph. Define $\omega(x) := |N[x]|$ for all $x \in V$. Then, a set $S \subseteq V$ dominates $F(G)$ vertices, each vertex at most once, if and only if S is an independent set of maximum ω-weight in G^2. In particular, $F(G) = \alpha_\omega(G^2)$ and G is efficiently dominatable if and only if $\alpha_\omega(G^2) = |V|$.*

The *net* is the graph obtained from a triangle by adding a pendant edge to each vertex, and we denote by E the graph obtained from the claw by subdividing two of its edges. It can be verified that the square of every (E,net)-free graph is claw-free. Based on Proposition 1 and the fact that the maximum weight independent set problem is solvable in polynomial time for claw-free graphs (see, e.g., [3]), this implies the following result.

Theorem 27 *The efficient domination problem can be solved in polynomial time for (E,net)-free graphs.*

It follows from Theorem 26 that every hereditary efficiently dominatable graph is (E, net)-free. Hence:

Corollary 1 *An efficient dominating set in a hereditary efficiently dominatable graph can be found in polynomial time.*

References

[1] D. W. Bange, A. E. Barkauskas and P. J. Slater, Efficient dominating sets in graphs. In: R. D. Ringeisen and F. S. Roberts (Eds.), Applications of Discrete Mathematics, pages 189–199, SIAM, Philadelphia, PA, 1988.

[2] N. Biggs, Perfect codes in graphs, J. Combin. Theory, Ser. B, 15 (1973) 288–296.

[3] Y. Faenza, G. Oriolo and G. Stauffer, An algorithmic decomposition of claw-free graphs leading to an $O(n^3)$-algorithm for the weighted stable set problem. In: SODA '11, pages 630–646.

On the Robust Knapsack Problem

Michele Monaci, [a] Ulrich Pferschy [b]

[a] *DEI, University of Padova, Via Gradenigo 6/A, I-35131 Padova, Italy.*
monaci@dei.unipd.it

[b] *Department of Statistics and Operations Research, University of Graz, Universitaetsstr. 15, A-8010 Graz, Austria.*
pferschy@uni-graz.at

Key words: knapsack problem, robust optimization, approximation ratio

1 Introduction

A standard assumption when solving an optimization problem is that all input data is known in advance. Unfortunately, this is not always the case when real problems are considered, since the models we use only provide an approximation of real systems and because uncertainty can change the effective values of some parameters. Two main methods have been proposed so far to deal with uncertainty. Stochastic optimization is used when a finite number of scenarios is possible and some information about the probabilistic distribution of data is available. On the contrary, robust optimization is based on the definition of a *robust counterpart* for a nominal problem, in which hard constraints are imposed to forbid extreme solutions that are likely to become infeasible.

A widely discussed definition of robustness was provided by Ben-Tal and Nemirovski [2,3], who considered uncertainty to be represented by a certain ellipsoidal set, so as to control the level of conservatism of the solution. The resulting models, despite tractable, turn out to be hard to solve in practice. A more recent progress in robust optimization is the work by Bertsimas and Sim [4], who give a definition of robustness which can be enforced by solving a robust problem that has the same computational complexity as the nominal problem, and that turns out to be efficiently solvable in practice, provided the nominal problem is as well.

In this paper, we consider robustness for the classical 0-1 knapsack problem (KP) (cf. Kellerer et al. [5]) which can be formulated as follows:

$$(KP) \quad \max \sum_{j \in N} p_j \, x_j \tag{1.1}$$

$$\sum_{j \in N} w_j \, x_j \leq c \tag{1.2}$$

$$x_j \in \{0,1\} \qquad j \in N \tag{1.3}$$

where $x_j = 1$ iff item j is inserted into the knapsack. For convenience we assume real weight values and $c = 1$.

Following [4] we study situations of uncertainty where the actual value of each weight coefficient w_j ($j \in N$) is not known in advance, but belongs to the interval $[w_j - \overline{w}_j, w_j + \overline{w}_j]$, and at most Γ coefficients can receive arbitrary weight values within this interval – the remaining item weights remain at w_j. The integer parameter Γ is used to control the trade-off between the required quality of the solution and its robustness with respect to data uncertainty. Indeed, this definition of uncertainty reflects the fact that it is very unlikely that all weight coefficients change simultaneously, and that whenever more that Γ coefficients change, they tend to balance each other. Formally, this means that (1.2) is replaced by

$$\sum_{j \in N} w_j \, x_j + \max_{S \subseteq N : |S| = \Gamma} \sum_{j \in S} \overline{w}_j \, x_j \leq 1. \tag{1.4}$$

In this contribution we will always assume that there is not uncertainty in the profit coefficients and the capacity. Finally, we make two further assumptions on the nature of uncertain weights. First, we assume that each item can be inserted in the knapsack in each robust solution, i.e., $w_j + \overline{w}_j \leq 1 \quad \forall j \in N$. Clearly, any item violating this assumption could be removed from consideration. Then, we assume that, for each item j, the weight uncertainty be at most a constant fraction of the nominal weight of item j, i.e., $\overline{w}_j = \delta w_j$ ($j \in N$). This second requirement is an essential constraint and follows exactly the setup introduced in [4, Sec. 6.1]. From these two assumptions one gets that, for a given instance, the uncertain set is uniquely defined by parameters δ and Γ. In addition, we get that

$$w_j \leq \frac{1}{1+\delta} \quad \forall j \in N. \tag{1.5}$$

The resulting problem has been defined as *Robust Knapsack Problem* (RKP) in [4], where computational experiments about the effect of Γ on the optimal solution value are given. Klopfenstein and Nace [6,7] showed that, under certain assumptions on the way the coefficients change, feasible solutions to (RKP) are also feasible for the *Chance-constrained Knapsack Problem*, i.e., the knapsack problem in which one is required to satisfy constraint (1.2) with a given probability $1 - \varepsilon$. In [6] some special cases in which the optimal solution of (RKP) is optimal for (KP) as well are considered, and the dynamic programming algorithm for (KP) is extended to (RKP). Aissi et al. [1] considered a more restrictive notion of robustness where a certain number of different data scenarios are given and the solution has to be determined before we get to know which scenario actually occurs.

2 Worst Case Ratio Between KP and RKP

In the following we will denote, for a given instance I, the integer optimal solution value of (KP) as $z^*(I)$. In addition, given δ and Γ, we denote by $z^R_{\delta,\Gamma}(I)$ the integer optimal solution value of (RKP). The ratio $r_{\delta,\Gamma}(I) = z^R_{\delta,\Gamma}(I)/z^*(I)$ provides

an indication of the relative worsening of the solution value for instance I due to uncertainty defined by δ, Γ. We are interesting in computing $r(I)$ in the worst case for a fixed uncertainty, i.e.

$$R_{\delta,\Gamma} = \min_I r_{\delta,\Gamma}(I) \tag{2.1}$$

Note that if assumption (1.5) is violated, one easily gets $R_{\delta,\Gamma}$ arbitrarily close to zero (for any $\delta > 0$ and $\Gamma > 0$), by considering an instance with an item $w_1 = 1$ and $p_1 = M$, while all other items have profit ε. The following theorem provides the exact value for $R_{\delta,\Gamma=1}$ for all values of δ.

Theorem 1

$$R_{\delta,1} = \frac{1}{1 + \lceil \delta \rceil} \qquad \text{for all } \delta > 0.$$

Note that for $\delta \leq 1$ this yields $R_{\delta,1} = \frac{1}{2}$. For general $\Gamma \geq 2$ and $\delta > \frac{1}{2}$, we get the following ratio.

Theorem 2

$$R_{\delta,\Gamma} = \frac{1}{1 + \lceil 2\delta \rceil} \qquad \text{for all } \delta > \frac{1}{2},\ \Gamma \geq 2.$$

For $\Gamma \geq 2$ and $\delta \leq \frac{1}{2}$ the following stronger result similar to Theorem 1 holds.

Theorem 3

$$R_{\delta,\Gamma} = \frac{1}{2} \qquad \text{for all } \delta \leq \frac{1}{2},\ \Gamma \geq 2.$$

3 LP Relaxation of RKP

As always for integer optimization problems we are also interested in the behavior of the LP relaxation. This means that we allow items to be partially packed into the knapsack, i.e. $x_j \in [0, 1]$. First we can show the worst case ratio between the robust and the best standard solution value of the LP-relaxations. In analogy to the definition of (2.1) we get:

Theorem 4

$$R^{LP}_{\delta,\Gamma} = \frac{1}{1 + \delta} \quad \textit{for all } \delta > 0,\ \Gamma \geq 1.$$

Considering the worst-case performance ratio of the LP-relaxation with respect to the optimal solution value of (RKP) we define

$$WR_{\delta,\Gamma} = \inf_I \left\{ \frac{z^R_{\delta,\Gamma}(I)}{\overline{z}^R_{\delta,\Gamma}(I)} \right\}, \tag{3.1}$$

where $\overline{z}^R_{\delta,\Gamma}(I)$ denotes the solution value of the LP-relaxation for instance I. Even for $\Gamma = 1$ no approximation can be guaranteed, although a constant bound can be derived for $\delta \leq 1$.

Theorem 5

$$WR_{\delta,\Gamma} = \frac{1}{n} \qquad \text{for some } \delta \geq 0 \text{ and all } \Gamma \geq 1.$$

Theorem 6

$$WR_{\delta,1} = \frac{1}{3} \qquad \text{for any } \delta \leq 1.$$

Finally, we consider combinatorial algorithms to solve the LP-relaxation of (RKP). For the special case $\Gamma = 1$, it is obvious that the available capacity of a robust solution depends on the weight $w_{\max}$ of the largest weight of an item included in the solution and is given by $c - \delta w_{\max}$.

It can be shown that the function that gives the best solution value of a solution with largest weight $w_{\max}$ is a piece-wise linear concave function with at most $2n$ linear pieces. Moreover, we can move from one linear piece to the next in logarithmic time by using an appropriate data structure. Thus computing the optimal solution of the LP-relaxation for $\Gamma = 1$ can be done in $O(n \log n)$ time.

For arbitrary Γ, we define a profit function$P(w_{\min})$ that depends on the smallest weight $w_{\min}$ among the largest Γ item weights in the best LP-solution. It can be shown that also $P(w_{\min})$ is piece-wise linear with $O(n)$ linear pieces, but not necessarily concave. Each of the linear pieces of $P(w_{\min})$ can be determined in linear time by a fairly complicated iterative procedure exploiting a theoretical result which describes the necessary structure of an optimal LP-solution for any given value of $w_{\min}$. This yields a total running time of $O(n^2)$.

References

[1] H. Aissi, C. Bazgan, D. Vanderpooten, Min-max and min-max regret versions of combinatorial optimization problems: A survey, European Journal of Operational Research 197, 427–438, (2009).

[2] A. Ben-Tal, A. Nemirovski, Robust convex optimization, Operations Research 23, 769–805, (1998).

[3] A. Ben-Tal, A. Nemirovski, Robust solutions of linear programming problems contaminated with uncertain data, Math. Progr. 88, 411–424, (2000).

[4] D. Bertsimas, M. Sim, The price of robustness, Operations Research 52, 35–53, (2004).

[5] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer, (2004).

[6] O. Klopfenstein, D. Nace, A robust approach to the chance-constrained knapsack problem, Operations Research Letters 36, 628–632, (2008).

[7] O. Klopfenstein, D. Nace, Valid inequalities for a robust knapsack polyhedron - Application to the robust bandwidth packing problem, Networks, (2011).

Sparsifying Distance Matrices for Protein-Protein Structure Alignments

Antonio Mucherino, [a] Inken Wohlers, [b] Gunnar W. Klau, [b]
Rumen Andonov [c]

[a] *CERFACS, Toulouse, France*
mucherino@cerfacs.fr

[b] *CWI, Life Sciences Group, Amsterdam, The Netherlands*
IRISA, Rennes, France
{i.wohlers,g.w.klau}@cwi.nl

[c] *IRISA, Rennes, France*
randonov@irisa.fr

Abstract

The problem of finding similarities between native protein conformations can be formulated as the problem of aligning inter-residue distance matrices. Recently proposed exact algorithms are able to solve this problem for large proteins only if the considered distance matrices are sparse. We propose a strategy for sparsifying distance matrices in which we keep the distances needed for uniquely reconstructing the conformations of the proteins.

Key words: protein structural alignment, sparse distance matrices, distance geometry

1 Introduction

We consider the problem of aligning native protein conformations [4]. Given two proteins A and B, the problem consists in finding structural similarities between the three-dimensional (3d) structure of A and the 3d structure of B. One way to approach to this problem is to consider inter-residue distance matrices (that can be computed from the known 3d coordinates of residues of A and B) and to identify distances which are similar, in some sense, in the two matrices. In this context, *aligning* inter-residue distance matrices is the process of discovering similarities between pairs of elements contained in the two matrices. This alignment problem is NP-hard [1]: many heuristic algorithms have been proposed for its solution, as well as some exact algorithms.

Structural alignments have important applications in biology [4]. They help in studying the evolutionary relationships among proteins and in investigating their

biological function. The hypothesis is that a protein's function is more conserved during evolution than its sequence. As function is in most cases dictated by structure, protein structure is more conserved than sequence. While in general proteins with similar sequence fold in similar ways, there are thus also proteins with different sequence that share the same structure. In such a case two proteins may share a common ancestor, but their sequences changed over time such that an evolutionary relationship cannot be detected by sequence comparison alone. Furthermore, structural similarities hint to possible functional similarities. It is evident then how structural alignment helps gaining the knowledge that is needed to solve important biological problems, such as unraveling molecular pathways and drug design.

We propose a strategy for sparsifying the protein inter-residue distance matrices which is based on a discrete reformulation of the distance geometry problem. In Section 2, we give more details about the alignment problem. Section 3 briefly introduces the discrete reformulation of the distance geometry problem and describes the strategy for sparsifying the inter-residue distance matrices. Initial computational experiments and conclusions are given in Section 4.

2 Aligning distance matrices

Recently, a general mathematical model for the alignment of inter-residue distance matrices has been proposed in [5]. Special cases of this model consider only sparse inter-residue distance matrices. Aligning proteins based on contact map overlap (CMO) is such a special case. In CMO, the matrices contain only small inter-residue distances which may denote a chemical interaction, *i.e.* a contact, between two residues. Exact algorithms like A_PURVA [6] perform very well for such sparse distance matrices and can oftentimes compute *optimal* solutions in running times comparable to or less than those of recent heuristics.

In the general case, the two compared inter-residue distance matrices contain non-zero elements only (except for the diagonal elements). Considering all available information can help, in theory, the alignment algorithm in performing its task. Nonetheless, exact alignment algorithms can in most cases not manage to compare all this data. One way to overcome this issue is to select only the distances that are useful for finding the biologically correct alignment.

3 Sparsifying distance matrices

In this work, we propose a strategy for sparsifying the inter-residue distance matrices. For each matrix, a subset of distances that is sufficient for a unique reconstruction of the protein's 3d conformation is selected. The problem of determining a protein conformation from an $n \times n$ distance matrix $D = (d_{ij})$ is a distance geometry problem (DGP) [2]. In particular, we consider a discrete reformulation of the distance geometry problem which allows us to construct high precision models of the original protein conformation and also to verify their uniqueness. We refer to this problem to as the discretizable DGP (DDGP) [3].

DDGP matrices $D = (d_{ij})$ must satisfy the following assumption:

$\forall j = \{2, \ldots, n\} \quad \exists i_1, i_2, i_3 \ :$

- $i_1 < j, i_2 < j, i_3 < j \quad (i_1 = i_2$ if $j \leq 2; \quad i_2 = i_3$ if $j \leq 3)$;
- $d_{i_1,j} \neq 0, d_{i_2,j} \neq 0, d_{i_3,j} \neq 0 \quad$ (no distances can be equal to 0);
- $d_{i_1,i_3} < d_{i_1,i_2} + d_{i_2,i_3}$.

Note that we implicitly suppose that there is a total ordering on the atoms of the considered molecule. This assumption allows to discretize the problem and to solve it by using an efficient Branch & Prune (BP) algorithm. This algorithm, differently from other algorithms based on heuristics and/or continuous formulations of the problem, is able to identify *all* possible solutions to the DDGP. Moreover, if the assumption for the discretization is strengthened, so that not only 3 distances but rather 4 distances are required, then there is a trilateration order on the atoms of the molecule. These instances can be solved in polynomial time (even if, in the worst case, the BP algorithm is exponential), and there are only two symmetric solutions. Since one of these two symmetric solutions can be discarded by biological considerations, DDGP distance matrices contain information enough to reconstruct uniquely the original protein conformation. They should therefore be able to describe well the corresponding protein structure.

In general, there exists more than one distance matrix which has this property. For this reason, we select the distance matrix in which the distances have the smallest values, because they are most likely able to capture interactions between residues, like hydrogen-bonding, and thus contain more information regarding the structure.

4 Results and conclusions

For each structure we obtain the full inter-residue distance matrix by computing the relative distances between any pair of C_α carbon atoms of each residue. Then, we sparsify the distance matrix by choosing the distances $d_{i_1,j}$, $d_{i_2,j}$, $d_{i_3,j}$ and $d_{i_4,j}$ so that they are the smallest distances for which the assumption for the discretization is satisfied, and the corresponding DDGP has two symmetric solutions only. We remark that distances between neighbouring residues are avoided (when possible) because they cannot be useful during the alignment process. Finally, we compare the results obtained by aligning these matrices with the results obtained for CMO matrices.

First computational results were obtained for the 98 protein pairs from the Sisy data set of manually curated, challenging structural alignments [7]. We use A_PURVA [6] to align the proposed sparse distance matrices. A_PURVA determines the structural alignment with the maximum number of aligned distances, *i.e.* with the highest number of overlapping non-zero matrix elements. For the considered data set, the average number of non-zero elements in CMO matrices is 932 and in DDGP matrices 1060 (A_PURVA considers no hetero atoms). The quality measure here is the alignment accuracy. 100% alignment accuracy means 100% overlap with the biologically correct reference alignment. The seven heuristic pairwise methods

analyzed in [7] reach between 56 and 83% average alignment accuracy (median $65 - 95\%$). The matrices obtained by DDGP reach on average 56% (median 67%) alignment accuracy and the CMO matrices 70% (median 86%). Therefore, DDGP matrices perform, in general, worse than CMO matrices (average -14%, median -19%).

This is due to the fact that pairs of distances that are important for detecting the similarity are missing. Nevertheless, the above preliminary results for sparsifying the distance matrices are interesting, since they have been obtained by simply choosing the smallest distances for which the matrices satisfy the DDGP assumptions and ensure the uniqueness of the conformation. We are therefore confident that a smarter search among the possible DDGP matrices could help in improving their performances. We are currently investigating the possibility of sparsifying the two matrices simultaneously so that, in average, distances that are considered in one matrix are also present in the second one. We furthermore plan to study how to identify during the execution of the alignment program the best distances to be considered. This may require the solution of several DDGPs for solving one alignment problem.

References

[1] R.H. Lathrop, *The Protein Threading Problem with Sequence Amino Acid Interaction Preferences is NP-complete*, Protein Engineering **7**(9), 1059–1068, 1994.

[2] L. Liberti, C. Lavor, A. Mucherino, N. Maculan, *Molecular Distance Geometry Methods: from Continuous to Discrete*, International Transactions in Operational Research **18**(1), 33–51, 2011.

[3] A. Mucherino, C. Lavor, L. Liberti, *The Discretizable Distance Geometry Problem*, in revision.

[4] M.L. Sierk, G.J. Kleywegt, *Déjà Vu All Over Again: Finding and Analyzing Protein Structure Similarities*, Structure **12**, 2103–2111, 2004.

[5] I. Wohlers, R. Andonov, G.W. Klau, *Algorithm Engineering for Optimal Alignment of Protein Structure Distance Matrices*, Optimization Letters, ISSN:1862–4472, DOI:10.1007/s11590-011-0313-3, 1–13, 2011.

[6] R. Andonov, N. Malod-Dognin, N. Yanev, *Maximum Contact Map Overlap Revisited*, Journal of Computational Biology **18**(1): 27–41, 2011.

[7] C. Berbalk, C.S. Schwaiger, P. Lackner, *Accuracy Analysis of Multiple Structure Alignments*, Protein Science **18**(10): 2027–2062, 2009.

Minimally 2-connected graphs and colouring problems

N. Narayanan

Dept. Mathematics, National Taiwan University, Taipei, Taiwan.

Key words: Minimally 2-connected graphs, colouring, Structural graph theory.

1 Extended Abstract

A 2-connected graph G is minimally 2-connected (*m2c*) , if $G - e$ contains a cut vertex for any edge e. Thus every edge is critical for the 2-connectivity. This class has been studied independently by Plummer [10] and Dirac [6] and they obtain some interesting structural properties of these classes. In the context of edge colouring problems, we revisit these results and this reveals some interesting structural properties which are useful in several graph colouring problems. We present the structural properties and application to 4 different problems viz. induced matching, acyclic and k-intersection edge colourings and balanced decomposition. The structure has similarity with structure of degenerate graphs and this enables one to also generalise some of these results to graphs of given degeneracy. All these results are co-authored by the author in [9,3,4,5]. In the following, $d(x)$ is the degree of a vertex, Δ maximum degree. Vertices of degree at least, at most and equal to k are denoted k^+, k^-, k. For each connected graph we can find a block-cut-vertex tree which contains all cut-vertices and blocks of G and a cut-vertex v of G is adjacent to a block B of G whenever $v \in V(B)$. An *end block* is a block containing at most one cut-vertex in G. Undefined terms can be found in standard textbooks in graph theory.

Theorem 1.1 ([6]) *If G is a minimally 2-connected graph, then for any edge e, $G - e$ decomposes into blocks such that each block is either an edge or a minimally 2-connected subgraph of G.*

Theorem 1.2 ([10]) *(1) A 2-connected graph is minimally 2-connected iff no cycle in the graph contains a chord.*

(2) If G is a minimally 2-connected graph which is not a cycle and S is the set of all degree 2 vertices of G, then $G-S$ is a forest with components $T_1, T_2, \ldots, T_s$ with $s \geq 2$ such that there is no S-path joining two vertices of the same tree T_i.

It follows from the above that,

Lemma 1.3 *Every subgraph of minimally 2-connected graphs contains some vertex x such that at least $d(x) - 1$ of its neighbours are 2^--vertices.*

The following lemma is a key observation. We provide a sketch of the proof.

Lemma 1.4 *If G is a minimally 2-connected graph, then $G - \{u, v\}$ is connected for any edge uv.*

Proof sketch: Observe that u and v have neighbours in every component and uv forms a chord to some cycle in G if there are more than one components. ∎

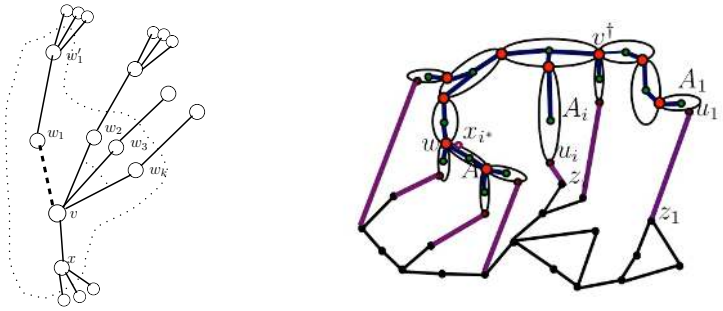


Fig. 1. 1. Structure of *m2c* graph. 2 H and $G - H$ in balanced decomposition.

Induced Matching: An edge colouring is an *induced matching* (aka *strong edge colouring*) if the subgraph induced by the set of vertices of any colour class is the colour class itself. In other words, any two edges of the same colour is separated by a path of length at least 2. The minimum k such that G admits a strong edge colouring using k colours is the strong chromatic index $\chi'_s(G)$. This problem of finding χ'_s is well studied and is notoriously difficult even for simple classes of graphs.

In [8], a conjecture by Erdös and Faudree at al [11] proved that for graphs where all cycle lengths are multiples of four, $\chi'_s(G) \leq \Delta^2$ and conjectured that it is linear in Δ. We prove a stronger result which implies the conjecture. A graph is chordless, if no cycle contains a chord.

Theorem 1.5 *(1) If G is 2-degenerate, then $\chi'_s(G) \leq 10\Delta - 10$.*
(2) If G is chordless, then $\chi'_s(G) \leq 8\Delta - 8$.

Proof Sketch: Let $\mathcal{B} = \{1, 2, \ldots, 4\Delta - 4\}$ and $\mathcal{A} = \{1', 2', \ldots, (4\Delta - 4)'\}$.

From the structure of *m2c* graphs (Lemma 1.3), we pick an edge xy where $d(y) \leq 2$. Inductively assume that for any *m2c* graph G on at most m edges, there is a strong edge colouring $\mathcal{C}$ that satisfies the following.

(1) The colouring $\mathcal{C}$ uses at most $8\Delta - 8$ colours.
(2) Every pendant edge (if any) is coloured from $\mathcal{B}$.
(3) For a pendant edge e coloured c, no edge at a distance 1 is coloured c'.

After colouring xy, if the property fails to hold, we can see that we may recolour the only other edge of y, say zy coloured c, to c'. It can be verified that the conditions of hypothesis still hold and we are done.

Acyclic edge colouring: Acyclic edge colouring is a proper edge colouring in which the union of any 2 colour classes is a forest. The minimum number of colours

needed to acyclically edge colour a graph is its acyclic chromatic index χ'_a. This is another hard problem where even for complete graphs we do not have good upper bounds. The best upper bound known is 16Δ while it is conjectured that for any graph G, $\chi'_a(G) \leq \Delta(G) + 2$, where Δ is the maximum degree. We obtain [9] a simple proof for this conjecture (actually a stronger version) for chordless graphs (graphs where no cycle contains a chord edge) making use of the properties of minimally 2-connected graphs. We give a sketch of the proof.

Theorem 1.6 *Let G be minimally 2-connected. Then $\chi'_a(G) \leq \Delta(G) + 1$.*

Proof Sketch: Note that we can find $uv \in E$ such that all but one neighbour of u have degree at most 2. Inductively $\Delta+1$ colour $G-uv$. It can be easily verified that uv can be properly coloured with some colour. It can create at most one bichromatic cycle since it is incident to a vertex of degree at most 2 and only through the high degree neighbour of u. With some case analysis, we show that one can swap the colours of two edges to avoid the bichromatic cycle.

k-Intersection edge colouring: The notion of k-intersection edge colouring is a generalisation of strong edge colouring and proper edge colouring concepts in a set theoretic sense. We say that a vertex *sees* colour c if c appears on one of it's incident edges. An edge colouring where a pair of adjacent vertices see at most k colours in common is called a *k-intersection edge colouring* (k-IEC). It is easy to see that when $k = 1$ this is strong edge colouring and when $k = \Delta$ it is proper edge colouring. It is known that the k-IEC chromatic index χ'_k is $O(\frac{\Delta^2}{k})$ for all values of k and there are graphs that require $\Omega(\frac{\Delta^2}{k})$.

We prove the following using structure of *m2c* and k-degenerate graphs [4].

Theorem 1.7 *(1) If G is ℓ-degenerate, then $\chi'_k(G) \leq (\ell+1)\Delta - k$ for all $\ell \leq k$.*
(2) If G is a subgraph of a minimally 2-connected graph, then for $k \geq 2$ we have $\chi'_k(G) \leq \max_{uv \in E}(d(u) + d(v) - k + 1)$.

Proof Sketch: Let G be an *m2c* graph on $m + 1$ edges. Choose some vertex $v \in V(G)$ as before. Let w be a 2^--neighbour of v. By the induction hypothesis, $H = G - e$ has a k-intersection edge colouring satisfying the bounds. The rest of the proof is a simple counting of the edges within a distance 1. Similar argument for the ℓ-degenerate case works as they have similar treelike structure.

Balanced Decomposition: A *balanced colouring* of a graph G is a pair (R, B) of subsets $R, B \subseteq V(G)$ such that $R \cap B = \emptyset$ and $|R| = |B|$. For any vertex subset $S \subseteq V$, the *subgraph induced by* S is the graph $G[S]$ with vertex set S and edge set $E[S] = \{xy \in E\colon x, y \in S\}$. A *balanced decomposition* of a balanced colouring (R, B) of G is a partition of vertices $V(G) = V_1 \cup V_2 \cup \ldots \cup V_r$ such that $G[V_i]$ is connected and *balanced*. The *balanced decomposition number* $f(G)$ of a graph G is the maximum over all balanced colourings minimum of $\max_{1 \leq i \leq r} |V_i|$ among all decompositions of G. We prove the following theorem which was conjectured in [2].

Theorem 1.8 *Let G be a 2-connected n-vertex graph. Then, $f(G) \leq \lfloor \frac{n}{2} \rfloor + 1$.*

Proof Sketch: We use the following key lemma.

Lemma 1.9 *If u and v are two distinct vertices in a 2-connected graph G, then*

there is an ordering $u = x_1, x_2, \ldots, x_n = v$ *of* $V(G)$ *such that the graphs* $G_i = G[x_1, x_2, \ldots, x_i]$ *and* $G'_i = G - V(G_i)$ *are connected for* $1 \leq i \leq n$.

Proof sketch: Let $x_1 = u$. Then G_1 and G'_1 are connected since G is 2-connected. Assume that G_{i-1} and G'_{i-1} are connected, where $v \in G'_{i-1}$. By the fact that G is 2-connected, every end block of G'_{i-1} has a non-cut-vertex adjacent to some vertex in G_{i-1}. Choose such a vertex x_i, which can be assumed to be different from v in the case when G'_{i-1} has at least two vertices. ∎

Note that any 2-connected G graph has a balanced decomposition to at least 2 balanced components. Start with a maximal subcomponent H of size at most $\lfloor \frac{n}{2} \rfloor - 1$. Observe that each end-block of $G - H$ has a non-cut vertex adjacent to a vertex in H. Since H is maximal, we can see that each of the neighbours of H should have the same colour (say *blue*). The proof proceeds to show that in such a case, $G - H$ has more *blue* vertices than *red* vertices contradicting the assumption that it was balanced.

References

[1] N. Alon, B. Sudakov and A. Zaks. Acyclic edge colorings of graphs. *Journal of Graph Theory*, 37:157–167, 2001.

[2] S. Fujita and H. Liu. The balanced decomposition number and vertex connectivity. *SIAM J. Discrete Math.*, 24:1597, 2010.

[3] G. J. Chang and N. Narayanan. Strong chromatic index of 2-degenerate graphs. *Submitted*, 2010.

[4] G. J. Chang and N. Narayanan. k-intersection edge colouring of ℓ-degenerate graphs. *Submitted*, 2011.

[5] G. J. Chang and N. Narayanan. On a conjecture on the balanced decomposition number. *Submitted*, 2011.

[6] G. A. Dirac. Minimally 2-connected graphs. *J. Reine Angew Math.*, 228:204–216, 1967.

[7] R. Faudree, A. Gyárfas, R. H. Schelp and Z. Tuza. Induced matchings in bipartite graphs. *Discrete Math.*, 78(1-2):83–87, 1989.

[8] M Molloy and B Reed. A bound on the strong chromatic index of a graph. *Journal of Combinatorial Theory Series B*, 69:103–109, 1997.

[9] R. Muthu, N. Narayanan and C. R. Subramanian. Some graph classess satisfying acyclic edge colouring conjecture. Submitted, 2010.

[10] M.D. Plummer. On minimal blocks. *Transactions of the American Mathematical Society*, 134 No. 1:85–94, 1968.

[11] R.J. Faudree, R.H. Schelp, A. GyÃ¡rfÃ¡s and Zs. Tuza. The strong chromatic index of graphs. *Ars Comb.*, 29(B):205–211, 1990.

Bipartite finite Toeplitz graphs

Sara Nicoloso, [a] Ugo Pietropaoli [b]

[a] *IASI - CNR, Viale Manzoni 30, 00185 Roma, Italy*
nicoloso@disp.uniroma2.it

[b] *Università di Roma Tor Vergata, Dipartimento di Ingegneria dell'Impresa, Via del Politecnico 1, 00133 Roma, Italy*
pietropaoli@disp.uniroma2.it

Key words: Toeplitz graphs, bipartiteness, chromatic number

1 Extended Abstract

Let n, a, b, c be distinct positive integers such that $1 \leq a < b < c < n$ (a, b, c are called *entries*). By $T_n(a, b, c) = (V, E)$ ($T_n(a, b)$, resp.) we denote the simple undirected *finite Toeplitz graph* where $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and $E = \{(v_i, v_j)$, for $|i - j| \in \{a, b, c\}$ ($|i - j| \in \{a, b\}$, resp.), see Fig. 1. If $|i - j| = a$ (b, c, resp.), we say that $v_i, v_j \in V$ are *a-adjacent* (b-, c-adjacent, resp.) and that $(v_i, v_j) \in E$ is an *a-edge* (b-, c-edge, resp.). By *a-path* A_p, $p = 0, 1, \ldots, a - 1$, we denote the path containing vertex v_p and made of a-edges only (notice that all the vertices v_x verifying $x \bmod a = p$ belong to A_p).

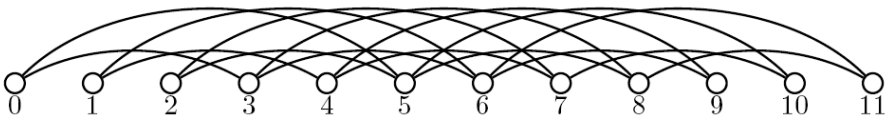


Fig. 1. The Toeplitz graph $T_{12}(3, 5)$.

In the literature, also Toeplitz graphs with an infinite number of vertices have been defined, and the bipartite ones are characterized in [4]. As for finite Toeplitz graphs, in [3] an $O(\log^2(b + 1))$ procedure is proposed to test a $T_n(a, b)$ for bipartiteness, and some results are stated for a subclass of bipartite $T_n(a, b, c)$'s. In this paper we provide a simple alternative condition to test a $T_n(a, b)$ for bipartiteness, and characterize the whole family of bipartite $T_n(a, b, c)$'s. A consequence of these results and those in [5] is a complete characterization of the chromatic number of $T_n(a, b, c)$'s.

The following theorem provides a simple characterization for bipartite $T_n(a, b)$'s.

Theorem 1.1 *Consider a Toeplitz graph $T_n(a,b)$. If $n \leq a+b-\gcd(a,b)$, it is always bipartite; if $n \geq a+b-\gcd(a,b)+1$, it is bipartite if and only if $\frac{a}{\gcd(a,b)}$ and $\frac{b}{\gcd(a,b)}$ are odd.*

As far as $T_n(a,b,c)$'s are concerned, the following result can be proved:

Theorem 1.2 *A Toeplitz graph $T_n(a,b,c)$ with $\gcd(a,b,c) > 1$ is bipartite if and only if $T_{\lceil \frac{n}{\gcd(a,b,c)} \rceil}\left(\frac{a}{\gcd(a,b,c)}, \frac{b}{\gcd(a,b,c)}, \frac{c}{\gcd(a,b,c)}\right)$ is.*

Thanks to this result, in the remainder of the paper we shall limit ourselves to consider $T_n(a,b,c)$'s with $\gcd(a,b,c) = 1$, stating three theorems which give a complete characterization of this class of graphs.

Theorem 1.3 *A Toeplitz graph $T_n(a,b,c)$ with $\gcd(a,b,c) = 1$ and $n \leq c + \gcd(a,b) - 1$ is bipartite if and only if $T_n(a,b)$ is.*

Theorem 1.4 *A Toeplitz graph $T_n(a,b,c)$ with $\gcd(a,b,c) = 1$ and $n \geq \max\{a+b-\gcd(a,b)+1;\, c+\gcd(a,b)\}$ is bipartite if and only if a, b, c are odd.*

All the remaining cases are the Toeplitz graphs $T_n(a,b,c)$ with $\gcd(a,b,c) = 1$ and $c+\gcd(a,b) \leq n \leq a+b-\gcd(a,b)$ which we now focus on. Observe that when b is a multiple of a, i.e. $b = ka$ for some integer k, then the a-path $v_i, v_{i+a}, \ldots, v_{i+ka}$ and the b-edge v_i, v_{i+ka} form a cycle with $k+1$ edges: if k is even, the cycle is odd (and the graph is non-bipartite); if k is odd, the cycle is even: in this case the a-path and the b-edge are both odd paths and, as we shall see later, it suffices to keep just one of them. The same holds if c is a multiple of a and motivates the following definition.

We define $E_n(a,b,c) = [e_{i,j}]$ as the matrix whose rows and columns are indexed from 0 to $a-1$, and from b to $n-1$, respectively, where $e_{i,j} = q$A iff $|i-j| = qa$, for some integer q, or $e_{i,j} =$ B (C, resp.) iff $|i-j| = b$ ($|i-j| = c$, resp.) and b (c, resp.) is not a multiple of a. The *multiplicity* of $e_{i,j}$ is the number of edges it represents. $E_n(a,b,c)$ is a diagonal matrix whose non-empty diagonals are: the leftmost A-*diagonal* of elements kA; a diagonal of B-elements (the B-*diagonal*); a diagonal of C-elements (the C-*diagonal*); and the rightmost A-*diagonal* of elements $(k+1)$A. Depending on the parity of k, either one of the A-diagonals is an *even* A-*diagonal*, and the other one is an *odd* A-*diagonal*. We distinguish two types of matrices: if $\left\lfloor \frac{b}{a} \right\rfloor = \left\lfloor \frac{c}{a} \right\rfloor$ then $E_n(a,b,c)$ is of *type 1*; if $\left\lfloor \frac{b}{a} \right\rfloor < \left\lfloor \frac{c}{a} \right\rfloor$ then $E_n(a,b,c)$ is of *type 2* (see Fig. 2).

	11	12	13	14	15	16
0	B	C		2A		
1		B	C		2A	
2			B	C		2A
3				B	C	
4	1A				B	C
5		1A				B
6			1A			

	19	20	21	22	23	24
0	B		3A			C
1		B		3A		
2			B		3A	
3				B		3A
4					B	
5	2A					B
6		2A				

Fig. 2. The type 1 matrix $E_{17}(7,11,12)$ (left) and the type 2 matrix $E_{25}(7,19,24)$ (right).

In order to investigate the bipartiteness of $T_n(a,b,c)$, we notice that a cycle in the graph must necessarily contain some b- or c-edges (in fact, the subgraph induced by all the a-edges is a collection of disjoint a-paths). We can prove that:

Lemma 1.5 *Consider a $T_n(a,b,c)$. If $n \leq a+b-\gcd(a,b)$, then every b-edge and every c-edge connects the smallest-indexed vertex of an a-path with the largest-indexed vertex of a different a-path, and every vertex has at most one b-edge and at most one c-edge incident to it.*

Two are the consequences of the above lemma. One is that in every cycle $\mathcal{C}$ the maximal sequences of consecutive a-edges are separated by non-empty paths alternating a b- and a c-edge; the other one is that every maximal sequence of consecutive a-edges in a cycle $\mathcal{C}$ is an a-path. These two facts show that on matrix $E(a,b,c)$ all the cycles of $T_n(a,b,c)$ are easily represented, as we now explain.

A cycle $\mathcal{C}$ of $T_n(a,b,c)$ can be mapped onto a $\mathcal{C}$losed $\mathcal{M}$anhattan $\mathcal{C}$urve ($\mathcal{CMC}$, for short) with corners in the non-empty elements of $E_n(a,b,c)$, and viceversa. Precisely: $\mathcal{CMC}$ has a corner in an element $e_{i,j}$ of a kA-diagonal (B- or C-diagonal, resp.) iff all the k a-edges of path A_i (the b-edge (v_i,v_j) or the c-edge (v_i,v_j)) belong to $\mathcal{C}$.

In order to prove the bipartiteness of $T_n(a,b,c)$, we are interested in the parity of the number of edges in a cycle. An easy way to evaluate such parity is to focus on the vertical segments of the corresponding $\mathcal{CMC}$. In fact, the sum of the multeplicity of the endpoints of each vertical segment is an *odd* quantity if and only if one endpoint belongs to the *even* A-diagonal. For this reason, an odd cycle in $T_n(a,b,c)$ is represented by a $\mathcal{CMC}$ of $E_n(a,b,c)$ with an *odd* number of vertical segments incident to elements of the *even* A-diagonal.

Another important observation is related to the "geometry" of a $\mathcal{CMC}$. In fact, the lengths of the vertical/horizontal segments in a $\mathcal{CMC}$ are limited to few quantities: the distances among two consecutive diagonals and the sum of suitable pairs of them. Let k_i, for $i = 1,2,3$ be the distance among the i-th and the $(i+1)$-th diagonal (from left to right), the following diophantine equation can be written

$$k_1 x + k_2 y + k_3 t = 0. \tag{1.1}$$

k_1, k_2, k_3 are easily defined: in a type 1 matrix, $k_1 = b \bmod a$, $k_2 = c - b$, $k_3 = a - c \bmod a$, and, clearly $k_1 + k_2 + k_3 = a$; in a type 2 matrix, $k_1 = b \bmod a$, $k_2 = a - b \bmod a = a\left\lceil \frac{b}{a} \right\rceil - b$, $k_3 = c \bmod a$, and, clearly, $k_1 + k_2 = a$.

A solution (x,y,t) to (1.1) corresponds to a $\mathcal{CMC}$ in $E_n(a,b,c)$ iff it verifies some constraints. First of all we have to ensure that the number of available vertical segments is not exceeded. That is: $|x| \leq U_x$, $|y| \leq U_y$, $|t| \leq U_z$, where U_x, U_y, U_t are obtained by applying a simple elimination procedure, which removes from the matrix those sets of elements corresponding to "dangling" paths. In addition, since we are looking for odd cycles of $T_n(a,b,c)$, we require that a solution contains an *odd* number of vertical segments incident to the *even* A-diagonal. That is: for a type 1 matrix we require x odd if $\left\lfloor \frac{b}{a} \right\rfloor$ even, and t odd if $\left\lfloor \frac{b}{a} \right\rfloor$ odd; for a type 2 matrix we require x odd if $\left\lfloor \frac{b}{a} \right\rfloor$ even, and $y+t$ odd if $\left\lfloor \frac{b}{a} \right\rfloor$ odd. A solution satisfying all the

required constraints will be called a *constrained solution*.

Equation (1.1) admits a solution as $\gcd(k_1, k_2, k_3)$ divides 0 (it takes $O(\log^2 a)$ to compute a solution by Euclid's algorithm), thus it admits an infinite number of them [2] (generated from the first one). Among them, we are interested in a constrained one, if any, as stated by the following theorem (notice that in the worst case we need to generate at most $O(a^2)$ solutions to find a constrained one, if any).

Lemma 1.6 *Consider a* $T_n(a, b, c)$ *with* $\gcd(a, b, c) = 1$ *and* $c + \gcd(a, b) \leq n \leq a + b - \gcd(a, b)$. *Every constrained solution to* (1.1) *corresponds to an odd cycle of* $T_n(a, b, c)$, *and viceversa*.

As a consequence

Theorem 1.7 $T_n(a, b, c)$ *with* $\gcd(a, b, c) = 1$ *and* $c + \gcd(a, b) \leq n \leq a + b - \gcd(a, b)$ *is bipartite if and only if* (1.1) *has no constrained solution*.

We remark that the above results, together with those in [5], completely characterize the chromatic number of Toeplitz graphs $T_n(a, b, c)$. In addition, Theorems 1.1, 1.2 and 1.4 immediately provide a simple characterization for bipartite *infinite* Toeplitz graphs with two or three entries.

References

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. McGraw Hill, 2001.

[2] H. Cohen. *Number Theory: Volume I: Tools and Diophantine Equations*. Springer Graduate Texts in Mathematics, 239, 2007.

[3] R. Euler. Characterizing bipartite Toeplitz graphs. *Theoret. Comput. Sci.*, 263:47–58, 2001.

[4] R. Euler, H. Le Verge, and T. Zamfirescu. Characterizing bipartite Toeplitz graphs. In Ku Tung-Hsin (Ed.), *Combinatorics and Graph Theory '95*, volume 1, pages 119–130, 1995.

[5] S. Nicoloso, and U. Pietropaoli. On the chromatic number of Toeplitz graphs. *Manuscript*, 2010.

A reduction algorithm for the weighted stable set problem in claw-free graphs

Paolo Nobili, [a] Antonio Sassano [b]

[a] *Dipartimento di Matematica, Università del Salento, Lecce, Italy*

[b] *Dipartimento di Informatica e Sistemistica "Antonio Ruberti", Università di Roma "La Sapienza", Roma, Italy*

Key words: claw-free graphs, stable set, clique reduction, matching

1 Introduction

The *Maximum Weight Stable Set Problem (MWSSP)* in a graph $G(V, E)$ with node-weight function $w : V \to \mathbb{R}$ asks for a maximum weight subset of pairwise non-adjacent *nodes*. The Maximum Weight Matching Problem is a special case of the Maximum Weight Stable Set Problem. In fact, the latter can be transformed into the former in a very specific class of graphs: the *line graphs*. The line graph of a graph $G(V, E)$ is the graph $L(G)$ with node set E and an edge ef for each pair $\{e, f\}$ of edges of E incident to the same node. A graph G is a *line graph* if and only if there exists a graph H (the *root graph* of G) with the property that $L(H) = G$. A graph $G(V, E)$ is *claw-free* if no vertex $v \in V$ has three mutually non-adjacent nodes in its neighbourhood. By letting $w(T) = \sum_{v \in T} w(v)$ (where $T \subseteq V$) we have that a stable set S in a claw-free graph has maximum weight if and only if there does not exist a path / cycle (*augmenting path / cycle*) P in G with strictly positive *weight* $w(P \setminus S) - w(P \cap S)$. A graph $G(V, E)$ is *quasi-line* if the neighbourhood of each node $v \in V$ can be covered by two cliques. Each line-graph is a quasi-line graph and each quasi-line graph is a claw-free graph. A 5-wheel $W_5 = (\bar{v}; v_0, v_1, v_2, v_3, v_4)$ is a graph consisting of a 5-hole $R = \{v_0, v_1, v_2, v_3, v_4\}$ and the node $\bar{v}$ adjacent to every node of R. If S is a stable set of $G(V, E)$ then a node $v \in V \setminus S$ is called *superfree* if $|N(v) \cap S| = 0$, *free* if $|N(v) \cap S| = 1$ and *bound* if $|N(v) \cap S| = 2$.

In 1980, Minty [1] proposed a $\mathcal{O}(|V|^6)$ algorithm to find a maximum weight stable set in a claw-free graph. Minty's crucial idea was that of defining a new map (different from the line transformation $L(G)$) from the node-weighted graph G to an edge-weighted graph H with the property that a stable set has maximum (node) weight in G if and only if a suitable matching has maximum (edge) weight in H. For more than twenty years no algorithm better than Minty's was proposed. Re-

cently, an astonishing $\mathcal{O}(|V|^3)$ algorithm has been proposed by Faenza, Oriolo and Stauffer (*SODA 2011*).

This paper proposes a different line of attack by combining the basic ideas of Minty's algorithm with the *clique reduction* by Lovasz and Plummer [2] producing a $\mathcal{O}(|V|^4 log(|V|))$ time algorithm for the same problem. Following Minty, we call a *wing* of $\{s,t\} \subseteq S$ the set $W(s,t) = \{u \in V \setminus S : N(u) \cap S = \{s,t\}\}$. By claw-freeness $\alpha(W(s,t)) \leq 2$; if $\alpha(W(s,t)) = 1$ the wing $W(s,t)$ is said to be a *clique-wing*. An *x-y-alternating path* with respect to S is an induced path $P = \{x, \ldots, y\}$ whose nodes alternate between S and $V \setminus S$ and x (y) is free or belongs to S. An x-y-alternating path P such that x and y are both free is an *x-y-augmenting path.*

2 Reducible and Strongly Reducible Cliques

Let S be a maximal stable set of a claw-free graph $G(V,E)$. A maximal clique Q is *reducible* [2] if $\alpha(N(Q)) \leq 2$. A maximal clique Q in $G(V,E)$ is *strongly reducible* if there exists a node $u \in Q$ (*hinge* of Q) with the property that $N(u) \setminus Q$ induces a clique in G and each node $x \in N(Q) \setminus N(u)$ is adjacent to each node in $Q \setminus \{u\}$.

Definition 2.1 *We call* w-reduction *of the pair* (G,w) *with respect to a* strongly reducible *clique Q with hinge node $u \in S$, the pair (G_Q, w_Q), where $G_Q(V_Q, E_Q)$ is the graph obtained from G by deleting Q and adding all edges between vertices of $N(Q)$. The added edges are called* false edges. *The new weighting vector w_Q indexed by V_Q is defined as $w_Q(s) = w(s)$ if $s \notin N(u)$ and by $w_Q(s) = w(s) - w(u) + w(t_s)$ if $s \in N(u)$, where t_s is any node maximizing $w(t)$ in the set $Q \setminus N(s)$. Each node t_s with this property is called a* companion *of s.*

Theorem 2.1 *Let S be a stable set in G, let Q be a strongly reducible clique with hinge node $u \in S$ and let x, y be two free nodes with respect to S that do not belong to Q. Then an alternating path P is a maximum weight x-y-augmenting path in G with respect to S and w if and only if $P_Q = P \setminus Q$ is a maximum weight x-y-augmenting path in G_Q with respect to $S_Q = S \setminus \{u\}$ and w_Q.*

3 Augmenting subgraph

Let S be a stable set in a claw-free graph $H(U,F)$ and let $\{x,y\} \subseteq U \setminus S$ be a pair of non-adjacent free nodes with respect to S. In this section we first define a subgraph G of H containing all possible x-y-augmenting paths with respect to S in H and then we show that either $\alpha(G) \leq 3$ or G is a line graph or a quasi-line graph which contains a strongly reducible clique.

Definition 3.1 *Let $H(U,F)$ be a connected claw-free graph. Let S be a stable set of H, let $U_F \subseteq U \setminus S$ and $U_{SF} \subseteq U \setminus S$ be the set of* free *and* super-free *nodes with respect to S and let $\{x,y\} \subseteq U_F$ be a pair of non-adjacent free nodes.*

Let $U_R \subseteq U$ be the smallest set containing all the nodes z satisfying one of the following conditions:

(i) $z \in U_{SF} \cup U_F \setminus \{x, y\}$;

(ii) $z \in (N(x) \cup N(y)) \setminus S$;

(iii) $z \in U \setminus S$*: there exists no pair* $\{s, t\} \subseteq U \setminus (S \cup U_R)$ *with the property that* $N(s) \cap N(t) \cap S = \emptyset$ *and* (s, u, z, v, t)*, with* $\{u, v\} = N(z) \cap S$*, is an alternating* P_5.

If x and y do not belong to the same connected component of $H[U \setminus U_R]$ then define $Aug(H, S, x, y)$ to be the empty graph. Otherwise, let $Aug(H, S, x, y)$ be the connected component of $H[U \setminus U_R]$ containing the nodes x and y. We call $Aug(H, S, x, y)$ the Augmenting Subgraph *of H with respect to S and $\{x, y\}$.*

In what follows, we will call *redundant* a node satisfying condition (iii) of Definition 3.1.

Theorem 3.1 *The Augmenting Subgraph $Aug(H, S, x, y)$ contains all the x-y-augmenting paths with respect to S in $H(U, F)$.*

In the rest of this section we will denote by $G(V, E)$ the graph $Aug(H, S, x, y)$.

Theorem 3.2 *If $\alpha(G) \geq 4$ then the augmenting subgraph G does not contain an induced 5-wheel.*

Fouquet proved that a claw-free and 5-wheel-free graph with stability number greater than 3 is a quasi-line graph. Hence, by Theorem 3.2, if $|S| \geq 4$ then G is a quasi-line graph and the neighborhood of each node $u \in S$ is partitioned into two cliques, say Q'_u and $\bar{Q}'_u$. Let $Q_u = Q'_u \cup \{u\}$ and $\bar{Q}_u = \bar{Q}'_u \cup \{u\}$. We prove that Q_u and $\bar{Q}_u$ are maximal cliques.

If $|S| \geq 4$ we denote by $\mathcal{C} = \{\{Q_s, \bar{Q}_s\} : s \in S\}$ a family of clique pairs with the property that $N(s) = Q_s \cup \bar{Q}_s \setminus \{s\}$ and $Q_s \cap \bar{Q}_s = \{s\}$ for each $s \in S$. The clique $\bar{Q}_s$ (Q_s) is said to be the *mate* clique of Q_s ($\bar{Q}_s$) with respect to s. Since S is a maximal stable set of a quasi-line graph G, the family $\mathcal{C}$ always exists and covers all the nodes of G.

Each family $\mathcal{C} = \{\{Q_s, \bar{Q}_s\} : s \in S\}$ partitions the wings $\{W(u, v) : u, v \in S\}$ into cliques, called *partial wings*. Observe that each wing is partitioned into at most two partial wings. If for each node $s \in S$ whose neighborhood $N(s)$ is partitioned into two clique-wings $W(s, v_1)$ and $W(s, v_2)$ we have $\{Q_s, \bar{Q}_s\} = \{W(s, v_1) \cup \{s\}, W(s, v_2) \cup \{s\}\}$ then we call $\mathcal{C}$ an *S-cover* of G.

We are now ready to state the main result of our contribution, namely that the augmenting graph G can be turned into a line graph by a sequence of reductions of strongly reducible cliques.

Theorem 3.3 *Let $|S| \geq 4$ and let $\mathcal{C}$ be an S-cover of the augmenting subgraph G. Then either G is a line graph or $\mathcal{C}$ contains a strongly reducible clique Q, where $Q \setminus S$ is a clique-wing.*

Theorem 3.3 has a natural algorithmic consequence. In order to check the optimality of a stable set S, one has to search, in an S-cover $\mathcal{C}$ of the augmenting subgraph, for a strongly reducible clique Q such that $Q \setminus S$ is a clique-wing. If such a clique

exists then we can reduce the augmenting subgraph to a smaller one. If, conversely, such a clique does not exist then the augmenting subgraph is a line graph and the optimality test can be performed by simply looking for a maximum weight augmenting path with respect to the matching M corresponding to S.

4 A $\mathcal{O}(n^4 \log n)$ Algorithm for MWSSP in claw-free graphs

Gabow's algorithm for Maximum Weight Matching applied to the root graph of a line graph H with n nodes, implies that a maximum weight x-y-augmenting path in H can be found in $\mathcal{O}(n^2 \log n)$. Using this fact, we show that, even if H is claw-free, the same problem can be solved in $\mathcal{O}(n^2 \log n)$. The algorithm consists of two phases. In the first phase, the augmenting subgraph $G = Aug(H, S, x, y)$ is constructed in $\mathcal{O}(n^2)$. In the second phase, a sequence of reductions of strongly reducible cliques is performed, each time updating the resulting graph and stable set in order to finally produce a (reduced) augmenting subgraph G which does not contain a strongly reducible clique anymore. Then, by Theorem 3.3, either G is a line graph or its stable set S satisfies $|S| \leq 3$. In the first case, we find a maximum weight x-y-augmenting path by applying Gabow's algorithm in the root graph; if $|S| \leq 3$ we can easily find a maximum weight x-y-augmenting path in $\mathcal{O}(n^2)$ by enumeration. In both cases, the x-y-augmenting path can be extended, by Theorem 2.1, to a maximum weight x-y-augmenting path in the original graph H in $\mathcal{O}(n)$. Moreover, we can show that the sequence of reductions and updates of the relevant structures can also be performed in $\mathcal{O}(n^2)$. Subsequently, we prove that in a claw-free graph H a maximum weight stable set can be obtained as an extension of a maximum weight stable set of $H - v$ by computing $\mathcal{O}(n)$ maximum weight augmenting paths. This implies an overall complexity of $\mathcal{O}(n^4 \log n)$ to compute a maximum weight stable set of a claw-free graph.

References

[1] Minty, George J., On maximal independent sets of vertices in claw-free graphs, J. Comb. Theory, Ser. B, 28, 284–304, 1980.

[2] Lovász, László and Plummer, M.D., Matching theory, Annals of Discrete Mathematics, 29. North-Holland Mathematics Studies, 121. Amsterdam etc.: North-Holland. XXXIII, 544 p, 1986.

Edge-chromatic sums of regular and bipartite graphs

P.A. Petrosyan, [a] R.R. Kamalian [b]

[a] *Institute for Informatics and Automation Problems, National Academy of Sciences, 0014, Armenia*
pet_petros@ipia.sci.am

[b] *Department of Informatics and Applied Mathematics, Yerevan State University, 0025, Armenia.*
rrkamalian@yahoo.com

Key words: edge-coloring, sum edge-coloring, regular graph, bipartite graph

1 Introduction

We consider finite undirected graphs that do not contain loops or multiple edges. Let $V(G)$ and $E(G)$ denote the sets of vertices and edges of G, respectively. For a graph G, let $\delta(G)$ and $\Delta(G)$ denote the minimum and maximum degree of G, respectively. The degree of a vertex $v \in V(G)$ is denoted by $d_G(v)$, the chromatic number of G by $\chi(G)$ and the edge-chromatic number of G by $\chi'(G)$.

A proper vertex coloring of a graph G is a mapping $\alpha : V(G) \to \mathbf{N}$ such that $\alpha(u) \neq \alpha(v)$ for every $uv \in E(G)$. If α is a proper vertex coloring of a graph G, then $\Sigma(G, \alpha)$ denotes the sum of the colors of the vertices of G. For a graph G, define the vertex-chromatic sum $\Sigma(G)$ as follows: $\Sigma(G) = \min_\alpha \Sigma(G, \alpha)$, where the minimum is taken among all possible proper vertex colorings of G. If α is a proper vertex coloring of a graph G and $\Sigma(G) = \Sigma(G, \alpha)$, then α is called a sum vertex coloring. The strength of a graph G ($s(G)$) is the minimum number of colors needed for a sum vertex coloring of G. The concept of sum vertex coloring and vertex-chromatic sum was introduced by Kubika [9] and Supowit [13]. In [10] Kubika and Schwenk showed that the problem of finding the vertex-chromatic sum is NP-complete in general and polynomial time solvable for trees. Jansen [7] gave a dynamic programming algorithm for partial k-trees. In papers [3,8,11] approximation algorithms were given for various classes of graphs. For the strength of graphs, a theorem like a Brook's one was proved in [6]. On the other hand, there are graphs with $s(G) > \chi(G)$ [4]. Some bounds for the vertex-chromatic sum of a graph were given in [14]. Similar to the sum vertex coloring and vertex-chromatic sum of graphs, in [3,5,6] was introduced sum edge-coloring and edge-chromatic sum of graphs. A proper edge-coloring of a graph G is a mapping $\alpha : E(G) \to \mathbf{N}$ such that $\alpha(e) \neq \alpha(e')$ for every pair of adjacent edges $e, e' \in E(G)$. If α is

a proper edge-coloring of a graph G, then $\Sigma'(G,\alpha)$ denotes the sum of the colors of the edges of G. For a graph G, define the edge-chromatic sum $\Sigma'(G)$ as follows: $\Sigma'(G) = \min_\alpha \Sigma'(G,\alpha)$, where the minimum is taken among all possible proper edge-colorings of G. If α is a proper edge-coloring of a graph G and $\Sigma'(G) = \Sigma'(G,\alpha)$, then α is called a sum edge-coloring. The edge-strength of a graph G ($s'(G)$) is the minimum number of colors needed for a sum edge-coloring of G. For the edge-strength of graphs, a theorem like a Vizing's one was proved in [6]. In [3] Bar-Noy et al. proved that the problem of finding edge-chromatic sum is NP-hard for multigraphs. Later, in [5] it was shown that the problem is NP-complete for bipartite graphs with maximum degree 3. Also, in [5] the authors proved that the problem can be solved in polynomial time for trees and $s'(G) = \chi'(G)$ for bipartite graphs. In [12] Salavatipour proved that the edge-chromatic sum and the edge-strength are NP-complete for r-regular graphs with $r \geq 3$. Also he proved that $s'(G) = \chi'(G)$ for regular graphs. On the other hand, there are graphs with $\chi'(G) = \Delta(G)$ and $s'(G) = \Delta(G) + 1$ [6].

In the present paper we give a polynomial time $\frac{11}{8}$-approximation algorithm for the edge-chromatic sum problem of r-regular graphs for $r \geq 3$. We also present a polynomial time $\frac{13}{9}$-approximation algorithm for the edge-chromatic sum problem of almost regular graphs with additional condition. Finally, we give two complexity results for the problem of finding the edge-chromatic sum of bipartite graphs with maximum degree 3.

2 Two approximation algorithms

A proper edge-coloring with $1, 2, \ldots, t$ colors we call a proper t-coloring. Let G be a graph and $R \subseteq V(G)$. A proper t-coloring of a graph G is called an R-sequential t-coloring [1,2] if the edges incident to each vertex $v \in R$ are colored by the colors $1, 2, \ldots, d_G(v)$.

It is easy to see that the edge-chromatic sum problem of graphs with $\Delta(G) \leq 2$ can be solved in polynomial time. On the other hand, in [12] it was proved that the problem of finding the edge-chromatic sum of an r-regular ($r \geq 3$) graph is NP-complete. Clearly, $\Sigma'(G) \geq \frac{nr(r+1)}{4}$ for any r-regular graph G with n vertices, since the sum of colors appearing on the edges incident to any vertex is at least $\frac{r(r+1)}{2}$. Moreover, it is easy to see that $\Sigma'(G) = \frac{nr(r+1)}{4}$ if and only if $\chi'(G) = r$ for any r-regular graph G with n vertices.

First we give some results on R-sequential colorings of regular and almost regular graphs and then we use these results for constructing approximation algorithms.

Theorem 1. If G is an r-regular graph with n vertices, then G has an R-sequential $(r+1)$-coloring with $|R| \geq \left\lceil \frac{n}{r+1} \right\rceil$.

Theorem 2. If G is a graph with $\chi'(G) = \Delta(G) = r$ and $\Delta(G) - \delta(G) \leq 1$ ($r \geq 3$), then G has an R-sequential r-coloring with $|R| \geq \left\lceil \frac{(r-1)n_r+n}{r} \right\rceil$, where $n = |V(G)|$ and $n_r = |\{v \in V(G) : d_G(v) = r\}|$.

In [3] it was shown that there exists a 2-approximation algorithm for the edge-chromatic sum problem on general graphs. Now we show that there exists a $\frac{11}{8}$-approximation algorithm for the edge-chromatic sum problem on regular graphs and $\frac{13}{9}$-approximation algorithm for the edge-chromatic sum problem on almost regular graphs ($\Delta(G) - \delta(G) \leq 1$) with additional condition.

Theorem 3. For any $r \geq 3$, there is a polynomial time $\left(1 + \frac{2r}{(r+1)^2}\right)$-approximation algorithm for the edge-chromatic sum problem on r-regular graphs.

Theorem 4. For any $r \geq 3$, there is a polynomial time $\frac{13}{9}$-approximation algorithm for the edge-chromatic sum problem on almost regular graphs G with $\chi'(G) = \Delta(G) = r$.

We also consider the problem of finding the edge-chromatic sum of bipartite graphs. In [1,2] it was proved the following:

Theorem 5. If $G = (U, V, E)$ is a bipartite graph with $d_G(u) \geq d_G(v)$ for every $uv \in E(G)$, then G has a U-sequential $\Delta(G)$-coloring.

Corollary. If $G = (U, V, E)$ is a bipartite graph with $d_G(u) \geq d_G(v)$ for every $uv \in E(G)$, then a U-sequential $\Delta(G)$-coloring of G is a sum edge-coloring of G and $\Sigma'(G) = \sum_{u \in U} \frac{d_G(u)(d_G(u)+1)}{2}$.

Finally, we give two complexity results for the problem of finding the edge-chromatic sum of bipartite graphs with $\Delta(G) = 3$.

Problem 1.

Instance: A bipartite graph $G = (U, V, E)$ with $\Delta(G) = 3$.

Question: Is $\Sigma'(G) = \sum_{i=1}^{3} i \cdot |\{u \in V(G) : u \in U \text{ and } d_G(u) \geq i\}|$?

Problem 2.

Instance: A bipartite graph $G = (U, V, E)$ with $\Delta(G) = 3$ and for $i = 1, 2, 3$

$$|\{u \in V(G) : u \in U \text{ and } d_G(u) = i\}| = |\{v \in V(G) : v \in V \text{ and } d_G(v) = i\}|.$$

Question: Is $\Sigma'(G) = \frac{1}{2} \sum_{i=1}^{3} i \cdot |\{w : w \in V(G) \text{ and } d_G(w) \geq i\}|$?

Theorem 6. Problems 1 and 2 are NP-complete.

References

[1] A.S. Asratian, Investigation of some mathematical model of scheduling theory, Doctoral Thesis, Moscow, 1980.

[2] A.S. Asratian, R.R. Kamalian, Investigation on interval edge-colorings of graphs, J. Combin. Theory Ser. B 62 (1994) 34-43.

[3] A. Bar-Noy, M. Bellare, M.M. Halldorsson, H. Shachnai, T. Tamir, On chromatic sums and distributed resource allocation, Inform. and Comput. 140 (1998) 183-202.

[4] P. Erdős, E. Kubika, A. Schwenk, Graphs that require many colors to achieve their chromatic sum, Congr. Numer. 71 (1990) 17-28.

[5] K. Giaro, M. Kubale, Edge-chromatic sum of trees and bounded cyclicity graphs, Inform. Proc. Letters 75 (2000) 65-69.

[6] H. Hajiabolhassan, M.L. Mehrabadi, R. Tusserkani, Minimal coloring and strength of graphs, Discrete Math. 215 (2000) 265-270.

[7] K. Jansen, The optimum cost chromatic partition Problem, in: Proc. of the 3rd Italian Conference on Algorithms and Complexity (CIAC'97), Lecture Notes in Computer Science 1203, 1997, pp. 25-36.

[8] K. Jansen, Aproximation results for the optimum cost chromatic partition problem, J. Algorithms 34 (2000) 54-89.

[9] E. Kubika, The chromatic sum of a graph, PhD Thesis, Western Michigan University, 1989.

[10] E. Kubika, A.J. Schwenk, An introduction to chromatic sums, in: Proc. of the 17th Annual ACM Computer Science Conference, ACM Press, New York, 1989, pp. 39-45.

[11] K. Kubika, G. Kubika, D. Kountanis, Approximation algoriths for the chromatic sum, in: Proc. of the First Great Lakes Computer Science Conference, Lecture Notes in Computer Science 507, 1989, pp. 15-21.

[12] M.R. Salavatipour, On sum coloring of graphs, Discrete Appl. Mathematics 127 (2003) 477-488.

[13] K.J. Supowit, Finding a maximum planar subset of nets in a channel, IEEE Trans. Comput. Aided Design CAD 6(1) (1987) 93-94.

[14] C. Thomassen, P. Erdős, Y. Alavi, P.J. Malde, A.J. Schwenk, Tight bounds on the chromatic sum of a connected graph, J. Graph Theory 13 (1989) 353-357.

Range minimization problems in path-facility location on trees

Justo Puerto, [a] Federica Ricca, [b] Andrea Scozzari [c]

[a] *Universidad de Sevilla,*
puerto@us.es

[b] *"Sapienza" Università di Roma ,*
federica.ricca@uniroma1.it

[c] *Università Telematica delle Scienze Umane "Niccolò Cusano", Roma,*
andrea.scozzari@unisu.it

Key words: Path location, range criterion, length constraint.

1 Extended Abstract

In location analysis, the issue of equity among clients has become a relevant criterion especially when locating public facilities. Equity refers to the distribution of the clients' demand in a geographical area and the objective is to locate facilities in order to ensure a low variability of the distribution of the distances from the demand points (clients) to a facility. In this paper we study the problem of locating path-shaped facilities on a tree network with nonnegative weights associated to the vertices and positive lengths associated to the edges. We consider the maximum and the minimum weighted distances of a client to a facility and minimize the *Range* function which is defined as the difference between the maximum and the minimum weighted distance from the vertices of the network to a facility. This problem arises, for example, when locating a transit line for commuters on a network with the aim of making the line easily accessible to all the clients scattered in a given territory.

The single point location problem under equity measures was considered in [1]. The extension to the case of locating path-shaped facilities of minimum Range can be found in [2], where the problem is formulated in the following three different versions: locating a path which minimizes the Range; locating a path which minimizes the maximum weighted distance subject to the minimum weighted distance bounded below by a constant; locating a path which maximizes the minimum weighted distance subject to the maximum weighted distance bounded above by a constant. However, in [2] all these problems are studied in the special case in which all the vertices of the network have the same weight. Here we extend these results to the more general case of arbitrary vertex weights. We discuss all the above three

formulations of the problem and their generalization including an additional constraint on the length of the path. For the solution of each problem we provide a polynomial time algorithm.

Let $T = (V, E)$ be a tree with $|V| = n$. Suppose that a nonnegative weight w_v is associated to each vertex $v \in V$, while a positive real length $\ell(e) = \ell(u, v)$ is assigned to each edge $e = (u, v) \in E$. For a path P in T, whose endpoints are or may be vertices of T (i.e., discrete and continuous case, respectively) the weighted Range objective function is defined as follows:

$$R(P) = \max_{u \in V \setminus P} w_u d(u, P) - \min_{u \in V \setminus P} w_u d(u, P), \tag{1.1}$$

where $d(u, P)$ denotes the distance from a vertex u of T to the path P, that is, the length of the shortest path from u to a vertex or an endpoint of P. In this paper we investigate the problem of finding a path P in T that minimizes the weighted Range function (Problem P1), along with the two constrained versions of the problem that, for a given nonnegative γ, can be formulated as follows:

$$\begin{aligned} P2: \qquad & \min \max_{u \in V \setminus P} w_u d(u, P) \\ & \min_{u \in V \setminus P} w_u d(u, P) \geq \gamma, \end{aligned} \tag{1.2}$$

and

$$\begin{aligned} P3: \qquad & \max \min_{u \in V \setminus P} w_u d(u, P) \\ & \max_{u \in V \setminus P} w_u d(u, P) \leq \gamma. \end{aligned} \tag{1.3}$$

In addition, we extend the study of all the three above problems also to the case when there is a bound on the length of the path, that is, $L(P) = \sum_{e \in P} \ell(e) \leq L$.

After a preprocessing phase, which is common for all the three problems, in order to solve problem (1.1) in the case when the endpoints of P are not necessarily vertices of the tree, the idea of the algorithm is to root T at a vertex r, and consider the rooted tree T_r. Then, for every pair of edges (u, v) and (k, h) in T_r, among all the paths P with endpoints x and y such that $x \in (u, v)$ and $y \in (k, h)$, one computes the path that minimizes the Range. We show that for solving this problem it is necessary to solve a sequence of linear programming problems each having a constant number of variables and $O(n)$ constraints. Hence, the overall time complexity for finding a (continuous) path P in T that minimizes the Range is $O(n^3)$. We also provide that this complexity remains unchanged if the length constraint is considered. When specializing the method for finding a path P whose endpoints are vertices of T (i.e., the discrete case), we obtain a lower time complexity and we solve problem (1.1), with or without the length constraint, in time $O(n^2)$.

Consider now problems (1.2) and (1.3). For both of them the idea is to take an edge (h, k) and the two subtrees that can be obtained from T by removing it. Denote

by $T(h)$ and $T(k)$ the subtrees rooted at h and k, respectively. The following two propositions hold:

Proposition 1.1 *Let $P(\bar{x}, h)$ be a path in T with $\bar{x}$ fixed in $T(h)$, and such that $\mu(P(\bar{x}, h)) \geq \gamma$. Then, there exists a unique point $\bar{y}$ in (h, k) such that for every $y \neq \bar{y}$ in (h, k) satisfying:*

$$\mu(P(\bar{x}, y)) \geq \mu(P(\bar{x}, \bar{y})) \geq \gamma,$$

one has:

$$E(P(\bar{x}, y)) \geq E(P(\bar{x}, \bar{y})).$$

Proposition 1.2 *Let $P(\bar{x}, h)$ be a path in T with $\bar{x}$ fixed in $T(h)$, and such that $E(P(\bar{x}, k)) \leq \gamma$. Then, there exists a unique point $\bar{y}$ in (h, k) such that for every $y \neq \bar{y}$ in (h, k) satisfying:*

$$E(P(\bar{x}, y)) \leq E(P(\bar{x}, \bar{y})) \leq \gamma,$$

one has:

$$\mu(P(\bar{x}, y)) \leq \mu(P(\bar{x}, \bar{y})).$$

In view of these two propositions, we show that it is possible to solve problems (1.2) and (1.3), again by solving a sequence of linear programming problems with an overall time complexity of $O(n^2)$. Table 8 reports the summary of the complexity results provided in this paper.

Table 8. Summary of the results.

Range-type Problems		without length constraints		with length constraints	
	Problem	Discrete	Continuous	Discrete	Continuous
P1	$\min R(P)$	$O(n^2)$	$O(n^3)$	$O(n^2)$	$O(n^3)$
P2	$\min \max_{u \in V \setminus P} w_u d(u, P)$ s.t. $\min_{u \in V \setminus P} w_u d(u, P) \geq \gamma$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^3)$
P3	$\max \min_{u \in V \setminus P} w_u d(u, P)$ s.t. $\max_{u \in V \setminus P} w_u d(u, P) \leq \gamma$	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^3)$

References

[1] Mesa J.A., Puerto J., Tamir A., Improved algorithms for several network location problems with equality measures. Discrete Applied Mathematics, vol. 130 (2003), 437-448.

[2] Puerto J., Ricca F., Scozzari A., Extensive facility location problems on networks with equity measures. Discrete Applied Mathematics, vol. 157 (2009), 5, 1069-1085.

The price of equity in the Hazmat Transportation Problem

Fabio Roda, [a] Pierre Hansen, [a,b] Leo Liberti [a]

[a] *LIX (UMR CNRS 7161), École Polytechnique, 91128 Palaiseau, France.*
{roda,liberti}@lix.polytechnique.fr

[b] *GERAD and HEC Montreal, Canada.*
pierre.hansen@gerad.ca

Key words: equity, hazmat, multiobjective optimization, transportation problem

1 Introduction

We consider the problem of the transportation of hazardous materials on a road network (Hazardous Materials Transportation Problem). We can figure it this way: there are N trucks which have to transport some kind of dangerous material from one or many production points to one or many garbage dumps and we have to select a set of paths which is optimal from the point of view of risk, cost and equity. The optimization of cost and risk on a network leads quite spontaneously to shortest path and flow problems which are milestones of Operational Research, but equity is somehow unusual and hard to define. We consider and compare two different ideas. The first approach simply requires that all the areas involved in the transportation network share the same level of risk. This is a fair and intuitive idea but it could also lead to "improper" solutions where risk is equal but uniformly high. The second (more interesting) definition of equity we use is inspired by the concept of fairness of J. Rawls [2,3]. Basically, in this context, the difference principle means that we may introduce disparities only if they advantage the worst-off, namely reduce the risk of the less favourite area (the most exposed to the risk).
The aim of this work is to provide rational elements to be able to estimate the cost of choosing a particular definition of equity (for hazmat trasportation). We investigate the relation between each definition of equity and the cost it generates. This can be used as a first criterion to make a choice.

2 Mathematical programming formulation

Let $G = (V, A)$ be a directed graph, modelling a road network.
We consider many origin-destination pairs $(s, t) \in C \subseteq V \times V$. For every pair

(s,t) there is a commodity to be trasported from a source s to a destination t to respond to a specific demand which we indicate with d_{st}. We look for a global route planning given by a multicommodity flow function $x : C \times A \to \mathbb{R}_+$ (the situation involving only one origin and one destination is a special case). Typically we can imagine that the road network covers a geographic area which is divided into zones; in particular each arc (road) belongs to a zone $\zeta \in Z$. For the sake of simplicity we assume that each arc belongs to only one zone. Each arc (i,j) has a positive traversal cost c_{ij}, a probability p_{ij} of an accident occurring on that arc, a value of damage (in monetary units) Δ_{ij} caused by a potential accident on that arc and a capacity χ_{ij}.

(1) **Sets**:
- $C \subseteq V \times V$ is the set of all pairs (s,t);
- Z is the set of all zones;
- $\zeta_l \subseteq A$ is a zone ($1 \leq l \leq |Z|$);

(2) **Parameters**:
- $1 \leq l \leq |Z|$ = zone index
- p_{ij}^{st} : probability of accident on an arc;
- Δ_{ij}^{st} : *damage* (in monetary units) caused by an accident on an arc ;
- c_{ij}^{st} : cost on an arc;
- s : source;
- t : destination (target);
- d_{st} : demand of commodity (st);
  We call $p_{ij}^{st}\Delta_{ij}^{st}$ *traditional risk* and we indicate it, alternatively, as r_{ij}^{st}.

(3) **Decision variables**:
$\forall (i,j) \in A,\ \forall (s,t) \in C\ \ x_{ij}^{st}$: flow of the commodity (st) on the arc (i,j)

(4) **Constraints**.
- (capacity) $\sum_{(st)\in C}\ x_{ij}^{st} \leq c(ij)$
- (demand) $\sum_{(i)\in V} x_{it}^{st} = d_{st}$
- (flow conservation) $\forall (st) \in C \quad \sum\limits_{(i,j)\in A} x_{ij}^{st} - \sum\limits_{(j,i)\in A} x_{ij}^{st} = \begin{cases} 1 \text{ if } i = s \\ -1 \text{ if } i = t \\ 0 \text{ otherwise} \end{cases}$

(5) **Objective function 1 (Cost):** minimize total cost

$$\min \quad \sum_{(i,j)\in A} \sum_{(s,t)\in C} c_{ij}^{st} x_{ij}^{st} \tag{2.1}$$

(6) **Objectives concerning equity, two possible versions:**

- **Objective function 2 (a) (Risk sharing):** minimize the difference of traditional risk between two zones

$$\min \quad \left(\sum_{\forall(\zeta_l,\zeta_m)\in Z\times Z} \left| \left(\sum_{(i,j)\in\zeta_l} \sum_{(s,t)\in C} r_{ij}^{st} x_{ij}^{st} \right) - \left(\sum_{(h,l)\in\zeta_m} \sum_{(s,t)\in C} r_{hl}^{st} x_{ij}^{st} \right) \right| \right) \tag{2.2}$$

- **Objective function 2 (b) (Rawls' principle)**: minimize the traditional risk of the least advantaged zone

$$\min \left(\max_{\zeta\in Z} \sum_{(i,j)\in\zeta} \sum_{(s,t)\in C} r_{ij}^{st} x_{ij}^{st} \right) \tag{2.3}$$

3 Methodology and Tests

The problem we are considering belongs to the special class of optimization problems called Multicriteria Optimization Problems (MOP) [5,7,6]. Probably, the most common approaches are the *weighted sum method* and the ϵ-*constraint method*. Since the latter can be used either with convex or with non-convex objectives space, and we plan to introduce many sources of non-convexity in next refinements of our model, we adopt from the outset the ϵ-*constraint method*. The basic idea of this method consists in the trasformation of all the objectives in constraints, out of one which is minimized (or maximized). Varying ϵ_i, alternative solutions are obtained (even if it is known that it is difficult to chose proper values for the vector ϵ and arbitrarily ones produce no feasible solutions).

We consider first small instances of the problem based on networks composed by a kept down number of nodes and involving a few zones and shipments and only one origin and destination. We used the AMPL modelling environment and the off-the-shelf CPLEX 10.1 solver running on a 64-bit 2.1 GHz Intel Core2 CPU with 4GB RAM. The results we got using an instance composed by 15 nodes distributed in 2 zones, with 10 shipments show that both kind of equity have a negative impact on the cost and make it grow, which is the awaited outcome. The aim is to establish which one makes it increase most. In order to solve this question we introduce some methodological expedients. In fact, even if we solved either C_1 and C_2 applying the ϵ-*constraint method*, we can not simply compare the cost for a fixed equal threshold of ϵ because it has a different meaning in the two situations. We have to normalize the comparison to "equal levels" of equity and to map the cost to the *share* of equity instead of its absolute value. For example, we compare the cost we get when we have the peak of equity in Risk Sharing and Rawls sense, then when we get the 99% of equity (independently from the different corresponding values of ϵ which are different in C_1 and C_2), the 98% ... and so on. Thus we establish first the maximal possible level of equity and then we (can) define the values corresponding to its fractions. We measure the increment of cost while equity varies from its possible minimum to its possible maximum and we map it on the share of equity. We discover that the raise of cost induced by equity in the sense of Rawls

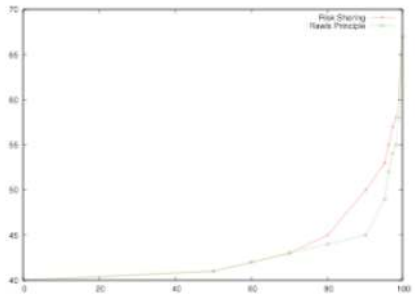


Fig. 1. Comparison between Risk Sharing and Rawls' Principle

is weaker that the one induced by the "naive" one. We report some sample results (in the format [Equity Share; Cost of Risk Sharing; Cost of Rawls' Principle] : [0;40;40], [50;41;41], [60;42;42], [70;43;43], [80;45;44], [90;50;45], [95;53;49], [96;55;52], [97;57;54], [98;58;55], [99;60;58], [100;67;67]). Fig.1 shows the corresponding plot. The tests are partial since we use small artificial instances. We plan to use real data and different multi objectives methods in future work to corroborate our conclusions.

References

[1] G.List and M. Abkowit, *Estimates of current hazardous material flow patterns*, Transportation Quarterly, n. 40, 1986. p. 483-502.

[2] J. Rawls, *A Theory of Justice*, Cambridge (US-MA), Harvard University Press, 1971.

[3] J. Rawls, *Political Liberalism*, Cambridge, (US-MA), Harvard University Press, 1993.

[4] J. N. Hooker, *Optimality Conditions for Distributive Justice*, In: International Transactions in Operational Research Special Issue: Special issue on Ethics and O.R.: Guest Edited by Jean-Pierre Brans, Joao Climaco, Giorgio Gallo and Fred Wenstop, Volume 17, Issue 4, pages 413-426, July 2010.

[5] M. Ehrgott, *Multicriteria optimization* In: Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, 2000.

[6] K. Miettinen, *Nonlinear multiobjective optimization*, Kluwer Academic Publishers, Boston, 1999.

[7] M. Ehrgott and X. Gandibleux, *Multiple Criteria Optimization. State of the art annotated bibliographic surveys*, Kluwer Academic, Dordrecht, 2002.

On graph of order-n with the metric dimension $n-3$

S.W. Saputro, R. Simanjuntak, S. Uttunggadewa, H. Assiyatun, E.T. Baskoro, A.N.M. Salman

Combinatorial Mathematics Research Division, Faculty of Mathematics and Natural Sciences, Institut Teknologi Bandung, Jl.Ganesha 10 Bandung 40132 Indonesia
{suhadi,rino,s_uttunggadewa, hilda, ebaskoro,msalman}@math.itb.ac.id

Key words: basis; diameter; metric dimension; resolving set.

1 Introduction

Throughout this paper, all graphs are finite, simple, and connected. The *vertex set* and the *edge set* of graph G are denoted by $V(G)$ and $E(G)$, respectively. The distance between two distinct vertices $u, v \in V(G)$ is the length of a shortest (u,v)-path in G and denoted by $d_G(u,v)$. For an ordered vertex set $W = \{w_1, \ldots, w_k\}$ of $V(G)$, a *representation* of $v \in V(G)$ with respect to W is defined as k-tuple $r(v \mid W) = (d_G(v,w_1), \ldots, d_G(v,w_k))$. The set W is a *resolving set* of G if every two distinct vertices $x, y \in V(G)$ satisfy $r(x \mid W) \neq r(y \mid W)$. A *basis* of G is a resolving set of G with minimum cardinality, and the *metric dimension* of G refers to its cardinality, denoted by $\beta(G)$.

The metric dimension and the resolving set problem in general graphs was firstly studied by Harary and Melter [3], and independently by Slater [7,8]. These problems have been widely investigated and arise in many diverse areas. These concepts have some applications in chemistry [1], robot navigation [5,6], and network [7,8]. In the navigation of robots in a graph space, for example, a resolving set for a graph corresponds to the presence of distinctively labelled "landmark" nodes in the graph. It is assumed that a robot navigating a graph can detect the distance to each of the landmarks, and hence uniquely determine its location in the graph.

Determining the metric dimension of a graph is a difficult problem generally. Garey and Johnson [2] showed solving this problem to a general graph is NP-complete. However, some characterization results for particular dimensions have been obtained. Chartrand *et al.* [1] and Khuler *et al.* [5] showed that a path is the only graph G with $\beta(G) = 1$. Chartrand *et al.* [1], characterized that K_n is the only graph G with $\beta(G) = n-1$. They also proved that $\beta(G) = n-2$ if and only if G is $K_{r,s}$ for $r, s \geq 1$, $K_r + \overline{K_s}$ for $r \geq 1, s \geq 2$, or $K_r + (K_1 \cup K_s)$ for $r, s \geq 1$.

These are the only known results of characterization of graphs with particular metric dimension.

In this paper, we consider the graphs of order n with metric dimension $n-3$. Generally, Chartrand *et al* [1] showed a connection between the metric dimension and the diameter.

Theorem 1.1 *[1] If G is a connected graph of order $n \geq 2$ and diameter d, then $f(n,d) \leq \beta(G) \leq n-d$, where $f(n,d)$ is the least positive integer k for which $k+d^k \geq n$.*

By Theorem 1.1, the diameter of graph of order n with metric dimension $n-3$ is 2 or 3. Furthermore, Hernando *et al*. [4] have characterized all graphs with metric dimension $n-3$ and diameter 3. In this paper, we classify all graphs of order n, metric dimension $n-3$, and diameter 2. This result completes the characterization of graphs of order n and metric dimension $n-3$.

2 Main Results

We consider the neighborhood of a vertex u in a graph G. The *open neighborhood* of u is $N(u) = \{v \in V(G) \mid uv \in E(G)\}$, and the *closed neighborhood* of u is $N[u] = \{v \in V(G) \mid uv \in E(G)\} \cup \{u\}$. For two vertices u and v in G, u and v are called *non-adjacent twin* if $N(u) = N(v)$, and *adjacent twin* if $N[u] = N[v]$. In both cases, the vertices u and v of G are called *twins*.

We define a relation $\equiv$ on $V(G)$ by $u \equiv v$ if and only if $u = v$ or u and v are twins. Hernando *et al*. [4] showed that $\equiv$ is an equivalence relation. For every $v \in V(G)$, let v^* be a vertex set of G which are equivalent to v under the relation $\equiv$. Let $\{v_1^*, v_2^*, ..., v_k^*\}$ be the partition of $V(G)$ induced by $\equiv$, where v_i is a representative of the set v_i^*. We define G^* as a *twin graph* of G, where $V(G^*) = \{v_1^*, v_2^*, \ldots, v_k^*\}$ and $v_i^* v_j^* \in E(G^*)$ if and only if $v_i v_j \in E(G)$.

By $G[v^*]$ we mean a subgraph of G induced by vertices in v^*. Since each v^* is an equivalence class then $G[v^*]$ is a complete graph or a null graph. We say that $v^* \in V(G^*)$ is of *type*:

- (1) if $|v^*| = 1$,
- (K) if $G[v^*] \cong K_r$ and $r \geq 2$,
- (N) if $G[v^*] \cong N_r$ and $r \geq 2$, where N_r is the *null* graph with r vertices and no edges.

A vertex of G^* is of type $(1KN)$ if it is of type (1) or (K) or (N). A vertex of G^* is of type $(1K)$ if it is of type (1) or (K). A vertex of G^* is of type $(1N)$ if it is of type (1) or (N). A vertex of G^* is of type (KN) if it is of type (K) or (N).

We also consider the neighborhood of vertices in G^*. For each $u^* \in V(G^*)$, we define $nat(u^*) = \{v^* \in V(G^*) \mid u^* \text{ and } v^* \text{ are non-adjacent twin}\}$ and $at(u^*) = \{v^* \in V(G^*) \mid u^* \text{ and } v^* \text{ are adjacent twin}\}$. Note that, $u^* \in nat(u^*)$ and $u^* \in at(u^*)$. We prove that for every $u^*, v^* \in V(G^*)$ where $\beta(G) = n-3$, $|nat(u^*)| \leq 2$, and $|at(v^*)| \leq 3$.

For a twin graph G^* containing u^* with either $|nat(u^*)| = 2$ or $2 \leq |at(u^*)| \leq 3$,

we consider a vertex set of twin graph $V(G^*)$ as $V_1^* \cup V_2^*$ where V_1^* is either $nat(u^*)$ or $at(u^*)$, and $V_2^* = V(G^*) \setminus V_1^*$. By using neighborhood properties, we construct $G^*[V_2^*]$ and thus G^* satisfying $\beta(G) = n - 3$, which can be seen in the following lemmas.

Lemma 2.1 *Let G be a connected graph of order n and diameter 2, and G^* be a twin graph of G containing a vertex u^* with $|nat(u^*)| = 2$. $\beta(G) = n - 3$ if and only if*

(1) $G^* \cong P_3$ with either two leaves of G^* are of type (K) and (KN) respectively, or two leaves of G^* are of type (K) and (1) and the other vertex is of type of (N).

(2) $G^* \cong C_4$ with two adjacent vertices are of type (K) and the other vertices are of type (1).

(3) $G^* \cong \overline{K_2} + P_3$ with two adjacent vertices of P_3 are of type $(1K)$, one vertex of $\overline{K_2}$ is of type (K), and the other vertices are of type (1).

(4) $G^* \cong \overline{K_2} + P_2$ with two vertices of degree 3 are of type (N) and $(1K)$, and two vertices of degree 2 are of type (1) and (K).

(5) $G^* \cong K_1 + (K_1 \cup P_3)$ with the two biggest degree vertices are of type $(1K)$, two vertices of degree 2 are of type (1) and (K), and the other vertex is of type (1).

Lemma 2.2 *Let G be a connected graph of order n and diameter 2, and G^* be a twin graph of G containing a vertex u^* with $|at(u^*)| = 2$. $\beta(G) = n - 3$ if and only if*

(1) $G^* \cong P_2 + P_3$ with a leaf of P_3 is of type (1), one vertex of P_2 is of type (N), and the other vertices are of type $(1K)$.

(2) $G^* \cong \overline{K_2} + P_2$ with two vertices of degree 3 are of type (N) and $(1K)$, and two vertices of degree 2 are of type (1) and (K).

(3) $G^* \cong K_1 + (K_1 \cup P_2)$ with either one vertex of degree 1 is of type (1), one vertex of degree 2 is of type (N), and the other vertices are of type $(1K)$, or the biggest degree vertex is of type $(1K)$, one vertex of degree 2 is of type (K), and the other vertices are of type (N).

Lemma 2.3 *Let G be a connected graph of order n and diameter 2, and G^* be a twin graph of G containing a vertex u^* with $|at(u^*)| = 3$. $\beta(G) = n - 3$ if and only if $G^* \cong K_3$ with at least two vertices of G^* are of type (N).*

For a twin graph G^* containing u^* such that every vertex $u^* \in V(G^*)$ satisfies $|nat(u^*)| = 1$ and $|at(u^*)| = 1$, we prove that $\beta(G) = n - 3$ if and only if $|V(G^*)| = 5$. This condition implies that there are 3 possible classes of G^*.

Lemma 2.4 *Let G be a connected graph of order n and diameter 2, and G^* be a twin graph of G containing a vertex u^* such that every vertex $u^* \in V(G^*)$ satisfies $|nat(u^*)| = 1$ and $|at(u^*)| = 1$. $\beta(G) = n - 3$ if and only if*

(1) $G^* \cong C_5$ with all vertices are of type (1).

(2) G^* is a C_5 with one chord where two adjacent vertices of degree 2 are of type (1) and the other vertices are of type $(1K)$.

(3) $G^* \cong K_1 + P_4$ with either the biggest degree vertex is of type (N) and the other vertices are of type (1), or the two smallest degree vertices are of type (1) and

the other vertices are of type $(1K)$*, or the smallest degree vertex which is of type* (K)*, is adjacent to all vertices of type* $(1K)$ *and the other vertices are of type* (1).

By Lemmas 2.1 - 2.4 together with the results of Hernando *et al.* [4], we obtain the following characterization of graph of order n and metric dimension $n-3$.

Theorem 2.5 *Let G be a connected graph of order n and G^* be a twin graph of G. $\beta(G) = n-3$ if and only if G^* is one of the graphs in Fig. 1.*

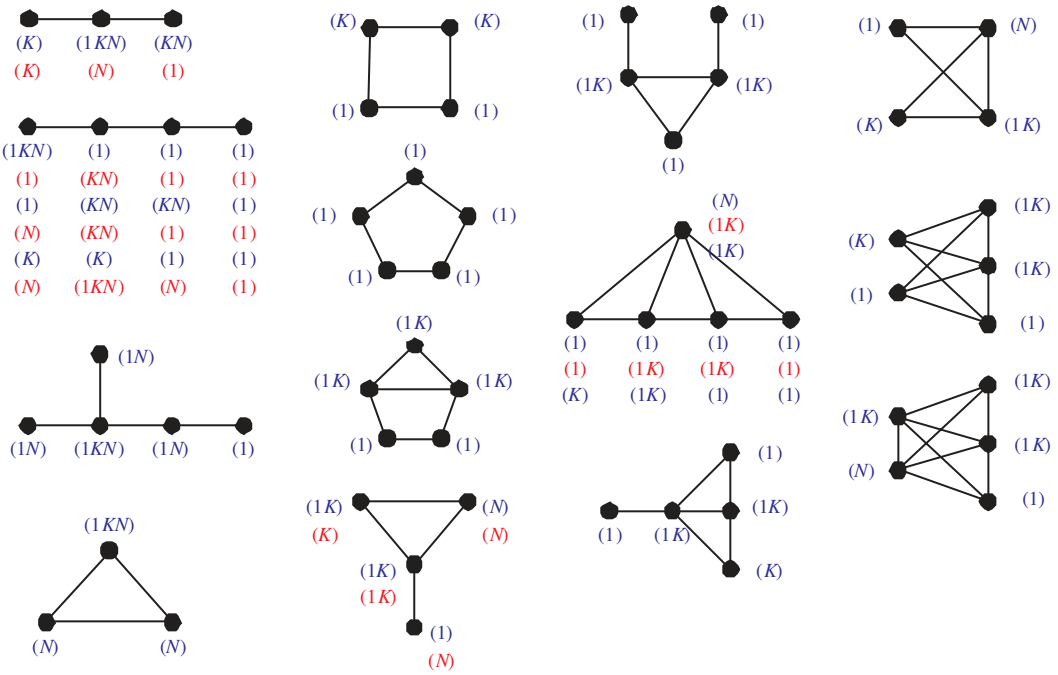


Fig. 1. Twin graph of G of order n with $\beta(G) = n-3$

References

[1] G. Chartrand, L. Eroh, M.A. Johnson, and O.R. Oellermann, Resolvability in graphs and the metric dimension of a graph, *Discrete Appl. Math.*, **105** (2000), 99-113.

[2] M.R. Garey, and D.S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP Completeness*, W.H.Freeman and Company, 1979.

[3] F. Harary, and R.A. Melter, On the metric dimension of a graph, *Ars Combin.* **2** (1976), 191-195.

[4] Carmen Hernando, Mercé Mora, Ignacio M. Pelayo, Carlos Seara, and David R. Wood. Extremal Graph Theory for Metric Dimension and Diameter, *The Electronic Journal of Combinatorics,* **17** (2010), #R30, pp.1-28.

[5] S. Khuller, B. Raghavachari, and A. Rosenfeld, Landmarks in graphs, *Discrete Appl. Math.*, **70** (1996), 217-229.

[6] B. Shanmukha, B. Sooryanarayana, and K.S. Harinath, Metric dimension of wheels, *Far East J. Appl. Math* **8** (3) (2002) 217-229.
[7] P.J. Slater, Leaves of trees, Proc. 6th Southeastern Conf. on Combinatorics, Graph Theory, and Computing, Vol **14** of *Congr. Numer.* (1975) 549-559.
[8] P.J. Slater, Dominating and reference sets in a graph, *J. Math. Phys. Sci.* Vol. **22** (1988), 445-455.

Matchings in balanced hypergraphs

Robert Scheidweiler, Eberhard Triesch

Lehrstuhl II für Mathematik, RWTH-Aachen University, 52056 Aachen, Germany
{scheidweiler,triesch}@math2.rwth-aachen.de

Key words: matching, vertex cover, balanced hypergraph

1 Introduction

The present work deals with the matching and vertex cover problem in balanced hypergraphs. This class of hypergraphs is, according to the definition by Berge in the 70s, one possible generalization of bipartite graphs. Several authors have investigated the matching problem in this class so far (cf. [1],[2], and [3]). On the one hand there are linear programming algorithms, which find maximum matchings and minimum vertex covers in balanced hypergraphs, due to the integrality of associated polytopes. On the other hand no polynomial matching algorithm is known, which makes use of the special combinatorial properties of this class of hypergraphs, e.g. its strong coloring properties. In our opinion this is the main reason for investigating the matching problem in balanced hypergraphs. Hence, the foremost aim of this work is to provide better insight into matching and vertex cover problems for balanced hypergraphs.

2 Basic notions and results

Let $V = \{v_1, \cdots, v_n\}$ be a finite set and $E = \{e_1, \cdots, e_m\}$ a collection of subsets of V. The pair $H = (V, E)$ is called hypergraph. As usual the elements v_i of V are the vertices of H and the elements e_i of E are the edges of H. Let $\{v_0, v_1, \cdots, v_l\} \subseteq V$ and $\{e_1, \cdots, e_l\} \subseteq E$. The sequence $P = v_0 e_1 v_1 e_2 \cdots e_l v_l$ is called a path if $v_{i-1}, v_i \in e_i$ for $i = 1, \cdots, l$ and $v_0, v_1, \cdots v_l$ are pairwise distinct. If $v_{i-1}, v_i \in e_i$ for $i = 1, \cdots, l$, $v_0, v_1, \cdots v_{l-1}$ are pairwise distinct, and $v_0 = v_l$ it is called a cycle. A path resp. cycle is called strong, if there is no edge $e_i, i = 1, \cdots, l$ containing three vertices of the path resp. cycle. Note, that the notions strong cycle and cycle resp. strong path and path are equal in the case of graphs. We define $\deg_H(v) = |\{e \in E \mid v \in e\}|$ the degree of $v \in V, \Delta(H) = \max\limits_{v \in V} \deg_H(v)$, and $\delta(H) = \min\limits_{v \in V} \deg_H(v)$. A hypergraph H with $\Delta(H) = \delta(H)$ is said to be

regular. A subset $M \subseteq E$ of pairwise disjoint edges is called matching of H. We consider an abitrary weight function $d : E \to \mathbb{N}$ of the edges and the special function $d(e) = |e|$ for all $e \in E$. Moreover, we denote the weight of a matching with maximum d-weight by $\Upsilon_d(H)$ resp. by $\Upsilon_V(H)$ for the special function. A matching is perfect, if it covers all vertices of the hypergraph. Let $x \in \mathbb{N}^{|V|}$. Then x is called a d-vertex cover if the inequality $\sum\limits_{v \in e} x_v \geq d(e)$ holds for every edge $e \in E$. x is called minimum if there is no vertex cover $\tilde{x}$ with $\sum\limits_{v \in V} x_v > \sum\limits_{v \in V} \tilde{x}_v$ and we denote the d-vertex cover number by $\tau_d(H) = \sum\limits_{v \in V} x_v$ resp. $\tau_V(H)$.

We call a hypergraph H balanced if H contains no strong cycle of odd length. There are several interesting parallels between bipartite graphs and these hypergraphs. At first we state Kőnig's theorem for balanced hypergraphs.

Theorem 1 *[5][1] Let $H = (V, E)$ be a balanced hypergraph. Then $\Upsilon_d(H) = \tau_d(H)$ for all weight functions $d : E \to \mathbb{N}$. In particular $\Upsilon_V(H) = \tau_V(H)$.*

Kőnig's theorem on edge colorings of bipartite graphs is also valid for this class.

Theorem 2 *[4] Let $H = (V, E)$ be a balanced hypergraph. The edge set E of H can be decomposed into $\Delta(H)$ edge disjoint matchings. In particular, if H is regular, the edge set of H can be divided into $\Delta(H)$ edge disjoint and perfect matchings.*

In the next theorem we investigate the matching number under the condition that a balanced hypergraph does not differ too much from regularity.

Theorem 3 *Let $H = (V, E)$ be a balanced hypergraph.*

- *If $\sum\limits_{v \in V} (\Delta(H) - \deg_H(v)) \leq q\Delta(H) - 1$, then $\Upsilon_V(H) \geq |V| - q + 1$.*
- *If $\sum\limits_{v \in V} (\Delta(H) - \deg_H(v)) \leq \Delta(H) - 1$, then H has a perfect matching.*
- *If $\sum\limits_{v \in V} (\deg_H(v) - \delta(H)) \leq \delta(H) - 1$, then H has a perfect matching.*

3 Decomposition Theory

The Gallai-Edmonds decomposition is a standard tool in matching theory for ordinary graphs. It is therefore natural to ask for a hypergraph version of this decomposition, especially in the case of balanced hypergraphs, for which the matching problem is known to be polynomial-time solvable. Our decomposition of the edge set of a balanced hypergraph generalizes the so called persistency partition, introduced by Costa for bipartite graphs and their maximum matchings. In the following we define the hypergraphs $H \backslash v := (V \backslash v, \{e \backslash \{v\} \mid e \in E\})$ and $H \backslash e := (V, E \backslash \{e\})$.

Theorem 4 *Let $H = (V, E)$ be a balanced hypergraph and $d : E \to \mathbb{N}$ a weight function. Then for the following six vertex and edge sets it holds:*

$$
\begin{aligned}
V_1^d(H) &:= \{v \in V \mid x_v > 0 \; \forall \textit{ min } d\textit{-vertex covers } x \textit{ of } H\} \\
&= \{v \in V \mid \Upsilon_d(H) < \Upsilon_d(H \setminus v)\}, \\
V_2^d(H) &:= \{v \in V \mid \exists \textit{ two min } d\textit{-vertex covers } x^1, x^2 \textit{ with } x_v^1 > 0, x_v^2 = 0 \; \} \\
&= \{v \in V \mid \Upsilon_d(H) = \Upsilon_d(H \setminus v) \textit{ and } v \in V(M) \; \forall \; d\textit{-max matchings } M\}, \\
V_3^d(H) &:= \{v \in V \mid x_v = 0 \; \forall \textit{ min } d\textit{-vertex covers } x \textit{ of } H\} \\
&= \{v \in V \mid \exists \; d\textit{-max matching } M \textit{ with } v \notin V(M)\}, \\
E_1^d(H) &:= \{e \in E \mid e \in M \; \forall \; d\textit{-max matchings } M\} \\
&= \{e \in E \mid \tau_d(H) > \tau_d(H \setminus e)\}, \\
E_2^d(H) &:= \{e \in E \mid \exists \textit{ two } d\textit{-max matchings } M^1 \textit{ and } M^2 \textit{ with } e \in M^1 \textit{ and } e \notin M^2 \; \} \\
&= \{e \in E \mid \tau_d(H) = \tau_d(H \setminus e) \textit{ and } \sum_{v \in e} x_v = d(e) \; \forall \textit{ min } d\textit{-vertex covers } x\}, \\
E_3^d(H) &:= \{e \in E \mid e \notin M \; \forall \;\; d\textit{-max matchings } M \textit{ of } H\} \\
&= \{e \in E \mid \tau_d(H) = \tau_d(H \setminus e) \textit{ and } \exists \textit{ min } d\textit{-vertex cover } x \textit{ with } \sum_{v \in e} x_v > d(e)\}.
\end{aligned}
$$

If we consider maximum matchings regarding to the number of covered vertices, we prefer a slightly different decomposition of the vertex set.

Theorem 5 *Let $H = (V, E)$ be a balanced hypergraph. Then it holds for the following vertex sets:*

$$
\begin{aligned}
V_1^V(H) &:= \{v \in V \mid x_v \geq 2 \; \forall \textit{ min } V\textit{-vertex covers } x\} \\
&= \{v \in V \mid \Upsilon_V(H) \leq \Upsilon_V(H \setminus v) \textit{ and } v \in V(M) \; \forall \; V\textit{-max matchings } M\}, \\
V_2^V(H) &:= \{v \in V \mid \exists \textit{ min } V\textit{-vertex cover } x \textit{ with } x_v = 1\} \\
&= \{v \in V \mid \Upsilon_V(H) - 1 = \Upsilon_V(H \setminus v)\}, \\
V_3^V(H) &:= \{v \in V \mid x_v = 0 \; \forall \textit{ min } V\textit{-vertex covers } x\} \\
&= \{v \in V \mid \exists \; V\textit{-max matching } M \textit{ with } v \notin V(M)\}.
\end{aligned}
$$

4 Hall's theorem

In this chapter we will give a new, short, and combinatorial proof of Hall's theorem for balanced hypergraphs by means of our decomposition theory.

Theorem 6 *[2] A balanced hypergraph $H = (V, E)$ has a perfect matching if and only if H satisfies the Hall condition, i.e., for all disjoint $A, B \subseteq V$ with $|A| > |B|$ there exists an edge $e \in E$ with $|e \cap A| > |e \cap B|$*

Proof. The "only if" direction is true because any hypergraph with a perfect matching clearly satisfies the Hall condition.

Suppose for the "if" direction that there is a balanced hypergraph $H = (V, E)$, which satisfies the Hall condition and has no perfect matching. Choose H with

these properties such that $|V|$ is minimal. Hence, $V = V_1^V(H) \cup V_3^V(H)$. Assume that $x_v = 2$ for all $v \in V_1^V(H)$ in a minimum vertex cover x. By Theorem 1, $2|V_1^V(H)| = \Upsilon_V(H)$, whereas the minimality of $|V|$ implies that $\Upsilon_V(H) = n - 1$. Hence,

$$2|V_1^V(H)| = n - 1 \Leftrightarrow |V_1^V(H)| + 1 = |V_3^V(H)|.$$

Moreover, $|e \cap V_3^V(H)| \leq |e \cap V_1^V(H)|$ for all $e \in E$, otherwise x would not be a vertex cover. Thus, the sets $A = V_3^V(H)$ and $B = V_1^V(H)$ violate the Hall condition. Therefore a vertex $v^* \in V$ with $x_{v^*} \geq 3$ exists for any minimum vertex cover x of H and $|V_3^V(H)| > |V_1^V(H)| + 1$.
For each $v \in V_3^V(H)$ we choose a matching M_v, which is a maximum matching of H not containing v. Now we consider the hypergraph

$$\tilde{H} = \left(V, \overset{*}{\bigcup_{v \in V_3^V(H)}} M_v \right).$$

(The union $\overset{*}{\bigcup}$ denotes a multiset union.) It holds $\deg_{\tilde{H}}(v) = |V_3^V(H)|$ for all $v \in V_1^V(H)$ and $\deg_{\tilde{H}}(v) = |V_3^V(H)| - 1$ for all $v \in V_3^V(H)$. Hence, we obtain $\sum_{v \in V} (\deg_{\tilde{H}}(v) - \delta(\tilde{H})) = |V_1^V(H)| \leq |V_3^V(H)| - 2 = \delta(\tilde{H}) - 1$. Therefore, because of Theorem 3, $\tilde{H}$ has a perfect matching and so we can conclude that H has a perfect matching, too. This is a contradiction.

References

[1] Delbert R. Fulkerson, Alan J. Hoffman, and Rosa Oppenheim, On balanced matrices, Mathematical Programming Study 1, 1974, p. 120–132

[2] Michele Conforti, Gérard Cornuéjols, Ajai Kapoor, and Kristina Vusković, Perfect matchings in balanced hypergraphs, Combinatorica 16 Issue 3, 1996, p. 325–329

[3] Andreas Huck and Eberhard Triesch, Perfect matchings in balanced hypergraphs - a combinatorial approach, Combinatorica 22 Issue 2, 2002, p. 409–416

[4] Claude Berge, Sur certains hypergraphes généralisant les graphes bipartites, Combinatorial Theory and its Applications I, Colloquium of Mathematical Society Janos Bolyai 4, 1970, p. 119–133

[5] Claude Berge and Michel Las Vergnas, Sur un théorème du type König pour hypergraphes, Annals of the New York Academy of Science 175, 1970, p. 32–40

Efficient enumeration of optimal and approximate solutions of a scheduling problem

Sergey Sevastyanov, [a 1] Bertrand M.T. Lin [b 2]

[a] *Sobolev Institute of Mathematics, Novosibirsk State University, Novosibirsk, 630090, Russia*
seva@math.nsc.ru

[b] *Institute of Information Management, Department of Information and Finance Management, National Chiao Tung University, Hsinchu, Taiwan*
bmtlin@mail.nctu.edu.tw

Key words: listing algorithm, polynomial delay, polynomial space, flowshop scheduling, makespan, permutational schedule

1 Extended Abstract

When dealing with optimization problems, one normally searches for a "best solution" with respect to a certain objective function. But in reality, the decision-making process is not that simple, and while solving a real life problem, we choose a "proper" solution from among those ones that are good enough with respect to a given criterion, while our ultimate choice depends on a variety of difficult-to-formalize conditions. In that case we may be interested in enumerating (without repetition) the set of all approximate solutions that meet a given bound on inaccuracy.

An interesting special issue is enumerating all optimal solutions of a specified optimization problem. Clearly, this could be always done by the direct enumeration of all feasible (and even not necessarily feasible) solutions, but in most cases such a solution approach is impracticable. And theoretically, we cannot treat such a method as an efficient one. However, in some cases, while solving the enumeration problem, we cannot avoid the complete enumeration of all solutions. (For example, this happens in the TSP problem when all $n!$ possible routes through n towns have the same total length.) Clearly, such a situation happened for a particular problem instance cannot be an indicator of inefficiency of our algorithm. On

[1] Supported by the Russian Foundation for Basic Research (grants nos. 08-01-00370 and 08-06-92000-HHC-a) and by the Federal Target Grant "Scientific and educational personnel of innovation Russia" for 2009–2013 (government contract no. 02.740.11.0429).
[2] Partially supported by the National Science Council of Taiwan (grants no. 98-2410-H-009-011, 98-2811-H-009-003).

the other hand, when we can guarantee for our algorithm that, whatever problem instance is given, it will enumerate proper and only proper solutions, such an algorithm cannot be treated as inefficient (even if the output is very large for some problem instances). If, in addition, finding each proper solution requires small (*i.e.*, polynomially bounded) time and space, we come to the notions of so called *polynomial delay* and *polynomial space* algorithms.

Definition 1.1 We say that a listing algorithm has *polynomial delay* if the time required for generating each new output element and the time needed to halt the algorithm after the last output are polynomial in the input size.

Definition 1.2 We say that a listing algorithm runs with *polynomial space* if at any step it consumes memory space polynomial in the input size.

Although the direction of enumerating optimal solutions for combinatorial problems is known for decades (since the paper by Johnson et al. (1988), where the notions of polynomial delay and polynomial space algorithms were introduced), in scheduling theory this research direction can hardly be positioned as "well developed". The more so this is valid with respect to designing algorithms of enumerating approximate solutions. (And very likely, such a research direction should be treated as novel in discrete optimization in all.)

In our paper we undertake the first (up to our knowledge) attempt of that kind investigation with respect to the classical scheduling problem stated by Johnson in his seminal paper (Johnson, 1954) from which the contemporary machine scheduling theory took its origin. In his paper, Johnson presented an efficient algorithm for solving the following two-machine problem. (As is known now due to Garey, Johnson and Sethi (1976), the corresponding three-machine problem is strongly NP-hard.)

Problem setting. We are given n jobs $\{J_1, \ldots, J_n\} = \mathcal{J}$ that are to be processed on two machines, M_1 and M_2. Each job J_j has exactly two operations O_{j1} and O_{j2} that must be executed in a strictly defined order: first O_{j1} on machine M_1, and then O_{j2} on machine M_2, with predefined processing times a_j and b_j, respectively. Each machine may process its operations also consecutively, in some order that should be chosen in the course of problem resolution. The objective is searching for a schedule S with the minimum length, or the minimum *makespan* denoted as $C_{\max}(S)$. According to the common three-field classification [5], the problem is normally denoted as $F2 \,||\, C_{\max}$.

As follows from the general project scheduling theory, to minimize the makespan in flowshop scheduling, it suffices to restrict our consideration to *active schedules* only, each of which is uniquely defined by the sequences of operations on machines M_1 and M_2. Moreover, as shown by Johnson (1954), we need not enumerate all combinations of the two sequences. It suffices to consider only pairs of identical sequences on the two machines specified by a unique permutation π of jobs, assuming that the operations of all n jobs on each machine (including zero length operations) are the elements of the permutation. Such schedules (S_π) are called *permutational*.

In his analysis of optimality of a given sequence, Johnson defined a relation "$\preceq$" on the set of jobs: $J_i \preceq J_j$, if $\min\{a_i, b_j\} \leq \min\{a_j, b_i\}$.
He found a sufficient condition for a sequence of jobs to be optimal (*Johnson's rule*): a permutation of jobs (job indices) $\pi = (\pi_1, \ldots, \pi_n)$ is optimal, if

$$J_{\pi_i} \preceq J_{\pi_j}, \text{ for all } i < j. \tag{1.1}$$

And though the relation "$\preceq$" does not define a linear order (for it is not transitive: the relations $J_1 \preceq J_2,\ J_2 \preceq J_3$ do not necessarily imply $J_1 \preceq J_3$), Johnson showed that for any problem instance there always exist job sequences (further referred to as *Johnson's sequences*) with property (1.1). Some of those sequences can be found in $O(n \log n)$ time, by implementing a so called *special Johnson's rule*.

It can be seen that in the case when all operation processing times are different, Johnson's rule defines a unique job sequence, because relation (1.1) becomes transitive and anti-symmetric. Yet in general, Johnson's rule yields a wide variety of optimal job sequences. Among the papers addressing the issue of enumeration of all Johnson's sequences, we would like to mention the book by Bellman, Esogbue and Nabeshima (1982), where a so called *General Working Rule* was derived, enabling one to generate any Johnson's sequence. This rule can be transformed into an efficient procedure of enumerating all Johnson's sequences without repetition. However, it is clear that in general case the set of optimal sequences is not limited to Johnson's sequences only, because rule (1.1), while being sufficient, is not necessary for a sequence of jobs to be optimal. So, the problem of enumerating all optimal permutations of jobs for $F2\,||\,C_{\max}$ problem is more general, and is of its own interest. A number of authors addressed this issue, trying to design enumeration procedures that would be more effective than the direct enumeration of all $n!$ permutations of n jobs (see, e.g., [2], [8]–[10]). Yet they could guarantee the efficiency of their procedures neither in polynomial delay, nor in polynomial space senses. As was claimed by Cheng (1992), the problem of designing an efficient algorithm for enumerating all optimal sequences for $F2\,||\,C_{\max}$ remained open that time.

In our paper we present the first, up to our knowledge, algorithm that enumerates with polynomial delay and polynomial space all optimal permutations of jobs in the two-machine flow-shop problem. In fact, our result is more general, since it concerns the issue of the complete enumeration of all **approximate solutions** that meet a given bound on inaccuracy.

Definition 1.3 A sequence π of job indices is called Δ-*optimal*, if $C_{\max}(S_\pi) \leq B + \Delta$, where B is the total load of machine M_2.

Our main result is designing an algorithm $\mathcal{A}(\Delta)$ with the following property.

Theorem 1.1 *Algorithm $\mathcal{A}(\Delta)$ enumerates all Δ-optimal permutations of n jobs without repetition, while requiring $O(n \log n)$ delay for generating each Δ-optimal permutation and for halting, when all proper permutations are enumerated, and consuming $O(n)$ memory space at every step.*

Clearly, this algorithm can also be used for enumerating all optimal sequences.

Our next result concerns the structure of the set of optimal solutions. Given a problem instance, let its Δ-optimal permutations of jobs be represented as vertices of a graph G_Δ. Two permutations π' and π'' are connected by an edge, iff one permutation can be obtained from the other by a transposition of two neighboring elements. We prove the following

Theorem 1.2 *For any $\Delta \geq 0$ graph G_Δ is connected.*

Corollary 4 *Any optimal permutation of jobs can be obtained from any Johnson's (optimal) permutation by consecutive transpositions of neighboring elements, so that all intermediate permutations are also optimal.*

References

[1] R. BELLMAN, A.O. ESOGBUE, AND I. NABESHIMA (1982). *Mathematical aspects of scheduling and applications*, Pergamon Press, Oxford.

[2] J.C. BILLAUT AND P. LOPEZ (1997). New results for the enumeration of optimal job sequences, Unpublished manuscript.

[3] T.C.E. CHENG (1992). Efficient implementation of Johnson's rule for the $n/2/F/F_{\max}$ scheduling problem, *Computers ind. Engng.*, **22**, 495–499.

[4] M.R. GAREY, D.S. JOHNSON, AND R. SETHI (1976). The Complexity of Flowshop and Jobshop Scheduling, *Mathematics of Operations Research*, **1**, 117–129.

[5] R.L. GRAHAM, E.L. LAWLER, J.K. LENSTRA, AND A.H.G. RINNOY KAN (1979). Optimization and approximation in determenistic sequencing and scheduling: A survey. *Annals of Discrete Mathematics* **5**, 287–326.

[6] S.M. JOHNSON (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, **1**, 61–67.

[7] D.S. JOHNSON, M. YANNAKAKIS, AND C.H. PAPADIMITRIOU (1988). On generating all maximal independent sets. *Information Proc. Letters*, **27**, 119–123.

[8] Y. LIN AND J. DENG (1999). On the structure of all optimal solutions of the two-machine flowshop scheduling problem, *OR Transactions*, **3**(2), 10–20.

[9] S.N.N. PANDIT AND Y.V. SUBRAHMANYAM (1975). Enumeration of all sequences, *Opsearch*, **12**, 35–39.

[10] W. SZWARC (1981). Extreme solutions of the two machine flow shop problem, *Naval Research Logistics Quarterly*, **28**(1), 103–114.

Majorization and the minimum number of dominating sets

Zdzislaw Skupień

AGH Kraków, al. Mickiewicza 30, 30–059 *Kraków, Poland*
skupien@agh.edu.pl

Key words: counting dominating sets, domination number, majorization, numerical partition

1 Abstract

A recent Wagner result, which characterizes graphs with minimum number of dominating sets, is based on an earlier characterization restricted to trees. A self-contained proof of both characterizations is presented. Unions of disjoint stars and majorization among related numerical partitions are objects and a tool of investigation. A characterization limited to graphs with upper-bounded domination number is established.

2 Results

A vertex subset S of a graph G is called a *dominating set* if each vertex of $G - S$ is adjacent to a member of S. Let $\partial(G)$ stand for the number of dominating sets in G. The following statement presents parts of two results, one on trees [2, Theorem 13] and another, due to Wagner [7, Corollary 3], on any graphs.

Given a positive integer k and $n = 3k$, any tree [graph] G on k vertices is an induced subgraph of an n-vertex tree [graph], H, with smallest number of dominating sets among n-vertex trees [graphs without isolated vertices] provided that all $2k$ additional vertices of H are leaves in H, which in pairs are joined to each vertex of G.

Therefore H is an example of a *graph built on* a UDS (the *union of disjoint stars*), which means that a UDS is a spanning subgraph (a factor) of H and all rays in UDS are pendant edges of H. Our aim is to present a self-contained proof of the above-mentioned Wagner theorem. The proof refers to UDSs, to partitions of vertices therein, and to majorization among corresponding numerical partitions. Majorization is very likely to help attacking new characterization problems in enumerative

domination theory. Besides, equipartition in the solution to an extremal characterization problem seems symptomatic of majorization hidden behind, cp. the famous Turán theorem and its extensions.

Lemma 1 *Let $S(r,\ell)$ be a star with ℓ rays, $\ell \geq 1$, and with a vertex designated x_o, a root, which necessarily is the center of the star if $\ell > 1$. Let a* UDS *comprise such stars as components. Then the numbers of dominating sets are*

$$\partial(S(r,\ell)) \quad = 1 + 2^{\ell}, \tag{2.1}$$

$$\partial(\mathrm{UDS}) = \prod_{S} (1 + 2^{\ell(S)}) \tag{2.2}$$

where S ranges over components of UDS *and $\ell(S)$ is the count of rays in S. Moreover, let G be a graph with a* UDS *as a spanning subgraph, say F. Then*

$$\partial(G) \geq \partial(\mathrm{UDS}) \tag{2.3}$$

where equality holds if and only if G is built on the UDS, $\mathrm{UDS} = F$.

The importance of UDSs in domination theory is established in [1].

Proposition 2 *Given a graph G without isolated vertices, a* UDS *which is a spanning subgraph of G can be found in time linear in the number $\|G\|$ of edges of G.*

We thus show that characterizing graphs with minimum ∂ reduces to the study of UDSs.

In general, the structure of the n-vertex graph H, with minimum $\partial(H)$, depends on the remainder $n \bmod 3$, which in what follows is represented by an integer r, $r = 0, \pm 1, -2$.

Theorem 3 *Assume that $n > 1$ and $r = 0, \pm 1, -2$ is such that $r \equiv n \pmod 3$. Let H be an n-vertex graph without isolated vertices and with smallest number of dominating sets. Then H is any graph built on a* UDS *with $(n-r)/3$ components such that all components are paths P_3 with no exception ($r = 0$), except one component which is P_2 ($r = -1$), or (for $n \bmod 3 = 1$) except either one component $K_{1,3}$ ($r = 1$) or two components both being P_2 ($r = -2$).*

Corollary 4 *Let any $n > 1$ and let H and r be as assumed in Theorem. Then any graph G of order $(n-r)/3$ is an induced subgraph of some of graphs H. Moreover,*

$$\partial(H) = \begin{cases} 3^{|r|} \cdot 5^{\lfloor n/3 \rfloor} & \text{if} \quad 3|n-r \text{ and } r = 0, -1, \\ 9 \cdot 5^{\lfloor (n-4)/3 \rfloor} & \text{if} \quad n \equiv 1 \pmod 3 \text{ and } r = 1, -2. \end{cases} \tag{2.4}$$

The characterization result of the above theorem will be established via minimization of the RHS in formula (2.2). This is the minimization over the distribution of the rays among components in a UDS, the distribution being represented by a numerical partition of the total number of rays. Next we refer to majorization [6] among partitions in order to characterize UDS of order n, with k stars and with minimum $\partial(\mathrm{UDS})$, $1 \leq k \leq n/2$. Finally we find the global minimum, the min-

imum over the number k of stars. To this end, we refer to $z := n/k$, the average number of vertices among stars in the UDS.

Theorem 3 and its corollary present the main Wagner result on the minimum number of dominating sets. The following result is equivalent to Wagner's [7, Theorem 4].

Theorem 5 *Assume that G ranges over n-vertex graphs which are built on* UDS*s with k components. Then there are integers a and b such that*
$n = (a+1)k + b$ *with* $a \geq 1$ *and* $0 \leq b < k$.
Moreover, $\partial(G)$ attains the minimum value, say

$$\partial_o := (1+2^a)^{k-b}(1+2^{a+1})^b, \tag{2.5}$$

if and only if G is built on the equipartite UDS*, with equipartition of the $n-k$ rays being represented by the integral k-partition $x := (\{a+1\}^b, \{a\}^{k-b})$.*

What follows is a new result.

Theorem 6 *Given positive integers n and k, let $\mathcal{G}_n(k)$ stand for the class of n-vertex graphs without isolated vertices and with domination number $\gamma \leq k$ where $n \geq 2k$. Each member of $\mathcal{G}_n(k)$ with minimum number ∂ of dominating sets is a graph built on an equipartite* UDS *with at most k components, the number of components being k (as in Theorem 5) if $k < (n+2)/3$, but for a greater k, i.e. $(n+2)/3 \leq k \leq n/2$, is (as in Theorem 3) either $(n-1)/3$ or $(n+2)/3$ if $n \equiv 1 \pmod 3$, and is $\lfloor (n+1)/3 \rfloor$ otherwise.*

References

[1] D. Bauer, F. Harary, J. Nieminen, C.L. Suffel, Domination alteration sets in graphs, Discrete Math. 47 (1983) 153–161.

[2] Dorota Bród, Z. Skupień, Trees with extremal numbers of dominating sets, Australas. J. Combin. 35 (2006) 273–290.

[3] E. Cockayne, S. Goodman, S. Hedetniemi, A linear algoritm for the domination number of a tree, Inf. Proc. Lett. 4 (1975) 41–44.

[4] T.W. Haynes, S.T. Hedetniemi, P.J. Slater, Fundamentals of Domination, Marcel Dekker, Inc., 1997.

[5] J. Nieminen, Two bounds for the domination number of a graph, J. Inst. Math. Appl. 14 (1974) 183–187.

[6] A.W. Marshall, I. Olkin, Inequalities: Theory of Majorization and Its Applications, Acad. Press, New York et al., 1979.

[7] S. Wagner, A note on the number of dominating sets of a graph, to appear.

Reducing the minimum T-cut problem to polynomial size linear programming

Rüdiger Stephan,

Institut für Mathematik, Technische Universität Berlin,
Straße des 17. Juni 136, 10623 Berlin
stephan@math.tu-berlin.de

Key words: T-cut, Gomory-Hu-tree, matching, compact formulation

1 Extended Abstract

Systems of linear inequalities that fully describe the feasible solutions of a combinatorial optimization problem (P) are called *compact* if they use a polynomial number of variables and inequalities. If such a system is known, problem (P) can be solved in polynomial time using any polynomial algorithm for linear programming. Twenty years ago Yannakakis [6] raised the question whether, or not, there exists a compact formulation for the *perfect matching polytope* which is the convex hull of all perfect matchings of a graph G. He gave a partial answer to this question by showing that there exists no *symmetric* compact formulation of the perfect matching polytope, that is, no compact formulation which is invariant under permuting the nodes of the underlying graph. Barahona [2] showed that the minimum perfect matching problem can be reduced to a sequence of $\mathcal{O}(m^2 \log n)$ minimum mean cycle problems, where n and m denote the number of nodes and edges of G. He also showed that the latter problem can be formulated as a compact linear program. This implies that the minimum perfect matching problem can be reduced to a polynomial sequence of compact linear programs. For the construction of this sequence it is, however, important that all intermediate optimal solutions are vertex solutions. Here, a feasible solution x of a linear program is called a *vertex solution* if x is a vertex of the polyhedron determined by the system of equations and inequalities of this linear program. In this extended abstract we show that the minimum T-cut problem, which is related to the perfect matching polytope in terms of separation, can be reduced to a sequence of only two compact linear programs, but also in our approach, the construction of the second linear program requires an optimal vertex solution of the first linear program.

We denote the node and edge set of a graph G by $V(G)$ and $E(G)$, respectively. A *cut* of G is a subset C of edges such that $C = \delta(U)$, where for any $\varnothing \neq U \subsetneq V(G)$, $\delta(U) := \{\{u, v\} \in E(G) | u \in U, v \in V(G) \setminus U\}$. For any nodes $s, t \in V(G)$, $\delta(U)$

is said to be an s,t-*cut* if $s \in U$, $t \in V(G) \setminus U$. Let T be a subset of the nodes of G of even size. A T-*cut* is a cut $\delta(U)$ of G such that $|T \cap U|$ is odd. Given capacities on the edges $c : E(G) \to \mathbb{R}_+$, in the *minimum T-cut problem* one wants to find a T-cut $\delta(U)$ of G minimizing $c(\delta(U))$, where for any $F \subseteq E(G)$, $c(F) := \sum_{e \in F} c_e$. This problem can, for instance, be solved with the famous algorithm of Padberg-Rao [4] which will be the basis for our LP-approach.

Let K be the complete graph on $V(G)$ and let $H \subseteq K$ be a spanning tree on $V(G)$. For any edge $f = \{s,t\} \in E(H)$, the cut C_f in G determined by the two components of $H - f$ is called the *fundamental cut induced by* f. A *Gomory-Hu-tree* for (G,c) is a spanning tree H on $V(G)$ such that every fundamental cut $C_{\{s,t\}}$ is a minimum s,t-cut in G. The algorithm of Padberg and Rao computes a Gomory-Hu-tree H for (G,c) and selects among the fundamental cuts C_f, $f \in E(H)$, that are T-cuts a T-cut minimizing $c(C_f)$. This T-cut can be shown to be an optimal solution of the minimum T-cut problem.

The following linear program combines a compact formulation for the spanning tree polytope defined on K with minimum s,t-cut formulations defined on G. The *spanning tree polytope* $P_{Tree}(K)$ is the convex hull of the characteristic vectors $\chi^{E(H)} \in \{0,1\}^{E(K)}$ of spanning trees $H \subseteq K$, where $\chi^F_e = 1$ if and only if $e \in F$. For each edge $f \in E(K)$, introduce a variable λ_f, and for each edge $f \in E(K)$, each node $u \in f$, and each node $v \in V(K)$, introduce a variable $\mu_{f,u,v}$. Moreover, for each $f \in E(K)$ and each $\{u,v\} \in E(G)$, introduce a variable $x^f_{\{u,v\}}$. Finally, for each edge $f \in E(K)$, fix a node $t_f \in f$. Consider the linear program

$$\begin{aligned}
&\min \textstyle\sum_{f \in E(K)} \sum_{e \in E(G)} c(e) x^f_e \\
&\text{s.t. } x^f_{\{u,v\}} + \mu_{f,t_f,u} - \mu_{f,t_f,v} \geq 0, \qquad (1.1)\\
&\qquad x^f_{\{u,v\}} + \mu_{f,t_f,v} - \mu_{f,t_f,u} \geq 0 \ \text{ for all } f \in E(K), \{u,v\} \in E(G),
\end{aligned}$$

$$\begin{aligned}
&\text{and } \lambda(E(K)) = |V(K)| - 1,\\
&\qquad \lambda \geq 0,\ \mu \geq 0\\
&\qquad \mu_{\{u,v\},u,v} = \mu_{\{u,v\},v,u} = 0 \qquad \text{for all } \{u,v\} \in E(K), \qquad (1.2)\\
&\qquad \lambda_e - \textstyle\sum_{v \in e} \mu_{e,v,w} = 0 \qquad \text{for all } e \in E(K),\ w \in V(K),\\
&\qquad \textstyle\sum_{w \in V(K) \setminus \{u\}} \mu_{\{u,w\},w,v} = 1 \qquad \text{for all } \{u,v\} \in E(K).
\end{aligned}$$

Theorem 28 *Any feasible solution $(\{\bar{x}^f\}_{f \in E(K)}, \bar{\lambda}, \bar{\mu})$ of (1.1)-(1.2) is an optimal vertex solution of (1.1)-(1.2) if and only if $\bar{\lambda}$ is the characteristic vector of a Gomory-Hu tree H for (G,c), for each $f \in E(H)$, the vector $(\bar{\mu}_{f,t_f,v})_{v \in V(K)}$ is the characteristic vector of the component of $H - f$ containing t_f, $\bar{x}^f$ is the characteristic vector of the fundamental cut in G induced by f, and for each $f \in E(K) \setminus E(H)$, $\mu_{f,u,v} = \bar{x}^f_v = 0$ for all $u \in f$, $v \in V(G) = V(K)$.*

We sketch the proof. Denote by P the polytope determined by the feasible so-

lutions of (1.1)-(1.2). Any $(\{x^f\}_{f\in E(K)}, \lambda, \mu) \in P$ is a vertex of P if and only if each component $x^f_{\{u,v\}}$ is chosen to be minimal, that is, $x^f_{\{u,v\}} = \max\{\mu_{f,t_f,v} - \mu_{f,t_f,u}, \mu_{f,t_f,u} - \mu_{f,t_f,v}\}$ and (λ, μ) is a vertex of the polytope Q constituted by (1.2). Q is an integer polytope [3] and a compact formulation for $P_{Tree}(K)$. One now verifies that $(\{x^f\}_{f\in E(K)}, \lambda, \mu)$ is vertex if and only if it satisfies the conditions mentioned in Theorem 28. Moreover, observe that a spanning tree H of K is a Gomory-Hu tree if and only if it minimizes $\sum_{f\in E(H)} c(C_f)$. For details, the interested reader is referred to [5].

Once we have computed a Gomory-Hu tree H using (1.1)-(1.2), we have first to identify the T-cuts among the fundamental cuts and then to select a T-cut C_f minimizing $c(C_f)$. Due to lack of space we skip the first part and instead of we denote by $F \subseteq E(H)$ the set of all edges $f \in E(H)$ that correspond to T-cuts. Indeed, given H, the characteristic vector of F can be easily determined by linear programming using path-flow formulations. Consider the following disjunctive programming approach [1] for finding a minimum T-cut:

$$
\begin{aligned}
\min \quad & \sum_{f\in E(K)} \sum_{e\in E(G)} c(e) y^f_e \\
\text{s.t.} \quad & (\{x^f\}_{f\in E(K)}, \lambda, \mu) \text{ is a vertex solution of (1.1)}, \\
& \nu_f = 1 \; \forall f \in F, && (1.3)\\
& \nu_f = 0 \; \forall f \in E(K) \setminus F, && (1.4)\\
& \sum_{f\in E(K)} \vartheta_f = 1, && (1.5)\\
& 0 \le \vartheta \le \nu, && (1.6)\\
& y^f_e \ge x^f_e + \vartheta_f - 1 \; \forall e \in E(G), f \in E(K). && (1.7)\\
& y^f_e \ge 0 \; \forall e \in E(G), f \in E(K). && (1.8)
\end{aligned}
$$

For any vertex solution $(\{x^f, y^f\}_{f\in E(K)}, \lambda, \mu, \nu, \vartheta)$, inequalities (1.3)-(1.6) imply that ϑ is a unit vector with $\vartheta_g = 1$ for some $g \in F$. For the same reason, $(\{x^f, y^f\}_{f\in E(K)}, \lambda, \mu, \nu, \vartheta)$ satisfies at least one of the two inequalities in (1.7), (1.8) at equality for each pair of edges $e \in E(G), f \in E(K)$. Hence, $y^f = 0$ for all $f \in E(K) \setminus \{g\}$, while y^g is the characteristic vector of a T-cut.

References

[1] E. BALAS, *Disjunctive programming*, Ann. Discrete Math. **5** (1979), pp. 3–51.

[2] F. BARAHONA, *Reducing matching to polynomial size linear programming.*, SIAM J. Optim. **3**, no. 4 (1993), pp. 688–695.

[3] R. MARTIN, *Using separation algorithms to generate mixed integer model reformulations*, Oper. Res. Lett. **10**, no. 3 (1991), pp. 119–128.

[4] M. W. PADBERG AND M. RAO, *Odd minimum cut-sets and b-matchings.*, Math. Oper. Res. **7** (1982), pp. 67–80.

[5] R. STEPHAN, *An extension of disjunctive programming and its impact to compact tree formulations*, Université catholique de Louvain, Core Discussion paper 2010/45, 2010.
[6] M. YANNAKAKIS, *Expressing combinatorial optimization problems by linear programs*, J. Comput. Syst. Sci. **43**, no. 3 (1991), pp. 441–466.

The dynamics of deterministic systems from a hypergraph theoretical point of view

Luis M. Torres, [a] Annegret K. Wagler [b]

[a] *Escuela Politécnica Nacional (Departamento de Matemática), Quito, Ecuador,*
luis.torres@epn.edu.ec

[b] *Université Blaise Pascal (Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes) and CNRS, Clermont-Ferrand, France,*
wagler@isima.fr

Key words: Petri nets, concurrent deterministic dynamic systems, hypergraphs

---

Petri nets constitute a well-established framework for modeling complex dynamic systems, see e.g. [1,3,6] for their broad application range. The structure of these systems is described by means of a network, while the studied dynamic processes are usually represented in terms of state changes.

The *network* is a weighted directed bipartite graph $G = (P \cup T, A, w)$ with two kinds of nodes, places and transitions, linked by weighted directed arcs. The places represent the system's components, while the transitions stand for their possible interactions.

Some places $B \subseteq P$ have bounded capacities, given by a vector $u \in \mathbb{Z}_+^B$. A *state* of the system is an assignment of *tokens* to places taking into account the capacities. Hence, any state can be represented as a vector $x \in \mathbb{Z}_+^P$ and the *potential state space* of the system is $\mathcal{X} := \left\{x \in \mathbb{Z}_+^P : x_i \leq u_i,\ \forall i \in B\right\}$.

Dynamic processes are described as sequences $x^1, \ldots, x^k$ of system states, where x^{i+1} is obtained from x^i by *switching* a transition $t \in T$. Thereby, t consumes w_{it} tokens from each pre-place $i \in P^-(t) := \{i \in P : (i,t) \in A\}$ and produces w_{ti} new tokens on each post-place $i \in P^+(t) := \{i \in P : (t,i) \in A\}$. A transition $t \in T$ is *enabled* at a state $x \in \mathcal{X}$ if switching t yields a valid successor state. Thus, the set of enabled transitions at state x is

$$T(x) := \left\{t \in T : x_i - w_{it} \geq 0,\ \forall i \in P^-(t);\ \ x_i + w_{ti} \leq u_i,\ \forall i \in P^+(t) \cap B\right\}.$$

A *Petri net* (G, x^0) is given by a network G and an initial state x^0, its state space is the set of all states reachable from x^0 by switching sequences of transitions. If $T(x)$ contains more than one transition for some state x, a decision between these alternatives is taken non-deterministically, leading to a branching system behavior which cannot model deterministic systems.

A dynamic system is *deterministic* if any state $x \in \mathcal{X}$ has a unique successor state $\mathrm{succ}(x)$. Hence, if there is a state x in such a system where more than one transition is enabled, a deterministic decision is taken in order to select a unique transition $\mathsf{trans}(x) \in T(x)$ that must be switched in order to reach $\mathrm{succ}(x)$.

An explicit encoding of $\mathsf{trans}(x)$, for instance state-wise, is at least exponential in the size of G and u. In [9] we propose a compact encoding, using orientations of the following conflict graph. The *transition conflict graph* of G is an undirected graph $\mathbb{K} = (T, \mathbb{E})$ having as nodes the transitions from G, where two transitions t, t' are joined by an edge if and only if there exists at least one state where both are enabled, i.e., $tt' \in \mathbb{E} \Leftrightarrow X(t) \cap X(t') \neq \emptyset$, where

$$X(t) := \left\{ x \in \mathcal{X} : x_i \geq w_{it},\, \forall i \in P^-(t);\;\; x_i \leq u_i - w_{ti},\, \forall i \in P^+(t) \cap B \right\}$$

denotes the set of states at which transition t is enabled. $\mathbb{K}$ can be constructed in $O(|P|\ |T|^2)$ time checking for box intersections or in a more efficient way suggested in [9]. It clearly follows that, for every state $x \in \mathcal{X}$, the set $T(x)$ induces a clique in $\mathbb{K}$, i.e., a set of mutually adjacent nodes.

A directed graph $\mathbb{D}$ obtained by orienting the edges of $\mathbb{K}$ is said to be a *valid orientation* if, for every state $x \in \mathcal{X}$ with $T(x) \neq \emptyset$, the clique induced by the nodes from $T(x)$ has a unique sink, and this sink coincides with $\mathsf{trans}(x)$. The arcs of $\mathbb{D}$ can be interpreted as priority relations between pairs of transitions, with $\mathsf{trans}(x)$ being the transition with *highest priority* in $T(x)$ [7]. Observe that the size of $\mathbb{D}$ is $O(|T|^2)$, which is polynomial in the size of G.

In [9], it is shown that for any undirected graph H, there is a network G having H as its transition conflict graph. Thus, in general, neither $\mathbb{K}$ nor $\mathbb{D}$ admit particular properties from a graph-theoretical point of view, which turns characterizing valid orientations into a difficult task. The aim of this note is to provide some insights into the structural properties of the transition conflict graph and its valid orientations in the context of hypergraph theory.

1 Hypergraphs related to the transition conflict graph

Let $G = (P \cup T, A, w)$ be a network and $\mathbb{K} = (T, \mathbb{E})$ its transition conflict graph. By definition, for every state $x \in \mathcal{X}$, the set $T(x)$ of enabled transitions induces a clique in $\mathbb{K}$. The converse is not necessarily true, but we show that at least the inclusion-wise maximal cliques in $\mathbb{K}$ are associated with states of the system.

The equivalence relation on $\mathcal{X}$, where $x \sim x'$ iff $T(x) = T(x')$, partitions the state space into $r \leq 2^{|T|}$ equivalence classes $\mathcal{X}_1, \ldots, \mathcal{X}_r$ of states that share the same sets of enabled transitions. Let $\tilde{x}^i \in \mathcal{X}_i$ with $1 \leq i \leq r$ be representative elements and define the *transition hypergraph* $\mathcal{H}_T := (T, \mathcal{E}_T)$ *of* G on the set T of transitions, whose family $\mathcal{E}_T$ of hyperedges is given by $\mathcal{E}_T := \{T(\tilde{x}^i) : 1 \leq i \leq r\}$.

The *state hypergraph of* G is the dual hypergraph $\mathcal{H}_{\tilde{\mathcal{X}}}$ of $\mathcal{H}_T$ and has $\tilde{\mathcal{X}} := \{\tilde{x}^1, \ldots, \tilde{x}^r\}$ as node set, where its family of hyperedges is determined by $\mathcal{E}_{\tilde{\mathcal{X}}} := \left\{ X(t) \cap \tilde{\mathcal{X}} : t \in T \right\}$.

A hypergraph $\mathcal{H} := (V, \mathcal{E})$ has the *Helly property* if, for any family $\mathcal{E}' \subseteq \mathcal{E}$ of pairwise intersecting hyperedges, there exists a node $v \in V$ contained in all hyperedges from $\mathcal{E}'$. This is the case if and only if the inclusion-wise maximal hyperedges of its dual hypergraph $\mathcal{H}^*$ are precisely the inclusion-wise maximal cliques of the *intersection graph* $G(\mathcal{H})$ of $\mathcal{H}$ [2]. Observe that in our case the transition conflict graph $\mathbb{K}$ is exactly $G(\mathcal{H}_{\tilde{\mathcal{X}}})$. We have shown the following:

Lemma 1.1 *$\mathcal{H}_{\tilde{\mathcal{X}}}$ satisfies the Helly property.*

Hence, as a direct consequence, we obtain:

Theorem 1.2 *The inclusion-wise maximal hyperedges from $\mathcal{E}_T$ are exactly the inclusion-wise maximal cliques of $\mathbb{K}$. Thus, for every inclusion-wise maximal clique Q there is some state $\tilde{x}^i \in \tilde{\mathcal{X}}$, $1 \leq i \leq r$, satisfying $T(\tilde{x}^i) = Q$.*

In general, $\mathbb{K}$ does not admit a special structure from a graph-theoretical point of view [9]. We show, however, that $\mathbb{K}$ has some interesting properties, provided that $\mathcal{H}_T$ and $\mathcal{H}_{\tilde{\mathcal{X}}}$ belong to certain classes of hypergraphs.

A *cycle* of length k in a hypergraph $\mathcal{H} = (V, \mathcal{E})$ is a sequence

$$(v_1, E_1, v_2, E_2, \ldots, v_k, E_k, v_{k+1})$$

such that $v_i \in V$ for all $1 \leq i \leq k+1$, $E_i \in \mathcal{E}$ for all $1 \leq i \leq k$, $E_i \neq E_j$ if $i \neq j$, $v_i, v_{i+1} \in E_i$ for all $1 \leq i \leq k$, and $v_{k+1} = v_1$. A hypergraph $\mathcal{H}^* = (V^*, \mathcal{E}^*)$ is called *arboreal* if there exists a tree T^* on the node set V^* such that every hyperedge of $\mathcal{H}^*$ induces a subtree in T^*. Due to structural characterizations in [4,5,8], arboreal hypergraphs are dual to acyclic hypergraphs and their intersection graphs are *chordal*, i.e. each cycle having four or more nodes contains an edge joining two nodes that are not adjacent in the cycle. Moreover, it can be shown that arboreal hypergraphs have the Helly property.

Combining the previous observations, we obtain the following result.

Theorem 1.3 *The following assertions are equivalent:*

- *$\mathcal{H}_T$ is acyclic.*
- *$\mathcal{H}_{\tilde{\mathcal{X}}}$ is arboreal.*
- *$\mathbb{K}$ is chordal.*

2 Valid orientations

A directed graph $\mathbb{D} = (T, \mathbb{A})$ obtained by orienting the edges of $\mathbb{K}$ is said to be *valid* if, for every state $x \in \mathcal{X}$ with $T(x) \neq \emptyset$, the subgraph of $\mathbb{K}$ induced by the nodes from $T(x)$ has a unique sink, and this sink coincides with $\mathsf{trans}(x)$.

Consider the equivalence relation between networks, where $G \sim_{\mathbb{K}} G'$ iff their conflict graphs are isomorphic. An orientation is *strongly valid* if it is valid for all networks of some equivalence class of $\sim_{\mathbb{K}}$.

By [9], an orientation is strongly valid if and only if it does not contain a directed cycle of length three. Using this, we can exclude the occurrence of directed cycles for systems with an acyclic transition hypergraph:

Theorem 2.1 *The transition hypergraph $\mathcal{H}_T$ of a network G is acyclic if and only if, for any orientation $\mathbb{D}$ of $\mathbb{K}$, the following two statements are equivalent:*

(i) $\mathbb{D}$ *is strongly valid.*

(ii) $\mathbb{D}$ *is acyclic.*

In some applications, for instance in systems biology, the set of rules governing the dynamic behavior of the system is not known a priory, but has to be reconstructed from information provided by experimental observations. For that, one could search for minimal sets of edges in $\mathbb{K}$ with the property that knowledge about their orientation allows to infer the orientation of the remaining edges [10]. Such kind of issues are specially relevant in practice, as they can be used to assist the design of experiments. We expect this problem to be easier in the special case of acyclic transition hypergraphs by benefiting from structural properties of the chordal transition conflict graphs.

References

[1] Adam, N.R., Atluri, V., Huang, W.K.: Modeling and analysis of workflows using Petri nets. J. Intell. Inf. Syst. **10**(2), 131–158 (1998).

[2] Berge, C., Duchet, P.: A generalisation of Gilmore's theorem. In: M. Fiedler (ed.) Recent Advances in Graph Theory, pp. 49–55. Acad. Praha, Prague (1975)

[3] Chaouiya, C., Remy, E., Thieffry, D.: Petri net modelling of biological regulatory networks. J. of Discrete Algorithms **6**(2), 165–177 (2008).

[4] Duchet, P.: Propriété de Helly et problèmes de représentations. In: Problèmes Combinatoires et Théorie des Graphes, Coll. Orsay 1976, pp. 117–118. CNRS Paris (1978)

[5] Flament, C.: Hypergraphes arborés. Discrete Math. **21**, 223–226 (1978)

[6] Gu, T., Bahri, P.A.: A survey of Petri net applications in batch processes. Comput. Ind. **47**(1), 99–111 (2002).

[7] Marwan, W., Wagler, A., Weismantel, R.: A mathematical approach to solve the network reconstruction problem. Math. Methods of Operations Research **67**, 117–132 (2008)

[8] Slater, P.J.: A characterization of soft hypergraphs. Canad. Math. Bull. **21**, 335–337 (1978)

[9] Torres, L.M., Wagler, A.: Encoding the dynamics of deterministic systems, Math. Methods of Operations Research, DOI http://dx.doi.org/10.1007/s00186-011-0353-6 .

[10] Torres, L.M., Wagler, A.: Model reconstruction for discrete deterministic systems, Elec. Notes Disc. Math. **36**, 175–182 (2010)

An Evolutionary Algorithm for the Multiobjective Risk-Equity Constrained Routing Problem

Nora Touati-Moungla, Dimo Brockhoff,

Laboratoire d'Informatique, École Polytechnique,
91128 Palaiseau Cedex, France

Key words: Evolutionary multiobjective optimization, Transportation of hazardous materials, Risk equity, Shortest Paths.

1 Introduction

The transportation of hazardous materials (*hazmat* from now on) has received a large interest in recent years, which results from the increase in public awareness of the dangers of hazmats and the enormous amount of hazmats being transported [4]. The main target of this problem is to select routes from a given origin-destination pair of nodes such that the risk for the surrounding population and the environment is minimized—without producing excessive economic costs. When solving such a problem by minimizing both cost and the total risk, typically several vehicles share the same (short) routes which results in high risks associated to regions surrounding these paths whereas other regions are not affected. In this case, one may wish to distribute the risk in an equitable way over the population and the environment. Several studies consider this additional minimization of the equity risk, but most of them consist of a *single* origin-destination hazmat routing for a specific hazmat, transport mode and vehicle type (see for example [1,4]). A more realistic *multi*-commodity flow model was proposed in [4] where each commodity is considered as one hazmat type. The objective function is formulated as the sum of the economical cost and the cost related to the consequences of an incident for each material. To deal with risk equity, the costs are defined as functions of the flow traversing the arcs which imposes an increase of the arc's cost and risk when the number of vehicles transporting a given material increases on the arc.

The majority of all hazmat routing studies deal with a single-objective scenario although the problem itself is multiobjective in nature and it is important to study the trade-offs among the objectives. Evolutionary Multiobjective Optimization (EMO) algorithms are able to compute a set of solutions showing these trade-offs within a single algorithm run which is the reason why we propose to use them for the problem of hazmat routing in this study (Sec. 3). Before, we formalize the routing of hazmat problem with three objectives (minimize total routing cost, total routing

risk *and* risk equity) as a multicommodity flow model as in [4] since this model is the most realistic one permitting to manage several hazmat types simultaneously (Sec. 2).

2 The multiobjective risk-equity constrained routing problem

Let the transportation network be represented as a directed graph $G = (N, A)$, with N being the set of nodes and A the set of arcs. Let C be the set of commodities, given as a set of point-to-point demands to transport a certain amount of hazmats. For any commodity $c \in C$, let s^c and t^c be the source node and the destination node respectively, and let V^c be the number of available trucks for the transportation of commodity c. Each commodity is associated with a unique type of hazmat. We assume that the risk is computed on each arc of the network and is proportional to the number of trucks traversing such an arc. We consider a set Q of regions, and we define r_{ij}^{cq} as the risk imposed on region $q \in Q$ when the arc $(i, j) \in A$ is used by a truck for the transportation of hazmat of type c. We remark that we employ a notion of *spread risk*, in that an accidental event on arc (i, j) within region $q \in Q$ may strongly affect another region $q' \in Q$. With each arc $(i, j) \in A$ a cost c_{ij}^c is associated, involved by the travel of a truck of commodity c on this arc.

The problem of transporting hazmat is multiobjective in nature: one usually wants to minimize the *total cost* of transportation, the *total risk* of transportation imposed on all regions and the *distributed risk*, which can be defined as a measure of risk that is shared among different regions. More specifically, for a given solution, each region $q \in Q$ will be affected by a risk ω_q which depends on the transportation patterns in all other regions. The third objective will then be $\max_{q \in Q} \omega_q$, and has to be minimized.

We introduce the integer variable y_{ij}^c for the number of trucks that use arc (i, j) for transporting commodity c. We assume a fixed number of trucks and that all trucks have the same load. The proposed model is defined as follows:

$$\min \quad \sum_{c \in C} \sum_{(i,j) \in A} c_{ij}^c y_{ij}^c \tag{2.1}$$

$$\min \quad \sum_{c \in C} \sum_{q \in Q} \sum_{(i,j) \in A} r_{ij}^{cq} y_{ij}^c \tag{2.2}$$

$$\min \quad \max_{q \in Q} \Big\{ \sum_{c \in C} \sum_{(i,j) \in A} r_{ij}^{cq} y_{ij}^c \Big\} \tag{2.3}$$

$$s.t. \sum_{j \in \delta^+(i)} y_{ij}^c - \sum_{j \in \delta^-(i)} y_{ji}^c = q_i^c \quad \forall i \in N, c \in C \tag{2.4}$$

$$y_{ij}^c \in \{0, 1, 2, \ldots\} \qquad \forall (i, j) \in A, c \in C. \tag{2.5}$$

The first objective in (2.1) is a cost function and is to be minimized. The second objective, given by (2.2), minimizes the total risk on all regions and objective (2.3) minimizes the maximum risk imposed on all regions. Constraints

(2.4) are conservation constraints, where $\delta^-(i) = \{j \in N : (j,i) \in A\}$ and $\delta^+(i) = \{j \in N : (i,j) \in A\}$ are the direct successors and predecessors of node i, and $q_i^c = V^c$ if $i = s^c$, $q_i^c = -V^c$ if $i = t^c$ and $q_i^c = 0$ otherwise.

3 Evolutionary multiobjective optimization

Evolutionary algorithms (EAs) and Evolutionary Multiobjective Optimization (EMO) algorithms in particular are general-purpose randomized search heuristics and as such well suited for problems where the objective function(s) can be highly non-linear, noisy, or even given only implicitly, e.g., by expensive simulations [6,5]. Since the third objective in the above problem formulation is nonlinear, we propose to use an EMO algorithm for the multiobjective risk-equity constrained routing problem here. Our EMO algorithm follows the standard iterative/generational cycle of an EA of mating selection, variation, objective function evaluation, and environmental selection and is build upon the state-of-the-art selection scheme in HypE [2] as implemented in the PISA platform [3]. The variation operators as well as the representation of the solutions, however, have to be adapted to the problem at hand in the following way in order to fulfill the problem's constraints at all times.

Representation: We choose a variable length representation as it has been theoretically shown to be a good choice for multiobjective shortest paths problems [7]: A solution is thereby represented by a list of paths of variable lengths with one path per truck. For the moment, we consider a fixed amount of trucks for each commodity and therefore a fixed number of paths through the network. In order to have every variable length path represent an uninterrupted path from source to destination at any time (see the constraints in (2.4)), we ensure all paths to always start with the source s^c for the corresponding commodity c, ensure with the variation operator that all neighbored vertices in the path are connected by an arc, and complete each path by the shortest path between the path's actual end node and the commodity's destination node t^c.

Initialization: Initially, we start with a single solution where the paths p for all trucks are empty ($p = (s^c)$). This corresponds to the situation where all trucks choose the shortest s^c–t^c path for their assigned commodity—implying the smallest possible overall cost but a high risk along the used route(s). Nevertheless, the initial solution is already Pareto-optimal and is expected to be a good starting point for the algorithm.

Variation: As mutation operator, we suggest to shorten or lengthen the path of one or several trucks. In order to generate a new solution s' from s, for each truck path, we draw a binary value $b \in \{0,1\}$ uniformly at random and create the new path p' from the old one $p = (v_1 = s^c, v_2, \ldots, v_l)$ as in [7]:

- if $b = 0$ and $l =:$ length$(p) \geq 2$, set $p' = (s^c, \ldots, v_{l-1})$
- if $b = 1$ and $|V_{\text{rem}} = \{v \in V \mid (v_l, v) \in A\}| \neq \emptyset$, choose v_{l+1} from V_{rem} uniformly at random and set $p' = (v_0, \ldots, v_l, v_{l+1})$.
- otherwise, use the same path p also in the new solution s'.

4 Conclusions

The transportation of hazmats is an important optimization problem in the field of sustainable development and in particular the equitable distribution of risks is of high interest. Within this study, we formalize this transportation problem as the minimization of three objectives and propose to use an evolutionary algorithm to cope with the non-linear equity risk objective.

The third objective function of our problem can be rewritten by minimizing the additional variable z as third objective and adding the constraints $\forall q \in Q : z \geq \sum_{c \in C} \sum_{(i,j) \in A} r_{ij}^{cq} f_{ij}^{c}$. Although this equivalent formulation makes the problem linear (with additional linear constraints), classical algorithms are expected to have difficulties with this formulation as well and our algorithm is supposed to be more efficient in the current formulation due to the fewer number of constraints. Note that, for the moment, the proposed EMO algorithm exists on paper only and an actual implementation has to prove in the future which additional algorithm components (such as problem-specific initialization, recombination operators, or other exact optimization (sub-)procedures) are necessary to generate solutions of sufficient quality and whether adaptively changing the number and capacity of trucks is beneficial.

References

[1] V. Akgun, E. Erkut and R. Batta, On finding dissimilar paths, European Journal of Operational Research 121(2):232-246, 2000.

[2] J. Bader and E. Zitzler. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76, 2011.

[3] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA—a platform and programming language independent interface for search algorithms. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, pages 494–508, 2003. Springer.

[4] M. Caramia and P. Dell'Olmo, Multiobjective management in freight logistics: increasing capacity, service level and safety with optimization algorithms, Springer London Ltd, 2008.

[5] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007.

[6] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK, 2001.

[7] C. Horoba. Analysis of a simple evolutionary algorithm for the multiobjective shortest path problem. In *Foundations of Genetic Algorithms (FOGA 2009)*, pages 113–120. ACM, 2009.

Popular Ranking

Anke van Zuylen, [a] Frans Schalekamp, [b] David P. Williamson [c]

[a] *Max-Planck-Institut für Informatik, Saarbrücken, Germany,*
anke@mpi-inf.mpg.de

[b] *Independent*

[c] *School of ORIE, Cornell University, Ithaca, NY, USA.*

Key words: Rank aggregation, Kemeny Rank Aggregation, Popular Ranking

1 Introduction

How do you aggregate the preferences of multiple agents in a fair manner? This is a question that has occupied researchers for several centuries. Suppose we have k voters who each give their preferences on n candidates. How should the candidates be ranked to best represent the input? Marquis de Condorcet [5] showed that there may not exist a "winner": a candidate who beats all other candidates in a pairwise majority vote. Borda [4] and Condorcet [5] (and many others after them) proposed different ways of aggregating the preferences of the voters, and argued over which method is the right one. Only in the middle of the 20th century, Arrow [2] showed that there is no right method: there exists no aggregation method that simulataneously satisfies three natural criteria (non-dictatorship, independence of irrelevant alternatives and Pareto efficiency).

This negative result notwithstanding, we still want to find aggregate rankings based on voters' inputs. In this paper, we consider the case when, rather than selecting a winner, we would like to find a permutation of the candidates that represents the voters' inputs. Each voter's input is assumed to be a permutation of the candidates, where a candidate is ranked above another candidate, if the voter prefers the former to the latter candidate. The goal is to find a permutation that minimizes the sum of the distances to the voters' permutations, where in principle any distance(-like) function on permutations can be used, e.g. Kendall distance or Footrule distance.

Young & Levenglick [9] show that the Kendall distance is the unique distance function such that the permutation(s) that minimize it have three desirable properties of being neutral, consistent and Condorcet. The latter property means that, if there exists a permutation such that the order of every pair of elements is the order preferred by a majority, then this permutation has minimum distance to the voters' permutations. This distance was already proposed by Kemeny [6] for other reasons ([6] defines axioms on the *distance function*, and finds that the Kendall distance adheres

to the axioms), and the problem of finding an optimal ranking with respect to this criterion is now known as Kemeny Rank Aggregation.
In this paper, we suggest a new way of thinking about this problem. Suppose instead of minimizing the total distance from the voters' inputs, we want to find a permutation that makes a majority of the voters "happy"? Of course, a voter is happy when we follow her opinion exactly, and we cannot do this simultaneously for a majority of the voters, unless a majority of the voters is in total agreement. Therefore, our goal is to find a permutation such that there exists no other permutation that a majority of the voters prefer, in the sense that their distance to the alternative permutation is smaller. We call such a permutation a *popular ranking*.
Unfortunately, we show that such a permutation is unlikely to exist: it only exists if Condorcet's paradox does not occur. Even worse than this, we show that if Condorcet's paradox does not occur, then it may still be the case that no popular ranking exists. The only positive news in this context is, perhaps paradoxically, an NP-hardness result: we show that if Condorcet's paradox does not occur, then we can efficiently compute a permutation, which may or may not be popular, but for which the voters will have to solve an NP-hard problem to compute a permutation that a majority of them prefer.

Related Work: Our work is inspired by Abraham *et al*. [1] where the notion of *popular matchings* is introduced. Popular ranking is also related to the problem of designing a voting mechanism in which the voters do not have an incentive to lie about their preferences. However, rather than considering deviations of a single voter, a popular solution is robust against deviations of a majority of the voters. We show that, if the input does not contain Condorcet's paradox, then there is a solution that may or may not be popular, but for which it is computationally hard for a majority of the voters to manipulate the output to their advantage. This result has a similar flavor as a result by Bartholdi *et al*. [3], who demonstrate a voting rule for deciding the "winner" of an election, for which it is computationally hard for a single voter to manipulate the output.

2 Popular Ranking

We are given a set of alternatives $[n]$ (where the notation $[n]$ means $\{1, 2, \ldots, n\}$) and a set of voters $[k]$, where each voter ℓ has a complete ordering of the alternatives. We will denote these complete orderings of a voter ℓ as a list of the alternatives, where an alternative earlier in the list is preferred to elements that succeed it, and use the notation $\pi_\ell^{-1} : [n] \to [n]$ (the use of "$^{-1}$" will become clear shortly), where $\pi_\ell^{-1}(i)$ is the alternative at position i in the ordering of voter ℓ. Note that we can interpret π_ℓ^{-1} as a permutation. Further, the inverse of π_ℓ^{-1}, which we will denote by π_ℓ, is well defined and can be interpretated as the position of the alternatives in the list of voter ℓ. We will use $\mathrm{list}(\pi_\ell)$ to denote the ordered sequence $(\pi_\ell^{-1}(1), \pi_\ell^{-1}(2), \ldots, \pi^{-1}(n))$.
The Kendall distance between two permutations π, σ, denoted by $K(\pi, \sigma)$, is defined as the number of pairwise disagreements of π and σ, i.e. $K(\pi, \sigma) = \#\{i, j :$

$\pi(i) < \pi(j)$ and $\sigma(i) > \sigma(j)\} + \#\{i, j : \pi(i) > \pi(j)$ and $\sigma(i) < \sigma(j)\}$.

Definition 2.1 *We say a permutation π is* popular, *if $\nexists \pi'$ such that $K(\pi_\ell, \pi') < K(\pi_\ell, \pi)$ for a strict majority of the voters $\ell \in [k]$.*

We define the *majority graph* $G = (V, A)$ for an instance as the directed graph which has a vertex for every $i \in [n]$ and an arc (i, j) if a majority of the voters $\ell \in [k]$ has $\pi_\ell(i) < \pi_\ell(j)$. Condorcet observed that such a graph may have a cycle; this is known as "Concorcet's paradox".

Lemma 2.1 *No popular ranking exists if the majority graph has a directed cycle.*

Proof (sketch) (sketch) If we order the elements from left to right according to a ranking π, then there must be some arc (i, j) in the graph that is a *back arc*, i.e. for which $\pi(j) < \pi(i)$. Let π' be the permutation we obtain by swapping i and j, i.e. $\pi'(i) = \pi(j), \pi'(j) = \pi(i)$ and $\pi'(t) = \pi(t)$ for all $t \neq i, j$. Then one can show that a strict majority of the voters prefer π' to π, namely the voters ℓ who have $\pi_\ell(i) < \pi_\ell(j)$. ∎

If the majority graph is acyclic, then a popular ranking could exist. We consider the case when the majority graph is a tournament, i.e. for every i, j exactly one of the arcs (i, j) and (j, i) is in G. Note that the majority graph is always a tournament if the number of voters is odd. By Lemma 2.1, the only permutation that *could* be popular is the permutation we obtain by topologically sorting the majority tournament. However, it is not the case that this ranking is always a popular ranking, as we show in the full version of this paper [8].Even though the topologically sort of the majority tournament is not necessarily a popular ranking, it turns out that it is a "good" permutation in the sense that it is NP-hard to find a ranking that a majority of the voters prefer. The proof of the following theorem is given in the full version [8].

Theorem 2.2 *Given an input to the popular rank aggregation problem with an acyclic majority graph, it is NP-hard to find a ranking ρ that a majority of the voters S prefers to a topological sort of the majority graph, even if S is given.*

3 Directions

We have seen that a popular ranking does not always exist, even if the majority graph has no cycles. Perhaps popularity is asking for too much and we should relax our objective. It is an interesting question whether there exists a suitable relaxation of the notion of popularity, so that one can get positive results. One way of relaxing the notion is looking for rankings with *least-unpopularity-factor* (McCutchen [7] introduced this notion for matchings). The bad news is that it can be shown that the unpopularity factor of the permutation π we obtain by topologically sorting the majority tournament may be unbounded. It is an open question however whether there exists a permutation with bounded unpopularity (and if so, what this uniform bound is) and whether such a permutation can be found in polynomial time.

Acknowledgements
The first author thanks Chien-Chung Huang for helpful discussions.

References

[1] D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM J. Comput.*, 37(4):1030–1045, 2007.

[2] K. J. Arrow. *Social choice and individual values*. Yale University Press, 1951.

[3] J. Bartholdi, C. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6:227–241, 1989.

[4] J. Borda. *Memoire sur les elections au scrutin*. Histoire de l'Academie Royal des Sciences, 1781.

[5] M. Condorcet. *Sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. L'Imprimerie Royale, Paris, 1785.

[6] J. Kemeny. Mathematics without numbers. *Daedalus*, 88:575–591, 1959.

[7] R. McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *LATIN'08*, pages 593–604, Búzios, Brazil, 2008.

[8] A. van Zuylen, F. Schalekamp, and D. P. Williamson. Popular ranking. Full version available at `http://www.mpi-inf.mpg.de/~anke/`.

[9] H. P. Young and A. Levenglick. A consistent extension of Condorcet's election principle. *SIAM Journal on Applied Mathematics*, 35(2):285–300, 1978.

Locally optimal dissimilar paths in road networks

Stéphanie Vanhove, [1] Veerle Fack

Department of Applied Mathematics and Computer Science, Ghent University
Krijgslaan 281 - S9, 9000 Ghent, Belgium
Stephanie.Vanhove@UGent.be

Abstract

We present a new algorithm for calculating alternative routes. Our algorithm aims to find paths that are sufficiently different from each other and avoid illogical detours. It generates a large number of paths, selects a dissimilar subset and finally improves this subset. The results are of good quality.

Key words: Graph algorithms, alternative routes, dissimilar paths, road networks

1 Introduction

Modern route planning applications often give the user a few possibilities rather than only the shortest or fastest route. This gives the user the freedom to choose the route that best fits his own needs. Having several possible routes is also interesting in the context of the transportation of hazardous material in order to spread the risk, as described by Dell'Olmo et al. [1]. Naturally, alternative routes are only useful if they are not too much alike. The routes should be "dissimilar". While there are many algorithms for generating a ranking of k shortest paths (e.g. Eppstein [2], Yen [3], Hershberger et al. [4]), these algorithms are not suitable for this problem since the paths generated by these algorithms are usually very similar. Therefore, algorithms which specifically aim to find dissimilar paths of good quality are necessary.

2 Overview

Akgün et al. [5] give a comprehensive overview of existing methods for calculating dissimilar paths. Many methods rely on generating a large amount of paths of which a dissimilar subset is selected. The authors define the dissimilarity D between two paths P_i and P_j as follows:

[1] Stéphanie Vanhove is supported by a research grant of the Research Foundation Flanders (FWO).

Definition 2.1 $D(P_i, P_j) = 1 - [L(P_i \cap P_j)/L(P_i) + L(P_i \cap P_j)/L(P_j)]/2$

The authors also point out that finding a subset of dissimilar paths is essentially a *p-dispersion problem*, in which a subset needs to be chosen from a set of points such that the minimal dissimilarity between any pair of points in the subset is maximized. Since this is a computationally hard problem, faster approximation methods are often used. Several approximation methods can be found in Erkut et al. [6].

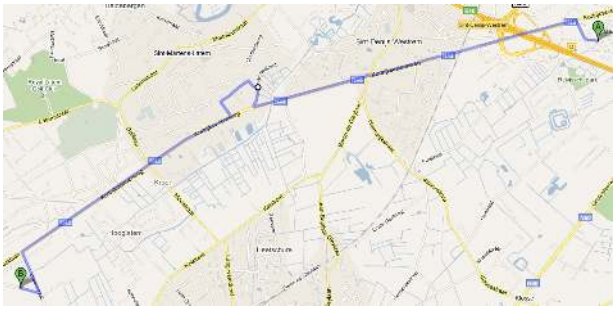


Fig. 1. A typical imperfection in generated alternative paths. The path makes a short detour in small side streets while it would be better to stay on the main road.

A problem with many of the existing algorithms, is that the generated paths can contain a lot of very small detours which are not logical to the driver. Figure 1 shows an example of such a situation. To overcome this problem, Abraham et al. [7] introduce the concept of *local optimality*.

Definition 2.2 *A path P is T-locally optimal if and only if every subpath P' of P with $weight(P') \leq T$ is a shortest path.*

T is usually chosen as a percentage of the shortest path weight, e.g. 25%.
In this work, we aim to develop an efficient method to find locally optimal dissimilar paths.

3 Our method

Our method for generating dissimilar paths starts by generating a large number of paths using a bidirectional heuristic. From these paths, a small subset of dissimilar paths is selected. Dissimilarity is defined as in Definition 2.1.

3.1 Generate a large number of paths

In order to generate a large number of paths, our algorithm uses a bidirectional heuristic based on the shortest path algorithm of Dijkstra [8]. Two shortest path trees are built. A forward instance of the algorithm of Dijkstra builds a forward shortest path tree from the start node, while a backward instance builds a backward shortest path tree to the target node. Both algorithms take turns in labeling the neighbours of one node. Each time both searches meet (i.e. a node is contained in

both shortest path trees) a new path is generated. Figure 2 illustrates this idea. The algorithm continues until the desired number of paths is obtained or until no more paths can be calculated.

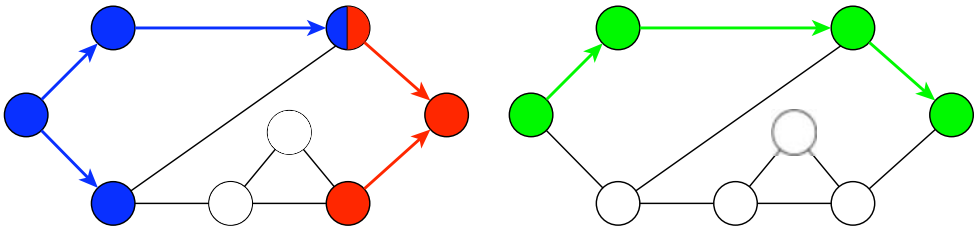


Fig. 2. Bidirectional heuristic. Left: the forward (blue) and backward (red) shortest path tree meet in the blue/red node. Right: path generated after both searches meet. The algorithm will continue to generate paths in the same way.

3.2 Find a subset of dissimilar paths

Our strategy for finding a subset of dissimilar paths is an approximation method based on the Greedy Construction heuristic proposed by Erkut et al. [6]. The shortest path is always included in the selected subset, since this is usually of interest to the user. After this, from all remaining paths, the path is chosen with the highest minimum dissimilarity to the paths already in the selection. This path is added to the selection. This process is repeated until the selection contains the desired number of paths.

3.3 Improve local optimality

The selected paths are now improved to make them locally optimal if necessary. The parts of the path which are not locally optimal are identified. If a subpath from u to v is not locally optimal, this subpath is replaced by the shortest path from u to v. This is done for all non-locally optimal subpaths in a path. However, there is no guarantee that the path is locally optimal after this. An iterative approach is needed which repeats this method until the path is locally optimal. Typically, less than 10 iterations are needed.

4 Conclusion and future work

An example of the paths found by our algorithm can be seen in Figure 3. The paths are clearly dissimilar without being excessively long and contain no illogical detours. This is a satisfying result and was calculated in just a few seconds.

In the future we aim to further speed up our algorithm and to further optimize the weight of the alternative paths.

Fig. 3. Example of a result found by our algorithm (shown as a layer on top of Google Maps). The shortest path is marked in red, the alternative paths in blue.

References

[1] Paolo Dell'Olmo, Monica Gentili, and Andrea Scozzari. Finding dissimilar routes for the transportation of hazardous materials. In *Proceedings of the 13th Mini-EURO Conference on Handling Uncertainty in the Analysis of Traffic and Transportation Systems*., pages 785–788, 2002.

[2] David Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28:652–673, 1998.

[3] Jin Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17:712–716, 1971.

[4] John Hershberger, Matthew Maxel, and Subhash Suri. Finding the k shortest simple paths: a new algorithm and its implementation. *ACM Transactions on Algorithms*, 3:45, 2007.

[5] Vedat Akgün, Erhan Erkut, and Rajan Batta. On finding dissimilar paths. *European Journal of Operational Research*, 121:232–246, 2000.

[6] Erhan Erkut, Yilmaz Ülküsal, and Oktay Yeniçerioglu. A comparison of p-dispersion heuristics. *Computers and Operations Research*, 21(10):1103 – 1113, 1994.

[7] Ittai Abraham et al. Alternative routes in road networks. In *SEA*, pages 23–34, 2010.

[8] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

Searching Circulant Graphs

Öznur Yaşar Diner, [a] Danny Dyer, [b]

[a]*Department of Information Technology, Kadir Has University, Turkey*
[b]*Department of Mathematics and Statistics,*
Memorial University of Newfoundland, Canada

Key words: Edge Searching, Circulant Graphs

1 Introduction

Edge searching (or graph searching) is an extensively studied graph theoretical problem. Its origins date back to the late 1960s in works of Parsons [10] and Breisch [3]. It was first faced by a group of spelunkers who were trying to find a person lost in a system of caves. They were interested in the minimum number of people they needed in the searching team and an optimal search strategy.

Assume that we want to secure a system of tunnels from a hidden intruder who is trying to avoid us and has unbounded speed. We model this system as a finite connected graph $G = (V, E)$ where junctions correspond to vertices and tunnels correspond to edges. We will launch a group of searchers into the system in order to catch the intruder. We assume that every edge of G is contaminated initially and our aim is to clean the whole graph by a sequence of steps. At each step we are allowed to do one of these moves: (1) Place a searcher at a vertex, (2) Remove a searcher from one vertex and place it on another vertex (a "jump"), (3) Slide a searcher from a vertex along an edge to an adjacent vertex. Note that placing multiple searchers on any vertex is allowed. We don't pose any restriction on the number of searchers used.

If a searcher slides along an edge $e = uv$ from u to v, then the edge e is *cleaned* if either (i) another searcher is stationed at u, or (ii) all other edges incident to u are already clean. An *edge search strategy* is a combination of the moves so that the state of all edges being simultaneously clean is achieved, in which case we say that the graph is *cleaned*. The least number of searchers needed to clean the graph is the *(edge) search number* of the graph and is denoted $s(G)$. The problem becomes cleaning (or searching) the graph using the fewest searchers. In this respect, we are interested in the optimal search strategies, those that use only $s(G)$ searchers.

For a given graph, it is a natural question to ask the following: *What is the smallest value of $s(G) = k$ with which we can clean the graph?* and *How can we clean the graph using the minimum possible number of searchers?*

Notice that even once an edge is cleaned, it may not necessarily be true that it will remain clean until the end of the search strategy. In other words, an edge can be cleaned at some step and at a later step it can get contaminated again. If a searcher is stationed at a vertex v, then we say that v is *guarded*. If a path does not contain any guarded vertex, then it is called an *unguarded path*. If there is an unguarded path that contains one endpoint of a contaminated edge and one endpoint of a cleaned edge e, then e gets *recontaminated*. Hence, a clean edge remains clean as long as every path from it to a contaminated edge is blocked by at least one searcher.

The edge search problem has many variants based on, for instance, how searchers move or how the edges are cleaned. Due to its closeness with the layout problems, the problem is related to widely utilized graph parameters such as pathwidth [5], bandwidth [6] and cutwidth of a graph which arises in VLSI design [4]. The problem and its variants are related to many applications such as network security [1] and robotics [8].

The NP-completeness of the Edge Searching problem and its variations invoked interest in solving these problems on special classes of graphs. In this note, we are going to consider edge searching of the Ciculant Graphs. This family of graphs play a significant role in many discrete optimization problems [2,9]. On a related pursuit evasion game, the cops and robber game, the cop number of the circulant graph with connection set of size s is fund to be at most $\lceil \frac{s+1}{2} \rceil$ in [7]. Here we give an upper bound on the edge search number of the circulant graphs of prime order and conjecture that this can be made small.

2 Searching Circulant Graphs

We consider edge searching of circulant graphs of prime order and state our conjecture. Let $(\mathcal{G}, +)$ be a finite group with identity element 0. Let $\mathcal{S} \subseteq (\mathcal{G} \backslash \{0\})$ such that $\mathcal{S} = -\mathcal{S}$, that is $a \in \mathcal{S}$ if and only if $-a \in \mathcal{S}$. Recall that $-a$ denotes the inverse of a in $(\mathcal{G}, +)$. The *Cayley graph* on a group $\mathcal{G}$ with *connection set (or generating set)* $\mathcal{S}$, denoted as $\mathrm{Cay}(\mathcal{G}, \mathcal{S})$, is the graph that is constructed as follows: (1) Each element of $\mathcal{G}$ corresponds to a vertex v_i, and, (2) There exists an edge joining v_i and v_j if and only if $v_i = v_j + a$ where $a \in \mathcal{S}$.

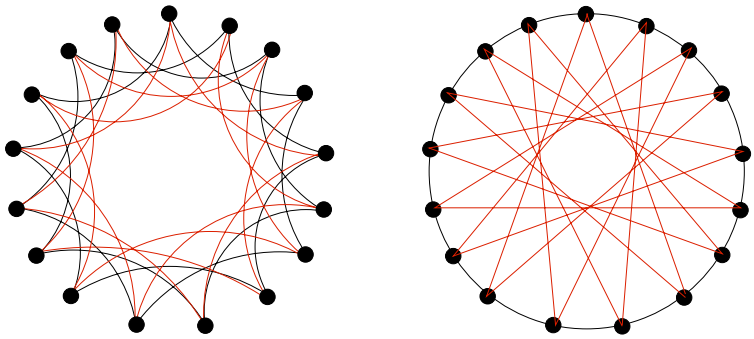


Fig. 1. The circulant graphs $\mathrm{circ}(17; 3, 4)$ and $\mathrm{circ}(17; 1, 7)$

A special class of Cayley graphs is those on cyclic groups. A *circulant graph*, denoted as $\mathrm{circ(n, S)}$, is the Cayley graph $\mathrm{Cay}(\mathbb{Z}_{\mathrm{n}}, \mathrm{S})$ where $\mathbb{Z}_n$ is the abelian group of integers modulo n.

Let p be a prime number and consider the circulant graph $\mathrm{circ}(p, S)$ where $\mathcal{S} \subseteq (\mathbb{Z}_p \backslash \{0\})$. Notice that $G = \mathrm{circ}(p, S)$ is a Hamiltonian cycle when $|S| = 2$ and $3 \leq p$; thus $s(G) = 2$. Nevertheless the calculation of the search number of $\mathrm{circ}(p, S)$ gets complicated rapidly as the size of the set S increases.

For brevity, we denote a circulant graph $G = \mathrm{circ}(p, S)$ with connection set $S = \{a, -a, b, -b\}$ as $\mathrm{circ}(p; a, b)$ where $1 \leq a < b \leq \frac{p-1}{2}$.

Theorem 2.1 If p is a prime number and $1 \leq a < b \leq \frac{p-1}{2}$, then

$$s(\mathrm{circ}(p; a, b)) \leq 2b + 1.$$

Proof (sketch) Place the $2b$ searchers on the following vertices: $v_1, v_2, \ldots, v_b$ and $v_{n-b+1}, v_{n-b+2}, \ldots, v_n$. Label these searchers as $\sigma_1, \sigma_2, \ldots, \sigma_{2b}$ respectively. Place σ_{2b+1} on v_1 and clean all the edges with end vetices in $\{v_{n-b+1}, v_{n-b+2}, \ldots, v_n, v_1, v_2, \ldots, v_b\}$. The only contaminated edge incident to v_1 is $v_1 v_{b+1}$, thus let σ_1 slide along this edge and clean v_1. Next let σ_1 clean all the edges with end vetices in $\{v_{n-b+1}, v_{n-b+2}, \ldots, v_n, v_2, \ldots, v_b, v_{b+1}\}$. Now σ_2 can slide along $v_2 v_{b+2}$ and clean v_2. We clean $v_1, v_2, \ldots, v_b$ in the same way. We repeat this shifting of searchers on $v_1, v_2, \ldots, v_b$ to $v_{b+1}, v_{b+2}, \ldots, v_{2b}$ for every group of b consequtive vertex and clean the whole graph. ∎

The upper bound in Theorem 2.1 is tight for some graphs (for instance, when $G = \mathrm{circ}(5; 1, 2)$). On the other hand, this bound will be big when b is large. We claim that this number can be made as small as twice the root of the order of the graph plus one.

In order to consider isomorphic circulant graphs, we use multiplication in $\mathbb{Z}_p$. Let $f : \{1, 2, \ldots, p\} \to \{1, 2, \ldots, p\}$ so that $f(n) = (n-1)a + 1$ where $1 \leq a < \frac{p-1}{2}$. It is a simple observation that f is an isomorphism between $\mathrm{circ}(p; a, b)$ and $\mathrm{circ}(p; 1, c)$ where $1 \leq a < b \leq \frac{p-1}{2}$ and $c = ba^{-1}$.

Furthermore, we can show that for any $k \geq 1$, $\mathrm{circ}(p; 1, c) \simeq \mathrm{circ}(p; k, ck)$.

The following conjecture gives a bound on the product of an element of $\mathbb{Z}_p$ and a positive integer less then the ceiling of the root of p.

Conjecture 4 For every prime p and every integer $i = 1, 2, \ldots, \frac{p-1}{2}$, there exists an integer j, $1 \leq j \leq \lceil \sqrt{p} \rceil$ such that either

$$ij \leq \lceil \sqrt{p} \rceil \pmod{p}, \text{ or, } p - ij \leq \lceil \sqrt{p} \rceil \pmod{p}.$$

An ongoing Maple code that mininimizes the maximum desired product shows that Conjecture 4 holds for up to the 6000th prime. Thus by Theorem 2.1 and the isomophisms we've given, the following is true for the first 6000 primes:

$$s(\mathrm{circ}(p, S)) \leq 2\lceil \sqrt{p} \rceil + 1 \tag{2.1}$$

for every circulant graph, $\mathrm{circ}(p, S)$, where $S \subseteq (\mathbb{Z}_p \backslash \{0\}), |S| \leq 4$.

This is a very good bound considering the size of the graph and the existing upper bounds on edge search number.

References

[1] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro, *Capture of an intruder by mobile agents*. Proceedings of the 14th annual ACM symposium on Parallel algorithms and architectures (2002) pp. 200–209.

[2] L. Barrière, P. Fraigniaud, C. Gavollile, B. Mans, and J.M. Robson. *On recognizing Cayley graphs*. LNCS, Vol. 1879 Springer-Verlag (2000) pp. 76–87.

[3] R. L. Breisch, *An intuitive approach to speleotopology*, Southwestern Cavers 6 (1967) pp. 72–78.

[4] F. Chung, *On the cutwidth and the topological bandwidth of a tree*, SIAM J. Algebraic Discrete Methods 6 (1985) pp. 268–277.

[5] J. A. Ellis, I. H. Sudborough, J. S. Turner, *The Vertex Separation and Search Number of a Graph*, Information and Computation 113 (1994) pp. 50–79.

[6] F. V. Fomin, P. Heggernes and J. A. Telle, *Graph searching, elimination trees, and a generalization of bandwidth*, Algorithmica 41 (2004) pp. 73–87.

[7] P. Frankl, *Cops and robbers in graphs with large girth and Cayley graphs*, Discrete Appl. Math. 17 (1987) pp. 301–305.

[8] L. J. Guibas, Jean-Claude Latombe, Steven M. LaValle, David Lin, Rajeev Motwani, *A Visibility-Based Pursuit-Evasion Problem*, International Journal of Computational Geometry and Applications 9, No: 4/5 (1999) pp. 471–499.

[9] E. A. Monakhova, O. G. Monakhov and E. V. Mukhoed, *Genetic Construction of Optimal Circulant Network Designs*, Evolutionary Image Analysis, Signal Processing and Telecommunications LNCS, Vol. 1596, (1999) pp. 215–223.

[10] T. Parsons, *Pursuit-evasion in a graph*, Theory and Applications of Graphs, Lecture Notes in Mathematics, Springer-Verlag (1976) pp. 426–441.

Detailed Conference Program

Tuesday 14

Opening session (9.10 Sala del Teatro)

Bicliques and Bipartite Graphs (Tue A1, Sala del Teatro)

Chairman: P. Nobili

9.20 Couturier J. F. and D. Kratsch, Bicolored independent sets and bicliques

9.50 Albano A. and A. Do Lago, New upper bound for the number of maximal bicliques of a bipartite graph

10.20 Nicoloso S. and U. Pietropaoli, Bipartite finite Toeplitz graphs

Games and multi-decisor systems (Tue A2, Sala S.S. Pietro e Paolo)

Chairman: R. Schrader

9.20 Van Zuylen A., F. Schalekamp and D. P. Williamson, Popular Ranking

9.50 Kern W. and X. Qiu, Improved Taxation Rate for Bin Packing Games

10.20 Albrecht K. and U. Faigle, Binary Betting Strategies with Optimal Logarithmic Growth

Coffee break (10.50–11.10)

Mathematical Programming (Tue B1, Sala del Teatro)

Chairman: A. Agnetis

11.10 Argiroffo G. and A. Wagler, Generalized row family inequalities for the set covering polyhedron

11.40 De Santis M., S. Lucidi and F. Rinaldi, A New Feasibility Pump-Like Heuristic for Mixed Integer Problems

12.10 Amaldi E., S. Coniglio and L. Taccari, Formulations and heuristics for the k-Piecewise Affine Model Fitting problem

12.40 Arbib C., G. Felici and M. Servilio, Sorting Common Operations to Minimize Tardy Jobs

Graph Theory 1 (Tue B2, Sala S.S. Pietro e Paolo)

Chairman: F. Bonomo

11.10 Milanic M., A hereditary view on efficient domination

11.40 Skupień Z., Majorization and the minimum number of dominating sets

12.10 Bermudo S. and H. Fernau, Computing the differential of a graph

12.40 Saputro S.W., R. Simanjuntak , S. Uttunggadewa, H. Assiyatun, E. Tri Baskoro, A.N.M Salman, On graph of order-n with the metric dimension $n - 3$

Lunch (13.10–14.10)

Cliques (Tue C1, Sala del Teatro)

Chairman: S. Nicoloso

14.10 Snels C., G. Oriolo and F. Bonomo, A primal algorithm for the minimum weight clique cover problem on a class of claw-free perfect graphs

14.40 Nobili P. and A. Sassano, A reduction algorithm for the weighted stable set problem in claw-free graphs

10.20 Bonomo F. and J. L. Szwarcfiter, Characterization of classical graph classes by weighted clique graphs

Scheduling (Tue C2, Sala S.S. Pietro e Paolo)

Chairman: J. Hurink

14.10 Adacher L. and M. Flamini, Modeling and solving aircrafts scheduling problem in ground control

14.40 Sevastyanov S. and B. Lin, Efficient enumeration of optimal and approximate solutions of a scheduling problem

15.10 Agnetis A., P. Detti, M. Pranzo and P. Martineau, Scheduling problems with unreliable jobs and machines

Coffee break (15.40–16.10)

Computational Biology (Tue D1, Sala del Teatro)

Chairman: U. Pferschy

16.10 Liberti L., B. Masson, C. Lavor and A. Mucherino, Branch-and-Prune trees with bounded width

16.40 Mucherino A., I. Wohlers, G. Klau and R. Andonov, Sparsifying Distance Matrices for Protein-Protein Structure Alignments

17.10 Cordone R. and G. Lulli, A Lagrangian Relaxation Approach for Gene Regulatory Networks

Telecom (Tue D2, Sala S.S. Pietro e Paolo)

Chairman: C. Arbib

16.10 Abrardo A, M. Belleschi and P. Detti, Resources and transmission formats allocation in OFDMA networks

16.40 Addis B., G. Carello and F. Malucelli, Network design with SRG based protection

17.10 Alba M., F. Clautiaux, M. Dell'Amico and M. Iori, Models and Algorithms for the Bin Packing Problem with Fragile Objects

Wednesday 15

Knapsack (Wed A1, Sala del Teatro)

Chairman: M. Iori

9.20 Monaci M. and U. Pferschy, On the Robust Knapsack Problem

9.50 Kosuch S., Approximability of the Two-Stage Knapsack problem with discretely distributed weights

10.20 Cello M., G. Gnecco, M. Marchese and M. Sanguineti, A Generalized Stochastic Knapsack Problem with Application in Call Admission Control

Graph Theory 2 (Wed A2, Sala S.S. Pietro e Paolo)

Chairman: A. Wagler

9.20 Yasar Diner O. and D. Dyer, Searching Circulant Graphs

9.50 Bina W., Enumeration of Labeled Split Graphs and Counts of Important Superclasses

10.20 Kochol M., Decomposition of Tutte Polynomial

Coffee break (10.50–11.10)

Routing (Wed B1, Sala del Teatro)

Chairman: G. Righini

11.10 Factorovich P., I. Méndez-Díaz and P. Zabala, The Pickup and Delivery Problem with Incompatibility Constraints

11.40 Kjeldsen N. and M. Gamst, The Boat and Barge Problem

12.10 Roda F., P. Hansen and L. Liberti, The price of equity in the Hazmat

12.40 Méndez-Díaz I., J. J. Miranda Bront, P. Toth and P. Zabala, Infeasible path formulations for the time-dependent TSP with time windows

Nonlinear Optimization (Wed B2, Sala S.S. Pietro e Paolo)

Chairman: L. Liberti

11.10 Rinaldi F., M. De Santis and S. Lucidi, Continuous Reformulations for Zero-one Programming Problems

11.40 Costa A., Pierre Hansen and Leo Liberti, Bound constraints for Point Packing in a Square

12.10 Coniglio S., The impact of the norm on the k-Hyperplane Clustering problem: relaxations, restrictions, approximation factors, and exact formulations

12.40 Varvitsiotis A. and M. Laurent, Computing the Grothendieck constant of some graph classes

Lunch (13.10–14.10)

Session in memory of Bruno Simeone I (Wed C, Sala del Teatro)

Chairman: I. Lari

14.10 Becker R., Research work with Bruno Simeone

14.40 Hansen P., Bruno Simeone's Work in Clustering

10.20 Crama Y., Control and voting power in complex shareholding networks

Coffee break (15.40–16.10)

Session in memory of Bruno Simeone II (Wed D, Sala del Teatro)

Chairman: F. Ricca

16.10 Golumbic M., Graph sandwich problems

16.40 Boros E., Incompatibility graphs and data mining

17.10 Serafini P., Separating negative and positive points with the minimum number of boxes

Thursday 16

Coloring (Thu A1, Sala del Teatro)

Chairman: P. Dell'Olmo

9.20 Calamoneri T. and B. Sinaimeri, Labeling of Oriented Planar Graphs

9.50 Petrosyan P. and R. Kamalian, Edge-chromatic sums of regular and bipartite graphs

10.20 Boehme T. and J. Schreyer, Local Computation of Vertex Colorings

Non deterministic systems (Thu A2, Sala S.S. Pietro e Paolo)

Chairman: S. Pickl

9.20 Faigle U. and A. Schoenhuth, Representations of Power Series over Word Algebras

9.50 Pascucci F. and M. Carli, Sensor Network Localization Using Compressed Extended Kalman Filter

10.20 Lozovanu D. and S. Pickl , Discounted Markov Decision Processes and Algorithms for Solving Stochastic Control Problem on Networks

Coffee break (10.50–11.10)

Cuts and Flows (Thu B1, Sala del Teatro)

Chairman: F. Malucelli

11.10 Cullenbine C. , K. Wood and A. Newman, New Results for the Directed Network Diversion Problem

11.40 Stephan R., Reducing the minimum T-cut problem to polynomial size linear programming

12.10 Bauer J., One Minimum-Cost Network Flow Problem to Identify a Graph's Connected Components

12.40 Vanhove S. and V. Fack, Locally optimal dissimilar paths in road networks

Metaheuristics (Thu B2, Sala S.S. Pietro e Paolo)

Chairman: M. Sanguineti

11.10 Cano R. G., G. Kunigami, C.C. De Souza and P. J. De Rezende, Effective drawing of proportional symbol maps using GRASP

11.40 Touati-Moungla N. and D. Brockhoff, An Evolutionary Algorithm for the Multiobjective Risk-Equity Constrained Routing Problem

12.10 Dell'Olmo P., R. Cerulli and F. Carrabs, The maximum labeled clique problem

12.40 Gaudilliere A., A. Iovanella, B. Scoppola, E. Scoppola and M. Viale, A Probabilistic Cellular Automata algorithm for the clique problem

Lunch (13.10–14.10)

Location (Thu C1, Sala del Teatro)

Chairman: P. Serafini

14.10 Tresoldi E., G. Righini and A. Ceselli, Combined Location and Routing Problems in Drug Distribution

14.40 Puerto J., F. Ricca and A. Scozzari, Range minimization problems in path-facility location on trees

10.20 Bruglieri M., P. Cappanera and M. Nonato, The gateway location problem for hazardous material transportation

Graph Theory 3 (Thu C2, Sala S.S. Pietro e Paolo)

Chairman: U. Faigle

14.10 Cerioli M.R., H. Nobrega and P. Viana, A partial characterization by forbidden subgraphs of edge path graphs

14.40 Golovach P., M. Kaminski and D. Thilikos, Odd cyclic surface separators in planar graphs

15.10 Narayanan N., Minimally 2-connected graphs and colouring problems

Coffee break (15.40–16.10)

Combinatorial Optimization (Thu D1, Sala del Teatro)

Chairman: G. Felici

16.10 Hossain S., Computing Derivatives via Compression : An Exact Scheme

16.40 Amaldi E., C. Iuliano and R. Rizzi, On cycle bases with limited edge overlap

Graph Theory 4 (Thu D2, Sala S.S. Pietro e Paolo)

Chairman: C. Snels

16.10 Scheidweiler R. and E. Triesch, Matchings in balanced hypergraphs

16.40 Torres L. and A. Wagler, The dynamics of deterministic systems from a hypergraph theoretical point of view

Closing session (17.10 Sala del Teatro).

Authors Index

Index