9th Cologne-Twente Workshop on Graphs and Combinatorial Optimization

Cologne, Germany, May 25-27, 2010

Extended Abstracts

Ulrich Faigle, Rainer Schrader, Daniel Herrmann (eds.)

Table of Contents

Appendix

Semidefinite Programming for Stochastic Wireless OFDMA Networks

Pablo Adasme [a,1], Abdel Lisser [a,2]

[a] *Laboratoire de Recherche en Informatique, Université Paris-Sud XI, Bâtiment 490, 91405, Orsay Cedex France*

Key words: Risk modeling, stochastic programming, semidefinite programming, resource allocation in OFDMA.

1 Introduction

Resource allocation such as maximizing link capacity or minimizing power consumption in a wireless OFDMA network are commonly formulated as mathematical programs [1]. These programs usually involve random variables in the input data. In this paper, we propose a (0-1) stochastic quadratic formulation. The study is made on the basis of an OFDMA quadratic model [4] in which a probabilistic constraint based approach is considered [2]. Then, a semidefinite programming (SDP) relaxation is derived to solve the stochastic quadratic model. The paper is organized as follows: Section 2 presents the stochastic quadratic formulation. Section 3 presents the SDP relaxation. Finally, section 4 concludes the paper.

2 Probabilistic formulation

We consider an OFDMA network composed by a base station (BS) and several mobile users. The BS has to assign a set of N sub-carriers to a set of K users using a modulation size of $c \in \{1, \ldots, M\}$ bits in each sub-carrier. The goal is to minimize the total power consumption in the network. We consider the following probabilistic constrained model [4]:

[1] Email: pablo.adasme@lri.fr
[2] Email: abdel.lisser@lri.fr

$$\text{SQIP0: } \min_{\{x_{k,n},y_{n,c}\}} \quad \sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{c=1}^{M} P_{k,n}^{c} x_{k,n} y_{n,c} \tag{1}$$

$$\text{st:} \quad \mathbb{P}\{\sum_{n=1}^{N} x_{k,n}\left[\sum_{c=1}^{M} c \cdot y_{n,c}\right] \geq R_k\} \geq (1-\alpha_k), \quad \forall k \tag{2}$$

$$\sum_{k=1}^{K} x_{k,n} \leq 1 \quad \forall n \tag{3}$$

$$\sum_{c=1}^{M} y_{n,c} \leq 1 \quad \forall n \tag{4}$$

$$x_{k,n}, y_{n,c} \in \{0,1\} \tag{5}$$

Here, the objective function represents the total power consumption. The first constraint corresponds to a chance constraint where α_k is the risk to be taken for each user k. In this model, we consider separated chance constraints. The second constraint imposes that each sub-carrier should be assigned to only one user at a time while the third one imposes that each sub-carrier must use one integer modulation size. The decision variables are given by $x_{k,n}$ and $y_{n,c}$, respectively. We assume that R_k are random variables with joint probability distribution H. Let us consider the case where H is concentrated in the finite number of points also called scenarios $R_k = (r_{k,1}, ..., r_{k,l}, ..., r_{k,L_k})$ with probabilities $p_{k,l}$ such that $\sum_{l=1}^{L_k} p_{k,l} = 1, p_{k,l} \geq 0, \forall k$.

Then, the problem (1)-(5) can be reformulated as follows [2]:

$$\text{SQIP1: } \min_{\{x_{k,n},y_{n,c}\}} \quad \sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{c=1}^{M} P_{k,n}^{c} x_{k,n} y_{n,c} \tag{6}$$

$$\text{st:} \quad \sum_{n=1}^{N} x_{k,n}\left[\sum_{c=1}^{M} c \cdot y_{n,c}\right] \geq r_{k,l} \quad \forall l \in \Gamma_k, \forall k \tag{7}$$

$$\sum_{l\in\Gamma_k} p_{k,l} \geq 1-\alpha_k \quad \forall k \tag{8}$$

$$\sum_{k=1}^{K} x_{k,n} \leq 1 \quad \forall n \tag{9}$$

$$\sum_{c=1}^{M} y_{n,c} \leq 1 \quad \forall n \tag{10}$$

$$x_{k,n}, y_{n,c} \in \{0,1\} \tag{11}$$

Constraints (8) mean that we have to choose a subset Γ_k of scenarios such that the sum of the probabilities of this subset is greater than $(1-\alpha_k)$. For this subset, the bit rate constraints will be active and valid, whereas for the scenarios not in this subset, the constraints are not activated.

This problem can be reformulated by introducing the auxiliary binary variable $\varphi_{k,l}$ for each observation $l = 1 : L_k, \forall k$ as follows:

$$\varphi_{k,l} = \begin{cases} 0 \text{ if } \ l \in \Gamma_k \\ 1 \text{ otherwise} \end{cases} \tag{12}$$

This yields the following problem:

$$\text{SQIP2: } \min_{\{x_{k,n},y_{n,c},\varphi_{k,l}\}} \quad \sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{c=1}^{M} P_{k,n}^{c} x_{k,n} y_{n,c} \tag{13}$$

$$\text{st:} \quad \sum_{n=1}^{N} x_{k,n} \left[\sum_{c=1}^{M} c \cdot y_{n,c} \right] \geq r_{k,l} - \mathcal{M}\varphi_{k,l} \quad \forall k, l = 1 : L_k \tag{14}$$

$$\sum_{l=1}^{L_k} p_{k,l}\varphi_{k,l} \leq \alpha_k \quad \forall k \tag{15}$$

$$\sum_{k=1}^{K} x_{k,n} \leq 1 \quad \forall n \tag{16}$$

$$\sum_{c=1}^{M} y_{n,c} \leq 1 \quad \forall n \tag{17}$$

$$x_{k,n}, y_{n,c}, \varphi_{k,l} \in \{0, 1\} \tag{18}$$

where $\mathcal{M}$ is an arbitrary number such that

$$\mathcal{M} \geq \max_{k,l}\{r_{k,l}\} + 1 \tag{19}$$

This problem is a quadratic optimization problem with binary variables. This quadratic problem is NP-hard, and so is its stochastic formulation. In this case, we seek lower bounds using strong relaxations, namely SDP relaxations.

3 Semidefinite relaxation

In order to write a SDP relaxation for SQIP2, we define the (0-1) vector $z^T = (x_{1,1}, \cdots, x_{1,N}, \cdots, x_{K,1}, \cdots, x_{K,N}, y_{1,1}, \cdots, y_{1,M}, \cdots, y_{N,1}, \cdots, y_{N,M}, \varphi_{1,1}, \cdots, \varphi_{1,L_1}, \cdots, \varphi_{K,1}, \cdots, \varphi_{K,L_k})$. Then, let Z be a symmetric positive semidefinite matrix defined as:

$$Z = \begin{pmatrix} zz^T & z \\ z^T & 1 \end{pmatrix} \succeq 0 \tag{20}$$

We can construct symmetric matrices $\mathcal{P}$ for the objective function in (13), $U_{k,l}$ for constraints in (14) and V_k for constraints in (15). We propose the following SDP relaxation for SQIP2:

$$\text{SSDP2} : \min_{Z} \quad Trace(\mathcal{P}Z) \tag{21}$$

$$\text{st: } Trace(U_{k,l}Z) \geq r_{k,l} \quad \forall k, l = 1 : L_k \tag{22}$$

$$Trace(V_kZ) \leq \alpha_k \quad \forall k \tag{23}$$

$$Trace([ex_n][ex_n]^T Z) \leq 1 \quad \forall n \tag{24}$$

$$Trace([ey_n][ey_n]^T Z) \leq 1 \quad \forall n \tag{25}$$

$$Trace(\zeta_{k,n}^c Z) \geq 0 \quad \forall k, n, c \tag{26}$$

$$diag(zz^T) = z \tag{27}$$

$$Z \succeq 0 \tag{28}$$

In this model, $[ex_n]$ and $[ey_n]$ are coefficient vectors for constraints in (16) and (17) according to vector z. Thus, the rank-1 matrices we construct with these vectors are used to strength our SDP relaxation [3]. The symmetric matrices $\zeta^c_{k,n}$ for all $\{k, n, c\}$ are used to have positive values in matrix Z only in the positions where $\{\mathcal{P}_{i,j}, i < j\}$ is positive, this is, in the entries of $\mathcal{P}$ where we put the elements $\{P^c_{k,n}, k, n, c\}$ from (13). Finally, constraint (27) together with constraint (28) form a relaxation constraint for the condition of $z_i \in \{0, 1\}$ for all i. The last constraint also imposes the condition on matrix Z to be positive semidefinite. Our SDP relaxation is tighter than the linear program (LP) we obtain by applying Fortet linearization method [5] to SQIP2 as shown by our preliminary results.

4 Conclusions

In this paper, we proposed a stochastic quadratic formulation for wireless OFDMA networks. To this purpose, we considered an OFDMA quadratic model [4] in which probabilistic constraints are added by using the approach of [2]. Finally, a SDP relaxation is derived. Numerical results are given.

References

[1] Amzallag, D. Armarnik, T. Livschitz, M. Raz, D., *"Multi-Cell Slots Allocation in OFDMA Systems,"* Mobile and Wireless Communications Summit, 16th IST, 2007.

[2] Lisser A., Lopez R. and Hu Xu, *"Stochastic Quadratic Knapsack with Recourse,"* International Network Optimization Conference, INOC-2009, April 2009.

[3] Helmberg, C., *"Semidefinite Programming for Combinatorial Optimization,"* ZIB-Report ZR-00-34, Konrad-Zuse-Zentrum Berlin, October 2000.

[4] Adasme Pablo, Lisser Abdel and Soto Ismael, *"Robust Semidefinite Relaxations for a New Quadratic OFDMA Resource Allocation Approach,"* Working Paper Number 1522, LRI, University of Paris Sud, France.

[5] Fortet R., *"Applications de l'algebre de boole en recherche operationelle,"* Revue Francaise de Recherche Operationelle, Vol. 4, pp. 17–26, 1960.

On finding a minimum weight cycle basis with cycles of bounded length

Edoardo Amaldi [a], Bernard Fortz [b], Claudio Iuliano [a]

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

[b] *Département d'Informatique, Faculté des Sciences, Université Libre de Bruxelles, Bruxelles, Belgium*
bernard.fortz@ulb.ac.be

Key words: undirected graphs, cycle basis, cycle, bounded length

1 Introduction

Consider a connected undirected graph $G = (V, E)$ without loops and multiple edges. Let $n = |V|$ and $m = |E|$ be respectively the number of vertices and edges. A generalized cycle is a subset of edges $C \subseteq E$ such that every vertex of V is incident to an even number of edges in C. All the cycles of G form a vector space, the so-called *cycle space*. Given an undirected graph G with a nonnegative weight w_e assigned to each edge $e \in E$, the *Minimum Cycle Basis* problem consists in finding a *cycle basis* $\mathcal{C}$ of minimum total weight $w(\mathcal{C}) = \sum_{C \in \mathcal{C}} w(C)$, where the weight of a cycle is defined as $w(C) = \sum_{e \in C} w_e$. This problem has been extensively studied, both from the algorithmic and the structural point of views. See the most recent works [1,2], the survey [9] and the references therein.

In this work, we investigate an interesting and natural variant, that we refer to as the *Minimum cycle basis with cycles of bounded length* problem. Given an undirected graph G with a nonnegative weight w_e and a nonnegative length l_e assigned to each edge $e \in E$ and a positive integer L, we wish to find a minimum (weight) cycle basis where each cycle C has a length $l(C) = \sum_{e \in C} l_e$ at most L. Without loss of generality we assume nonnegative integer weights and lengths on all the edges. The special case in which each cycle must contain at most k edges ($l_e = 1$ for each $e \in E$) is referred to as *Minimum k-edge-cycle basis*. Cycles with a bounded number of edges naturally arise in a number of contexts, see for instance [5], where k-edge-cycle bases play an important role.

2 Minimum cycle bases with cycles of bounded length

The existence of a cycle basis with cycles of length at most L clearly depends on the value of L and, as also noticed in [7], it can be checked in polynomial time by looking for a cycle basis with a shortest (in terms of length) longest cycle, see [3] for the algorithm. Even if the longest cycle has a length smaller than L it is hard to find one with minimum total weight.

Proposition 1 *The problem of finding a minimum cycle basis with cycles of bounded length is NP-hard.*

Proof We proceed by polynomial-time reduction from the *Partition* problem, known to be NP-complete [6], to the decision version of the above problem. In the Partition problem, given a set of N items, each with an integer size a_j, we have to decide whether there exists a subset A of items such that $\sum_{j\in A} a_j = \frac{1}{2}\sum_{j=1}^{N} a_j$. For each instance of the Partition problem it is easy to construct a special instance of the Minimum cycle basis with cycles of bounded length problem such that the answer to the former is yes if and only if the answer to the latter is yes. Consider a graph with $2N+1$ vertices $v_1 \dots v_{2N+1}$ and assume that they are ordered along a line. For every odd i, with $1 \le i \le 2N$, v_i is connected to v_{i+1} by an edge with weight $a_{(i+1)/2}$ and length 0 and to v_{i+2} by an edge with length $a_{(i+1)/2}$ and weight 0. For every even i, with $1 \le i \le 2N$, v_i is connected to v_{i+1} by an edge with both weight and length 0. The vertices v_1 and v_{2N+1} are also connected by an edge with both weight and length 0. Hence, the total number of edges is equal to $3N+1$. Let $W = L = \frac{1}{2}\sum_{j=1}^{N} a_j$. A minimum cycle basis consists of $N+1$ cycles: the N smaller cycles with both weight and length a_j (for a partial total weight $2W$) and the larger cycle given by the edge joining v_1 and v_{2N+1} plus a path through the other vertices. Finding a cycle basis with total weight $\le 3W$ and with cycles whose length is bounded by L corresponds to finding a path with total weight $\le W$ and length $\le L$ from v_1 to v_{2N+1} in the above graph without the edge joining them. This path yields a partition into nonzero-weight edges and nonzero-length edges that solves the Partition problem. □

We now show that the Minimum k-edge-cycle basis problem, namely the special case where $l_e = 1$ for each $e \in E$, can be solved in polynomial time. We adapt Horton's approach [8] to the bounded problem. In Horton's algorithm, a polynomial subset of candidate cycles is generated. For every vertex $x \in V$ and every edge $e = \{y, z\} \in E$ we consider the cycle C formed by the union of the two minimum weight paths p_{xy} and p_{xz} from x to the endpoints of e, y and z, plus the edge e itself, i.e., $C = p_{xy} + p_{xz} + \{y, z\}$. We say that C has a representation $(x, \{y, z\})$. The candidate cycles are then sorted by nondecreasing weight and a minimum cycle basis is given by the $m - n + 1$ lightest independent cycles.

Unfortunately, a minimum k-edge-cycle basis is not guaranteed to be contained in the set of Horton candidate cycles. As an example, consider the sunflower graph in [9, Fig. 7] and assume that the three edges of the internal triangle have weight equal to 3 whereas the other edges have weight equal to 1. The unique 3-edge-cycle basis is given by the four triangles, but the internal one is not a Horton candidate cycle.

The proposition in [8] stating that given a cycle C in a minimum cycle basis, for any pair of vertices $u, v \in C$ the minimum weight path p_{uv} must be contained in C, is no longer valid. Denoting by p^l_{uv} the minimum weight path between vertices u and v with a most l edges, we have the following result.

Proposition 2 *For any two vertices u and v of a cycle C in a minimum k-edge-cycle basis, let $P_1(u, v)$ and $P_2(u, v)$ be the two paths joining vertices u and v in C. Given two integers l_1 and l_2 greater than the number of edges in $P_1(u, v)$ and $P_2(u, v)$, respectively, and such that $l_1 + l_2 = k$, at least one between $p^{l_1}_{uv}$ and $p^{l_2}_{uv}$ must be contained in C.*

Proof Suppose it is not true. Then C can be obtained as the composition of three cycles $P_1(u, v) + p^{l_2}_{uv}$, $P_2(u, v) + p^{l_1}_{uv}$, and $p^{l_1}_{uv} + p^{l_2}_{uv}$, all of lighter weight than C and with a number of edges bounded by k. Thus C cannot be contained in a minimum k-edge-cycle basis. □

All the candidate k-edge-cycles can be generated by considering $C = p^{l_1}_{xy} + p^{l_2}_{xz} + \{y, z\}$ for any vertex $x \in V$, edge $e = \{y, z\} \in E$ and all the $k - 2$ possible choices of positive integers l_1 and l_2 such that $l_1 + l_2 = k - 1$, as in [7]. This naive approach can, however, be improved on by exploiting the notion of *isometric cycle* [1] for the unconstrained case. A cycle C is isometric if and only if it has a representation $(x, \{y, z\})$ for each vertex $x \in C$.

Proposition 3 *Every isometric cycle C has a representation $(x, \{y, z\})$ for a certain pair of vertex $x \in V$ and edge $\{y, z\} \in E$ that is balanced, i.e., such that the difference between the number of edges in p_{xy} and p_{xz} is 1 if C has an even number of edges and 0 if odd.*

For the lack of space, we cannot report the proof that is based on the efficient $O(nm)$ procedure for detecting isometric cycles proposed in [1]. The above result is also valid for cycles with at most k edges. Indeed, we only need to generate the candidate k-edge-cycles $C = p^{l_1}_{xy} + p^{l_2}_{xz} + \{y, z\}$ for every vertex $x \in V$ and edge $e = \{y, z\}$, for a choice of l_1 and l_2 leading to a balanced representation, if it exists. In this set of $O(nm)$ candidate k-edge-cycles, each cycle has a length of at most k. Since k is a constant, the total number of edges in all these cycles is $O(nm)$. Thus, by using the improved independence test recently proposed in [2], we obtain an $O(m^2 n / \log n)$ deterministic algorithm like for the unconstrained case. The independence test is inspired by de Pina's method [4], that maintains at each step a basis of the linear space that is

orthogonal to the subspace spanned by the cycles selected so far. It takes advantage of the divide and conquer scheme presented in [10] and uses a bit packing technique that exploits the sparseness property, namely the fact that the number of edges in all the candidate cycles is $O(nm)$.

Finally, it is worth pointing out that, although de Pina's algorithm [4] (improved in [10]) can be easily adapted to solve Minimum k-edge-cycle basis problem by just considering minimum weight paths with at most k edges, it leads to a worse $O(m^2n + mn^2 \log n)$ complexity.

References

[1] E. Amaldi, C. Iuliano, T. Jurkiewicz, K. Mehlhorn, and R. Rizzi. Breaking the $O(m^2n)$ barrier for minimum cycle bases. In A. Fiat and P. Sanders, editors, *ESA*, volume 5757 of *LNCS*, pages 301–312. Springer, 2009.

[2] E. Amaldi, C. Iuliano, and R. Rizzi. Efficient deterministic algorithms for finding a minimum cycle basis in undirected graphs. In F. Eisenbrand and B. Shepherd, editors, *Integer Programming and Combinatorial Optimization (IPCO)*, volume 6080 of *LNCS*, pages 397–410. Springer, 2010.

[3] D. M. Chickering, D. Geiger, and D. Heckerman. On finding a cycle basis with a shortest maximal cycle. *Inf. Process. Lett.*, 54(1):55–58, 1995.

[4] J. C. De Pina. *Applications of shortest path methods*. PhD thesis, University of Amsterdam, The Netherlands, 1995.

[5] B. Fortz and M. Labbé. Two-connected networks with rings of bounded cardinality. *Computational Optimization and Applications*, 27(2):123–148, 2004.

[6] 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.

[7] D. S. Hochbaum and E. V. Olinick. The bounded cycle-cover problem. *INFORMS J. on Computing*, 13(2):104–119, 2001.

[8] J. D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM J. Computing*, 16(2):358–366, 1987.

[9] 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.

[10] T. Kavitha, K. Mehlhorn, D. Michail, and K. E. Paluch. An $\tilde{O}(m^2n)$ algorithm for minimum cycle basis of graphs. *Algorithmica*, 52(3):333–349, 2008.

The classification of B-perfect graphs

Stephan Dominique Andres

Department of Mathematics and Computer Science, FernUniversität in Hagen, Germany

Key words: game chromatic number, game-perfectness, trivially perfect, graph colouring game

1 Introduction

Consider the following game, played on an (initially uncolored) graph $G = (V, E)$ with a color set C. The players, Alice and Bob, move alternately. A move consists in coloring a vertex $v \in V$ with a color $c \in C$ in such a way that adjacent vertices receive distinct colors. If this is not possible any more, the game ends. Alice wins if every vertex is colored in the end, otherwise Bob wins.

This type of game was introduced by Bodlaender [2]. He considers a variant, which we will call game g, in which Alice must move first and passing is not allowed. In order to obtain upper and lower bounds for a parameter associated with game g, two other variants are useful. In the game B Bob may move first. He may also miss one or several turns, but Alice must always move. In the other variant, game A, Alice may move first and miss one or several turns, but Bob must move. So in game B Bob has some advantages, whereas in game A Alice has some advantages with respect to Bodlaender's game.

For any variant $\mathcal{G} \in \{B, g, A\}$, the smallest cardinality of a color set C, so that Alice has a winning strategy for the game $\mathcal{G}$ is called *$\mathcal{G}$-game chromatic number* $\chi_{\mathcal{G}}(G)$ of G.

Email address: dominique.andres@fernuni-hagen.de (Stephan Dominique Andres).

URL: http://www.fernuni-hagen.de/MATHEMATIK/DMO/mitarbeiter/andres.html (Stephan Dominique Andres).

Let $\omega(G)$ be the clique number of a graph G. G is called *B-perfect* if, for any induced subgraph H of G, $\chi_B(H) = \omega(H)$. Analogously, we define *A-perfect* with respect to the game A and *g-perfect* with respect to Bodlaender's game. These concepts were introduced in [1] and are game-theoretic analoga of *perfect* graphs which are those graphs in which, for any induced subgraph H, the clique number equals the chromatic number $\chi(H)$. For any graph H,

$$\omega(H) \leq \chi(H) \leq \chi_A(H) \leq \chi_g(H) \leq \chi_B(H).$$

In particular, B-perfect graphs are g-perfect, g-perfect graphs are A-perfect, and A-perfect graphs are perfect. We consider the problem of characterizing these classes of graphs. The (probably most difficult) case of perfect graphs has been solved by the Strong Perfect Graph Theorem [3]:

Theorem 1 (Chudnovsky, Robertson, Seymour, Thomas (2006)) *A graph is perfect if, and only if, it does neither contain an odd hole nor an odd antihole as induced subgraph.*

In this talk we will characterize B-perfect graphs.

2 Main result

Theorem 2 *Let G be a graph. Then the following conditions are equivalent:*

(i) *G is B-perfect.*
(ii) *G does neither contain a C_4, nor a P_4, nor a split 3-star, nor a double fan as induced subgraph (see Fig. 1).*
(iii) *For every (nonempty) component H of G, there is $k \geq 0$, so that*

$$H = K_1 \vee (H_0 \cup H_1 \cup \ldots \cup H_k),$$

where the H_i are complete graphs for $i \geq 1$, and H_0 is either empty or there are $p, q, r \in \mathbb{N}$, so that $H_0 = K_p \vee K_r \vee K_q$ (see Fig. 2).

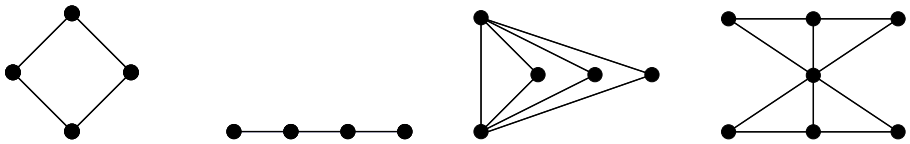


Fig. 1. 4 forbidden induced subgraphs for B-perfect graphs

PROOF. (i) $\Longrightarrow$ (ii): Winning strategies for Bob with ≤ 2 colors on C_4 resp. P_4 resp. with ≤ 3 colors on the split 3-star resp. the double fan are obvious.

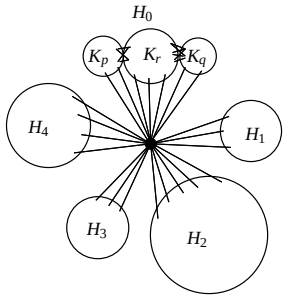


Fig. 2. Structure of a component according to (iii)

(iii) $\Longrightarrow$ (i): We describe a winning strategy for Alice with $\omega(G)$ colors on a graph G as in (iii). This is sufficient since every induced subgraph of G is of the same type as described in (iii). For $H_0 = K_p \vee K_r \vee K_q$ let the K_p and the K_q be the *ears*. Alice always responds to Bob's moves in the same component H (if Bob passes, in an arbitrary component). As long as Bob does not play in an ear, Alice does not play in an ear; she first colors the universal vertex of H. If Bob plays in an ear K_p, Alice colors a vertex in the corresponding ear K_q with the same color (in case there is no uncolored vertex she uses the strategy described before). If Alice is forced to start coloring an ear, then all non-ear-vertices are colored, so a coloring of the ears is possible without creating danger for a non-ear-vertex.

(ii) $\Longrightarrow$ (iii): We examine the structure of a graph G without induced P_4, C_4, split 3-star, double fan. Let H be a component of G. We use the following lemma of Wolk [5].

Lemma 3 (Wolk (1965)) *A connected graph without induced C_4 and P_4 (a so-called* trivially perfect graph [4]*) has a universal vertex.*

So, H has a universal vertex v. Let $H_0, \ldots, H_n$ be the components of $H \setminus v$. Using the fact that H does not contain a double fan we can prove the following

Claim 4 *At most one of the H_i is not complete.*

Let H_0 be the (only) component of $H \setminus v$ which is not complete. Let K be the largest clique of H_0. We are done if we show:

Claim 5 *$H_0 \setminus K$ induces a clique.*

Claim 6 *$H_0 \setminus K$ induces a module of H_0 (i.e. if $x \in K$, either x is adjacent to all $y \in H_0 \setminus K$ or to none.)*

The proof of Claim 5 uses Lemma 3 again and the fact that H does neither contain a split 3-star nor a P_4. The proof of Claim 6 uses Claim 5 and the fact that H does neither contain a P_4 nor a C_4.

Claim 4, Claim 5 and Claim 6 together imply that H has the structure as described in (iii): $H_0 \setminus K$ corresponds to the K_p, its neighbors correspond to the K_r, and the rest of H_0 corresponds to the K_q. This completes the proof of Theorem 2. □

3 Open problems

Problem 7 *Characterize A-perfect graphs by forbidden induced subgraphs.*

Problem 8 *Characterize g-perfect graphs by forbidden induced subgraphs.*

We discuss some partial results concerning these problems. The following are already known, cf. [1]:

Theorem 9 *A triangle-free graph G is A-perfect if, and only if, every component of G is either K_1 or $K_{m,n}$ or $K_{m,n} - e$, where e is an edge.*

Theorem 10 *Complements of bipartite graphs are A-perfect.*

References

[1] Andres, S. D., *Game-perfect graphs*, Math. Meth. Oper. Res. **69** (2009), 235–250

[2] Bodlaender, H. L., *On the complexity of some coloring games*, Int. J. Found. Comput. Sci. **2**, no.2 (1991), 133–147

[3] Chudnovsky, M., N. Robertson, P. Seymour, and R. Thomas, *The strong perfect graph theorem*, Ann. Math. **164** (2006), 51–229

[4] Golumbic, M. C., *Trivially perfect graphs*, Discrete Math. **24** (1978), 105–107

[5] Wolk, E. S., *A note on "the comparability graph of a tree"*, Proc. Am. Math. Soc. **16** (1965), 17–20

Linear Visualization of a Road Coloring Algorithm

A.N. Trahtman, T. Bauer and N. Cohen

Bar-Ilan University, Dep. of Math., 52900, Ramat Gan, Israel

Abstract

The visualization has become essential in many application areas. The finite graphs and automata undoubtedly belong to such areas. A problem of a visual image of a directed finite graph has appeared in the study of the *road coloring conjecture*.

Given a finite directed graph, a coloring of its edges turns the graph into finite-state automaton. The visual perception of the structure properties of automata is an important goal. A synchronizing word of a deterministic automaton is a word in the alphabet of colors of its edges that maps the automaton to a single state. A coloring of edges of a directed graph is synchronizing if the coloring turns the graph into a deterministic finite automaton possessing a synchronizing word.

The road coloring conjecture [1], [2] was stated about forty yeas ago for a complete strongly connected directed finite graph with constant outdegree of all its vertices where the greatest common divisor (gcd) of lengths of all its cycles is one. The edges of the graph being unlabelled, the task is to find a labelling that turns the graph into a deterministic finite automaton possessing a synchronizing word. Such graph has according to the conjecture a synchronizing coloring.

The problem belonged to the most fascinating problems in the theory of finite automata [9], [4] and was mentioned in the popular Internet Encyclopedia "Wikipedia" on the list of the most interesting unsolved problems in mathematics. The positive solution of the road coloring problem [13] is a basis of a polynomial-time implemented algorithm of $O(n^3)$ complexity in the worst case.

The realization of the considered algorithm is demonstrated by a high-speed visualization program. The visibility of inner structure of a digraph without doubt is a matter of interest not only for road coloring, the range of the application may be significantly wider.

Crucial role in the visualization plays for us the correspondence of the layout to the human intuition, the perception of the structure properties of the graph and the rapidity of the appearance of the image. We use for this aim some known approaches [11], [14] together with some new productive ideas. Our algorithm for the visualization is linear in the size of the automaton. This algorithm not complicated at first sight successfully solves a whole series of tasks of the disposal of the objects.

The visualization of the transition graph of the automaton is a help tool of the

study of the automata. Thus the linearity of the algorithm is comfortably and important. Both the road coloring algorithm and the visualization algorithm are implemented in the package TESTAS (www.cs.biu.ac.il/∼trakht/syn.html).

As usual, we regard a directed graph with colors assigned to its edges as a finite automaton, whose input alphabet consists of these colors. The graph is called *transition graph* of the automaton.

An automaton is *deterministic* if no state has two outgoing edges of the same color. In *complete* automaton each state has outgoing edges of any color.

Let $|P|$ denote the size of the subset P of states from an automaton (of vertices from a graph).

Let Ps be the set of states $\mathbf{p}s$ for $\mathbf{p} \in P$ $s \in \Sigma^+$. For the transition graph Γ of an automaton let Γs denote the map of the set of states of the automaton.

A word $s \in \Sigma^+$ is called a *k-synchronizing* word of the automaton with transition graph Γ if both $|\Gamma s| = k$ and for all words $t \in \Sigma^*$ holds $|\Gamma t| \geq k$.

A pair of distinct states $\mathbf{p}, \mathbf{q}$ of an automaton (of vertices of the transition graph) will be called *synchronizing* if $\mathbf{p}s = \mathbf{q}s$ for some $s \in \Sigma^+$.

A synchronizing pair of states $\mathbf{p}$, $\mathbf{q}$ of an automaton is called *stable* if for every word u the pair $\mathbf{p}u, \mathbf{q}u$ is also synchronizing [4], [**?**].

We call the set of all outgoing edges of a vertex a *bunch* if all these edges are incoming edges of only one vertex.

Imagine a map with roads which are colored in such a way that a fixed sequence of colors, called a synchronizing sequence, leads to a fixed place whatever is the starting point. Finding such a coloring is called em road coloring problem. The roads of the map are considered as edges of a directed graph. The visual presentation of a road coloring algorithm is essentially based on the paths of the graph. The paths must be visible as well as cycles, bunches and other structure components of the graph. In particular, the notion of the bunch according to the following lemma plays some role in the road coloring algorithm.

Lemma 1 *[13] If some vertex of graph* Γ *has two incoming bunches then there exists a stable pair by any coloring.*

The role of the length of a path is also important.

Lemma 2 *[13] Let any vertex of the graph* Γ *have no two incoming bunches. Then a subgraph of* Γ *of some color has maximal subtree.*

A crucial role in the visualization plays in our opinion the correspondence of the layout to the human intuition, the perception of the structure properties of the graph and the rapidity of the appearance of the image. The automatically drawn graphical image must resemble the last one of a human being. The considered visualization is a help tool for any program dealing with transition graph of DFA and in particular for the road coloring algorithm.

Our main objective is the visual representation of the transition graph of a deterministic finite automaton based on the structure properties of the graph. Any deterministic finite automaton is accepted by the algorithm.

Among the important visual properties of a graph one can mention paths, cycles, strongly connected components, cliques, bunches etc. These important properties reflect the inner structure of the digraph. The special significance plays here the strongly connected components (SCC). Thus our first step is the eduction and selection of the SCC. We choose to place SCC according to a cyclic layout[11], [14]. According to approach the vertices are placed at the periphery of a circle. Our modification of the approach considered two levels of circles, the first level consists of strongly connected components, the second level corresponds to the whole graph with SCC at the periphery of the circle. The visual placement is based on the structure of the graph considered as a union of the set of strongly connected components.

It is clear that the curve edges (used, for instance, in the package GraphViz [6], [10]) hinder to recognize the cycles and paths. Therefore, we use only direct and, hopefully, short edges. We have changed some priorities of the layout and, in particular, eliminate the goal of reducing the number of intersections of the edges as it was an important aim in some algorithms [10]. The intersections of the edges are even not considered in our algorithm. This approach gives us an opportunity to simplify essentially the algorithm and to reduce its complexity. Our main intent is only not to stir by the intersections of the edges to conceive the structure of the graph. The intersections are placed in our algorithm far from the vertices due to the cyclic layout [14], [11] we use. The area of vertices differs of the area of the majority of intersections.

The problem of the placing of the labels near corresponding edges is sometimes very complicated and frequently the connection between edge and its label is not clear. Our solution is to use colors on the edges instead of labels and exclude the placing of labels.

The quick linear algorithm for finding SCC [3] is implemented in the program. The vertices of every SCC belong to a cycle in the graph layout. So strongly connected components can be easily recognized by observer. All SCC are placed on the periphery of a big circle. So the pictorial diagram demonstrates the structure of the graph and the visualization can be considered as a kind of structure visualization.

The periphery of a circle of SCC is the most desirable area for placing the edges because the edges in this case are short. We choose the order of the vertices of the SCC on the circle according to this purpose. The length of some edges can be reduced in a such way. It also helps an observer to recognize paths and cycles on the screen.

The linearity of the algorithm ensures the momentary appearance of the layout. It is favorably also for educational purposes because the road coloring conjecture can be stated in simple terms and initial explorations can be done immediately. It can be understood by any student with a little experience in the graph theory. "The Road Coloring Conjecture makes a nice supplement to any discrete mathematics course" [7].

The complexity of the algorithm describes the following

Lemma 3 *The time and space complexity of the visualization algorithm described above is linear in the sum of states and edges of the transition graph of automaton.*

References

[1] R.L. Adler, L.W. Goodwyn, B. Weiss. Equivalence of topological Markov shifts, Israel J. of Math. 27(1977), 49-63.

[2] R.L. Adler, B. Weiss. Similarity of automorphisms of the torus, Memoirs of the Amer. Math. Soc., Providence, RI, 98(1970).

[3] A. Aho, J. Hopcroft, J. Ulman. *The Design and Analisys of Computer Algorithms*, 1974, Addison-Wesley.

[4] K. Culik II, J. Karhumaki, J. Kari. A note on synchronized automata and Road Coloring Problem, Lect. Notes in Comput. Sci., 2295, 2002, 175-185.

[5] J. Černy. Poznamka k homogenym eksperimentom s konechnymi automatami. Math.-Fyz. Čas., 14, 1964, 208-215.

[6] J. Ellson, E. Gansner, L. Koutsofios, et al.. GraphViz - open source graph drawing tools, Graph Drawing, 2265, 2002, 483-484.

[7] J. V. Rauff, Way back from anywhere: exploring the road coloring conjecture. Math. and Comput. Education. 01, 2009.

[8] A. Roman. Synchronizing finite automata with short reset words. Applied Math. and Comput. 1, 209, 2009, 125-136.

[9] A. Mateescu, A.Salomaa, Many-valued truth function. Černy conjuncture and road coloring, EATCS Bulletin, 68(1999) 134-150.

[10] M.Simonato, 2004, *An Introduction to GraphViz*.

[11] JM. Six, IG. Tollis, A framework for user-grouped circular drawings Lect. Notes in Comp. Sci., 1731, 1999, 107-116.

[12] A.N. Trahtman, Notable trends concerning the synchronization of graphs and automata. El. Notes in Discr. Math., 25, 2006, 173-175

[13] A.N. Trahtman. Synchronizing Road Coloring. 5-th IFIP WCC-TCS, Springer, 273, 2008, 43-53.

[14] R. Wiese, M. Eiglsperger, M. Kaufmann. 2002, A framework for circular drawings of network

Exact Bipartite Crossing Minimization under Tree Constraints ⋆

Frank Baumann [a] Christoph Buchheim [a] Frauke Liers [b]

[a] *Technische Universität Dortmund, Fakultät für Mathematik, Vogelpothsweg 87, 44227 Dortmund, Germany*

[b] *Universität zu Köln, Institut für Informatik, Pohligstraße 1, 50969 Köln, Germany*

Key words: tanglegram, crossing minimization, quadratic programming

1 Abstract

A *tanglegram* consists of a pair of (not necessarily binary) trees. Additional edges, called *tangles*, may connect the leaves of the first with those of the second tree. The task is to draw a tanglegram with a minimum number of tangle crossings while making sure that the trees are drawn crossing-free. This problem has relevant applications in computational biology, e.g., for the comparison of phylogenetic trees. Most existing approaches are only applicable for binary trees. In this work, we show that the problem can be formulated as a quadratic linear ordering problem (QLO) with side constraints. In [1] it was shown that, appropriately reformulated, the QLO polytope is a face of some cut polytope. It turns out that the additional side constraints do not destroy this property. Therefore, any polyhedral approach to max-cut can be used in our context. We show that our approach is very efficient in practice for both random and real-world binary as well as general tanglegrams.

2 Introduction

The task of drawing tanglegrams arises in several relevant applications, e.g., in computational biology for the comparison of phylogenetic trees. Moreover,

⋆ Financial support from the German Science Foundation (DFG) is acknowledged under contracts Bu 2313/1–1 and Li 1675/1–1.

tanglegrams occur when analyzing software projects in which a tree represents package, class and method hierarchies. This application yields tanglegrams on trees that are not binary in general [5]. Most of the literature is concerned with the case of binary trees and leaves that are in one-to-one correspondence. [4] showed the NP-hardness of tanglegram layout, even in the case of binary trees. Our approach, to be explained below, generalizes the exact integer-programming (IP) approach of [5] for binary tanglegrams. In the latter, minimizing the number of tangle crossings reduces to solving an unconstrained quadratic binary optimization problem, which is well-known to be equivalent to a maximum cut problem in some associated graph with an additional node [2]. In an undirected graph $G = (V, E)$, the cut $\delta(W)$ induced by a set $W \subseteq V$ is defined as the set of edges (u, v) such that $u \in W$ and $v \notin W$. If edge weights are given, the weight of a cut is the total weight of edges in the cut. Now the maximum cut problem asks for a cut of maximal weight or cardinality. While in the recent paper by [7] the focus is on binary instances, a fixed-parameter algorithm for general tanglegram instances is presented. According to our knowledge, this is the only algorithm that could deal with non-binary trees; however, no implementation or running times are provided making it impossible to evaluate its practical performance.

3 An Exact Model for General Tanglegrams

Let $G = (V_1 \cup V_2, E)$ be a bipartite graph. The task is to draw G with straight line edges. The nodes in V_1 and V_2 have to be placed on two parallel lines H_1 and H_2 such that the number of edge crossings is minimal. Assume for a moment that the nodes on the H_1 are fixed, and only the nodes on H_2 are permuted. For each pair of nodes on H_2, we introduce a variable x_{uv} such that $x_{uv} = 1$ if u is drawn to the left of v and $x_{uv} = 0$ otherwise. For edges (i, k) and (j, l) with $i, j \in H_1$ and $k, l \in H_2$, such that i is left of j, a crossing exists if and only if l is left of k. We thus have to punish x_{lk} in the objective function. The task of minimizing the number of crossings is now equivalent to determining a minimum linear ordering on the nodes of H_2. Note that bipartite crossing minimization with one fixed layer is already NP-hard [3]. If the nodes on both layers are allowed to permute, the problem can be modeled as a quadratic optimization problem over linear ordering variables. The quadratic linear ordering problem (QLO) is

$$
\begin{array}{lll}
 & \min & \sum_{(i,j,k,l)\in I} c_{ijkl} x_{ij} x_{kl} \\
(QLO) & \text{s.t.} & x \in P_{LO} \\
 & & x_{ij} \in \{0,1\} \; \forall (i,j) \in J
\end{array}
$$

where P_{LO} is the linear ordering polytope and

$$I = \{(i,j,k,l) \mid i,j \in H_1,\ i<j,\ \text{and } k,l \in H_2,\ k<l\}$$
$$J = \{(i,j) \mid i,j \in H_1 \text{ or } i,j \in H_2, i<j\}$$

We replace each product $x_{ij}x_{kl}$ by a new binary variable y_{ijkl} and add the linearization constraints $y_{ijkl} \leq x_{ij}, y_{ijkl} \leq x_{kl}, y_{ijkl} \geq x_{ij} + x_{kl} - 1$. We call the resulting linearized problem (LQLO). In [1] it was shown that a 0/1 vector (x,y) satisfying $y_{ijkl} = x_{ij}x_{kl}$ is feasible for (LQLO) if and only if

$$x_{ik} - y_{ijik} - y_{ikjk} + y_{ijjk} = 0 \ \forall (i,j,k,l) \in I, \tag{1}$$

which is a quadratic reformulation of the constraints defining P_{LO}. Note that (LQLO) is a quadratic binary optimization problem where the feasible solutions need to satisfy further side constraints. As unconstrained binary quadratic optimization is equivalent to the maximum cut problem [2], the task is to intersect a cut polytope with a set of hyperplanes. [1] showed that the hyperplanes (1) cut out faces of the cut polytope.

Let us consider a triple of leaves a, b, c in one of the trees. In case all pairwise lowest common ancestors coincide, all relative orderings between a, b, and c are feasible. However, if the lowest common ancestor of, say, a and b is on a lower level than that of, say, a and c, then c must not be placed between a and b; see Figure 1.

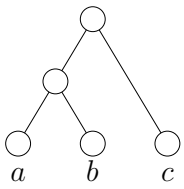


Fig. 1. Leaf c is not allowed to lie between a and b.

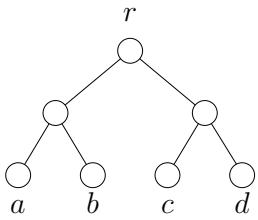


Fig. 2. Variables x_{ac} and x_{bd} can be identified.

Therefore, we derive a betweenness restriction for every triple of leaves such that two of the leaf pairs have different lowest common ancestors. Each restriction of the form 'c cannot be placed between a and b' can be written as $x_{ac}x_{cb} = 0$ and $x_{bc}x_{ca} = 0$, i.e., $y_{accb} = 0$ and $y_{cabc} = 0$. As mentioned above, the polytope corresponding to (LQLO) is isomorphic to a face of a cut polytope [1]. Since all y-variables are binary, the additional betweenness constraints are always face-inducing for (LQLO).

Theorem 1 *The problem of drawing tanglegrams with a minimum number of edge crossings can be solved by optimizing over a face of a suitable cut polytope.*

4 Results

We implemented the above model. The naive approach is to solve the linearized model (LQLO) using a standard integer programming solver. A more advanced approach is to solve the quadratic reformulation (1), using separation of cutting planes for max-cut, both in the context of integer and semidefinite programming. For the IP-based methods, we used CPLEX 11.2, whereas for the SDP approaches, we used the bundle method by [6]. For random and real-world binary as well as general trees with low tangle density, the SDP approach usually needs considerably more time than the IP-based methods. Furthermore, memory requirements strongly increase with system size. On average, the fastest approaches are the pure standard linearization *IP* and the quadratic reformulation *QP*. In fact, we can optimize tanglegrams with more than 500 leaves in each tree which is the range of real-world instance sizes. For big enough tangle density the SDP approach usually outperforms the IP approaches. Memory requirements, however, usually prohibit solving binary instances with more than 500 leaf nodes and tangle density of 1%. For general trees, best performance is often found for the quadratic reformulation. These non-binary instances could not be solved before by any other exact method.

References

[1] C. Buchheim and A. Wiegele, and L. Zheng. Exact Algorithms for the Quadratic Linear Ordering Problem. *INFORMS Journal on Computing*. To appear.

[2] C. De Simone. The Cut Polytope and the Boolean Quadric Polytope. *Discrete Mathematics*, 79; 71–75, 1989.

[3] P. Eades and N. C. Wormald. Edge crossings in drawing bipartite graphs. *Algorithmica*, 11; 379–403, 1994.

[4] H. Fernau and M. Kaufmann, and M. Poths. Comparing trees via crossing minimization. *Journal of Computer and System Sciences*. In Press.

[5] M. Nöllenburg, M. Völker, A. Wolff, and D. Holten. Drawing Binary Tanglegrams: An Experimental Evaluation. In *Proc. of the Workshop on Algorithm Engineering and Experiments, ALENEX 2009*, pages 106-119, SIAM, 2009.

[6] F. Rendl, G. Rinaldi, and A. Wiegele. A Branch and Bound Algorithm for Max-Cut Based on Combining Semidefinite and Polyhedral Relaxations. In M. Fischetti and D. P. Williamson, editors, *IPCO 2007*, volume 4513 of *Lecture Notes in Computer Science*, pages 295-309. Springer, 2007.

[7] B. Venkatachalam, J. Apple, K. St. John, and D. Gusfield. Untangling Tanglegrams: Comparing Trees by Their Drawings. *Bioinformatics Research and Applications*, pages 88–99, 2009.

On the convergence of feasibility based bounds tightening

Pietro Belotti

Dept. of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA

Sonia Cafieri

Dept. Mathématiques et Informatique, ENAC, 7 av. E. Belin, 31055 Toulouse, France

Jon Lee

Dept. of Mathematical Sciences, IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, USA

Leo Liberti *

LIX, École Polytechnique, 91128 Palaiseau, France

Key words: spatial Branch-and-Bound, range reduction, lattice, fixed point, MINLP, global optimization, constraint programming.

1 Introduction

Global Optimization and Mixed-Integer Nonlinear Programming problems such as $\min\{f(x) \mid g^L \leq g(x) \leq g^U \wedge x^L \leq x \leq x^U \wedge \forall j \in Z\ (x_j \in \mathbb{Z})\}$, where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $g^L, g^U \in \mathbb{R}^m$, $x^L, x, x^U \in \mathbb{R}^n$ and $Z \subseteq \{1, \ldots, n\}$,

⋆ Financial support by grants System@tic "EDONA", ANR 07-JCJC-0151 "Ars", ANR 08-SEGI-023 "Asopt", Digiteo Emergence "Paso" is gratefully acknowledged.
* Corresponding author.
Email addresses: belotti@lehigh.edu (Pietro Belotti), sonia.cafieri@enac.fr (Sonia Cafieri), jonlee@us.ibm.com (Jon Lee), liberti@lix.polytechnique.fr (Leo Liberti).

are usually solved to ε-guaranteed approximation by the spatial Branch-and-Bound (sBB) algorithm [2], a variant of the usual Branch-and-Bound for dealing with nonlinear, possibly nonconvex f, g. Since the gap between the original problem P and its convex relaxation $\bar{P}$ is due both to integral variable restrictions being lifted as well as nonconvex functions being replaced by a convex relaxation, sBB is able to branch at continuous variables as well as integer ones. If $\bar{x}$ solves $\bar{P}$, the standard disjunction used at a node in the sBB search tree is $x_j \leq \bar{x}_j \vee x_j \geq \bar{x}_j$, the more usual one $x_j \leq \lfloor \bar{x}_j \rfloor \vee x_j \geq \lceil \bar{x}_j \rceil$ being used only if $j \in Z$.

At any sBB node, it is important to make sure that the variable ranges $x^L \leq x \leq x^U$ for that node are as tight as the constraint restrictions $g^L \leq g(x) \leq g^U$ allow. Letting $\mathcal{F}(P)$ be the feasible region of P, we would wish to replace $X^0 = [x^L, x^U]$ with $\tilde{X} = [\tilde{x}^L, \tilde{x}^U]$ such that $\tilde{x}_i^L = \min_{x \in \mathcal{F}(P)} x_i$ and $\tilde{x}_i^U = \max_{x \in \mathcal{F}(P)} x_i$ for all $i \leq n$. Since these $2n$ problems are as hard as P, we relax these requirements. There are two standards relaxations: Optimization Based Bounds Tightening (OBBT) [3,2] and Feasibility Based Bounds Tightening (FBBT) [1,2]. The former consists in replacing $\mathcal{F}(P)$ with $\mathcal{F}(\bar{P})$. The latter, whose convergence properties are the object of this paper, is also known in Constraints Programming as a range reduction device. FBBT relies on interval arithmetic to derive the constraint ranges $\bar{G} = [\bar{g}^L, \bar{g}^U]$ implied by the variable ranges X^0 at a given sBB node; if $\bar{G} \supsetneq G^0 = [g^L, g^U]$, FBBT uses inverse interval arithmetic to propagate G^0 back to tightened variable ranges X'. This basic FBBT step is iterated until convergence, generating an interval sequence $X^0, X^1, \ldots$. Since OBBT is usually slower than FBBT, OBBT is only applied at the root sBB node and FBBT is applied at each node. Furthermore, to simplify inverse interval arithmetic, FBBT often only considers a subset of linear constraints in g. We shall therefore make the assumption — without excessive loss of generality — that the symbol g denotes the linear constraints of P, which we denote as $g^L \leq Ax \leq g^U$ for some $m \times n$ matrix A.

The main trouble with FBBT is that its worst-case running time is infinite in the size of its input (m, n, X^0, A, G^0). For example, on the instance $(2, 2, ([0,1],[0,1]), \left(\begin{smallmatrix} a & -1 \\ 1 & -a \end{smallmatrix}\right), ([0,0],[0,0]))$, FBBT yields the infinite interval sequence $([0, 1/a^{2k-1}], [0, 1/a^{2k}])$ whenever $a > 1$. Enforcing finite convergence by terminating at the first iteration k such that $\mathscr{L}(X^{k-1} \triangle X^k) \leq \varepsilon$, where $\varepsilon > 0$ is given and $\mathscr{L}$ is the Lebesgue measure in $\mathbb{R}$, yields a finite but unbounded worst-case time complexity: given a fixed iteration bound K there are always instances where the FBBT takes longer than K iterations to reach the ε termination condition (it suffices to decrease the value a appropriately). In practice, such occurrences are far from rare, specially when the coefficients of Ax are obtained by previous floating point operations, which might cause a small but positive $|a_i - a_j|$ even if a_i, a_j are supposed to be equal.

In this paper we propose a new method for finding the limit point of the FBBT sequence in polynomial time, based solving a Linear Program (LP) modelling the greatest fixed point of the FBBT in the interval lattice.

2 Fixed points in the interval lattice

A *lattice* is a set Λ partially ordered by the relation $\sqsubseteq$ endowed with two operations $\sqcup$ (*join*), $\sqcap$ (*meet*) such that $x \sqsubseteq x \sqcup y$, $y \sqsubseteq x \sqcup y$ and $x \sqcap y \sqsubseteq x$, $x \sqcap y \sqsubseteq y$. A lattice is *complete* if there exist elements $\bot, \top$ such that $\bot \sqsubseteq x \sqsubseteq \top$ for all $x \in \Lambda$. An operator $F : \Lambda \to \Lambda$ is *monotone* if $x \sqsubseteq y$ implies $F(x) \sqsubseteq f(y)$ and *deflationary* if $F(x) \sqsubseteq x$ for all $x \in \Lambda$. The set of all real intervals forms a lattice $\mathscr{I}$ under set inclusion $\subseteq$, with set intersection $\cap$ as meet and interval union (smallest interval including two intervals) $\cup$ as join. The lattice structure is extended to arrays of intervals in the standard way. In the rest of the paper, we let X be the interval vector $(X_1, \ldots, X_n) \in \mathscr{I}^n$ and $G = (G_1, \ldots, G_m) \in \mathscr{I}^m$.

FBBT consists of two phases: upwards and downwards propagation. We define $\mathsf{up} : \mathscr{I}^n \to \mathscr{I}^m$ and $\mathsf{down} : \mathscr{I}^m \to \mathscr{I}^n$ as:

$$\mathsf{up}(X) = (G_i^0 \cap \sum_{j \le n} a_{ij} X_j \mid i \le m) \tag{1}$$

$$\mathsf{down}(G) = \bigcap_{i \le m} (X_j \cap \frac{1}{a_{ij}}(G_i - \sum_{\ell \ne j} a_{i\ell} X_\ell) \mid j \le n), \tag{2}$$

where all arithmetic operators have interval semantics [5]. We now define the FBBT iteration as an operator $\mathsf{fbbt} : \mathscr{I}^n \to \mathscr{I}^n$ by $\mathsf{fbbt}(X) = \mathsf{down}(\mathsf{up}(X \cap X^0))$ (we remark that our definition of fbbt depends on the initial interval vector X^0). Because all linear interval arithmetic and lattice operators are monotone [5] and the composition of monotone operators is monotone [6], fbbt is a monotone operator. Furthermore, because of the intersections, intervals are changed only if the up and down actions make them smaller. Again, the composition of deflationary operators is deflationary [6]: hence fbbt is deflationary. By applying Thm. 12.9 in [6] to the dual lattice obtained by inverting $\top$ and $\bot$, $\sqsubseteq$ and $\sqsupseteq$, meet and join, we have that the sequence $(\mathsf{fbbt}^k(X) \mid k \ge 0)$ converges to the greatest fixed point (gfp) of fbbt, i.e. the largest (in the lattice order) interval vector X such that $\mathsf{fbbt}(X) = X$. In other words $\mathrm{gfp}(\mathsf{fbbt}) = \sup\{X \mid X = \mathsf{fbbt}(X)\}$. By Tarski's Fixed Point Theorem [7], equality can be replaced with $\subseteq$. Furthermore, the operator $|\cdot| : \mathscr{I}^n \to \mathbb{R}$ given by $|X| = \sum_{j \le n}(x_j^U - x_j^L)$ is monotone with the lattice order; since the lattice is complete, we obtain:

$$\mathrm{gfp}(\mathsf{fbbt}) = \mathrm{argmax}\{|X| \mid X \subseteq \mathsf{fbbt}(X)\}, \tag{3}$$

which we state as the following "interval linear problem" with parameters

(m, n, X^0, A, G^0) and interval decision variable arrays X, G:

$$\max\{|X| \mid X \subseteq X^0 \wedge G \subseteq \mathsf{up}(X) \wedge X \subseteq \mathsf{down}(G)\}, \tag{4}$$

It is possible to write an ordinary LP whose optimal solution is the same as (4).

3 Computational validation

By way of preliminary computational validation of our approach, we solved four significant instances using CPLEX 11.0 [4] on an Intel Core 2 Duo 1.4GHz with 3GB RAM running Linux. For each instance we record the gfp, the seconds of user CPU time to solution with either method, the absolute error $E = \sum_{j \leq n} \mathscr{L}(X^* \triangle \mathrm{gfp}(\mathsf{fbbt}))$ where X^* is the output of either method ($\varepsilon_{\mathrm{FBBT}} = 10^{-6}$), and the relative error R of the FBBT obtained by letting it run only for as long as the LP method takes to converge to the (precise) gfp's.

Instance	$x_1 - 1.01x_2 = 0$ $-1.01x_1 + x_2 = 0$ $x_1, x_2 \in [-1, 1]$	$x_1 - 1.01x_2 = 1$ $-1.01x_1 + x_2 = -1.01$ $x_1, x_2 \in [-1, 1]$	$x_1 - 1.01x_2 = 0$ $-1.01x_1 + x_2 = 0$ $x_3 + 100x_1 \geq -100$ $x_1, x_2 \in [-1, 1],\ x_3 \in [-10, 1]$	$x_1 - 1.01x_2 = -1.01$ $-1.01x_1 + x_2 = 1$ $x_3 + 100x_1 \geq -100$ $x_1, x_2 \in [-1, 1],\ x_3 \in [-10, 1]$
gfp	([0,0],[0,0])	([1,1],[0,0])	([0,0],[0,0],[-10,0])	([0,0],[1,1],[-10,0])
$\mathrm{CPU}_{\mathrm{FBBT}}$	0.04	0.04	0.06	0.06
$\mathbf{CPU_{LP}}$	**0**	**0**	**0.004**	**0.004**
E_{FBBT}	1e-4	5e-5	1.5e-6	5e-5
$E_{\mathbf{LP}}$	**0**	**0**	**0**	**0**
R_{FBBT}	2.29	1.47	2.33	1.21

References

[1] D.E. Andersen and K.D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71:221–245, 1995.

[2] 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.

[3] A. Caprara and M. Locatelli. Global optimization problems and domain reduction strategies. *Mathematical Programming*, to appear.

[4] ILOG. *ILOG CPLEX 11.0 User's Manual*. ILOG S.A., Gentilly, France, 2008.

[5] R.E. Moore, R.B. Kearfott, and M.J. Cloud. *Introduction to Interval Analysis*. SIAM, Philadelphia, 2009.

[6] S. Roman. *Lattices and Ordered Sets*. Springer, New York, 2008.

[7] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

Branch-and-price for the multi-depot pickup and delivery problem with heterogeneous fleet and soft time windows

Andrea Bettinelli, Alberto Ceselli, Giovanni Righini

Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano, via Bramante 65, 26013 Crema, Italy

Key words: branch-and-price, pickup and delivery, soft time windows

1 Introduction

The multi-depot pickup and delivery problem with heterogeneous fleet and soft time windows (MDPDPHSTW) requires to find a minimum cost routing for a fleet vehicles with different capacities and based at different depots, satisfying a given set of customers. A customer request is associated with two locations: a source where a certain demand must be picked up and a destination where this demand must be delivered. Each route must satisfy pairing constraints (pickup and delivery of a customer must both be visited in the same route) and precedence constraints (the delivery location must be visited after the corresponding pickup location). Further, each pickup and delivery location has a time window for the service that can be violated at the cost of a linear penalty. This problem has application in various scenarios, such as urban courier services, less-than-truckload transportation, door to door transportation services.

The problem has been widely studied in the version with hard time windows constraints (PDPTW). Several exact approaches based on branch-and-cut [1,8] and branch-and-price [3,10,7] have been proposed. For comprehensive reviews on routing problems involving pickup and delivery, the reader is referred to the works of Savelsberg and Sol [9], Cordeau et al. [2] and Parragh et al. [5]. The soft time windows case was address in an early work by Sexton and Choi [11]. They developed a heuristic algorithm based on Benders' decomposition.

Email address: {name.surname}@unimi.it (Andrea Bettinelli, Alberto Ceselli, Giovanni Righini).

Recently, Liberatore et al. [4] and Qureshi et al. [6] proposed branch-and-price approaches for the vehicle routing problem with soft and semi-soft time windows.

In this work we propose a branch-and-price algorithm for the MDPDPHSTW which combines ideas proposed in [7] with bidirectional label extension and decremental state space relaxation, and uses a modified version of the algorithm developed by Liberatore et al. [4] to handle soft time windows.

2 Problem description

Given a set $\mathcal{K}$ of vehicle types, a set $\mathcal{H}$ of depots and a set $\mathcal{N}$ of customer requests, the problem can be defined on a directed graph $G = (V, A)$, where V contains a vertex for each depot $h \in \mathcal{H}$ and for the pickup and delivery locations of each customer $i \in \mathcal{N}$. Non negative weights d_{ij} and t_{ij} are associated with each arc $(i, j) \in A$; they represent the transportation cost and the traveling time respectively. A service time s_j and a time window $[a_j, b_j]$ are associated to each vertex $j \in V$; if the service at location j starts inside its time window no penalty is incurred, otherwise a linear penalty, proportional to the anticipation or delay through non-negative coefficients α_j and β_j respectively, has to be paid. If we call T_j the starting time of service at location j, the penalty term $\pi(T_j)$ is defined as follows:

$$\pi(T_j) = \begin{cases} \alpha_j(a_j - T_j) & \text{if } T_j < a_j \\ 0 & \text{if } a_j \leq T_j \leq b_j \\ \beta_j(T_j - b_j) & \text{if } T_j > b_j. \end{cases}$$

Each vehicle type $k \in \mathcal{K}$ has given capacity w_k and fixed cost f_k. A limited number of vehicles is available: at most u_{hk} vehicles of type $k \in \mathcal{K}$ can be based at depot $h \in \mathcal{H}$.

The objective is to minimize the sum of vehicles fixed costs and routing costs (including penalties), satisfying the following conditions: (a) all customers is served, (b) each customer is visited by only one vehicle, (c) each route begins at a depot and ends at the same depot, (d) the capacity of the associated vehicle is not exceeded, (e) pickup and delivery of a customer are performed in the same route, (f) the pickup vertices are visited earlier than the corresponding delivery vertices, (g) the number of available vehicles of each type for each depot is not exceeded.

3 Formulation

We consider a set covering formulation of the MDPDPHSTW. We say that a route is feasible if it satisfies conditions (b), (c), (d), (e) and (f). Let Ω_{hk} be the set of all feasible routes using a vehicle of type $k \in \mathcal{K}$ from depot $h \in \mathcal{H}$. We associate a binary variable x_r with each feasible route, which takes value 1 if and only if route r is selected. Let a_{ir} be a binary coefficient with value 1 if and only if customer i is visited by route r. Let c_r be the cost of route r; it is equal to the sum of the vehicle fixed cost f_k and the routing costs. With these definitions we obtain the following integer linear programming model:

$$\min \sum_{h\in\mathcal{H}} \sum_{k\in\mathcal{K}} \sum_{r\in\Omega_{hk}} c_r z_r \tag{1}$$

$$\text{s.t.} \sum_{h\in\mathcal{H}} \sum_{k\in\mathcal{K}} \sum_{r\in\Omega_{hk}} a_{ir} z_r \geq 1 \qquad \forall i \in \mathcal{N} \tag{2}$$

$$\sum_{r\in\Omega_{hk}} z_r \leq u_{hk} \qquad \forall h \in \mathcal{H},\ k \in \mathcal{K} \tag{3}$$

$$z_r \in \{0,1\} \qquad \forall h \in \mathcal{H},\ k \in \mathcal{K},\ r \in \Omega_{hk} \tag{4}$$

Constraints (2) are standard set covering constraints, modeling condition (a), while (3) impose limits on the maximum number of available vehicles of each type at each depot, modeling condition (g). The objective is to minimize the overall cost of the selected routes. In the remainder we indicate this formulation as Master Problem (MP).

4 Branch-and-price

We solve the linear relaxation of the MP to obtain a lower bound which is used in a tree search algorithm. The number of variables is exponential in the cardinality of the customer set $\mathcal{N}$, thus we use a column generation approach.

Given a depot h and a vehicle type k, the problem of finding the most negative reduced cost column encoding a route for vehicle k using depot h turns out to be a *Resource Constrained Elementary Shortest Path Problem* (RCESPP). We solve it by using a dynamic programming algorithm whose structure is similar to the one proposed in [7], enriched with bidirectional label propagation and decremental state space relaxation techniques. Further, we need to deal with soft time windows. For this purpose we modify the algorithm proposed by Liberatore et al. [4] for the VRPSTW. The key idea is to store the cost of a path as a convex piecewise linear function of the time and to perform partial dominance on time intervals. In principle it would be necessary to solve, at each iteration, an instance of RCESPP for every combination of depot and

vehicle type, but in practice it is possible to avoid multiple executions. Two branching rules are used, one on the number of vehicles and the other one on the arcs used.

References

[1] J.-F. Cordeau. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, 54:573–586, 2006.

[2] J.-F. Cordeau, G. Laporte, and S. Ropke. Recent models and algorithms for one-to-one pickup and delivery problems. In B. L. Golden, S. Raghavan, and E.A. Wasil, editors, *Vehicle Routing: Latest Advances and Challenges*, pages 327–357. Springer, 2008.

[3] Y. Dumas, J. Desrosiers, and F. Soumis. The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54:7–22, 1991.

[4] F. Liberatore, M. Salani, and G. Righini. A pricing algorithm for the vehicle routing problem with soft time windows. In L. Bertazzi, M.G. Speranza, and J.A.E.E. van Nunen, editors, *Proceedings of the International Workshop on Distribution Logistics, Brescia, 2006*, volume 619 of *LNEMS*, pages 251–266, 2009.

[5] S. Parragh, K. Doerner, and R. Hartl. A survey on pickup and delivery problems: Part ii: Transportation between pickup and delivery locations. *Journal fr Betriebswirtschaft*, 58, 2008.

[6] A.G. Qureshi, E. Taniguchi, and T. Yamada. An exact solution approach for vehicle routing and scheduling problems with soft time windows. *Transportation Research Part E: Logistics and Transportation Review*, 45:960–977, 2009.

[7] S. Ropke and J.F. Cordeau. Branch and cut and price for the pickup and delivery problem with time windows. *TRANSPORTATION SCIENCE*, 43:267–286, 2009.

[8] S. Ropke, J.F. Cordeau, and G. Laporte. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks*, 49:258–272, 2007.

[9] M.W.P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29:17–29, 1995.

[10] M.W.P. Savelsbergh and M. Sol. Drive: Dynamic routing of independent vehicles. *Oper. Res.*, 29:474–490, 1998.

[11] T.R. Sexton and Y.M. Choi. Pickup and delivery of partial loads with soft time windows. *American Journal of Mathematical and Management Sciences*, 6:369–398, 1986.

Bisimplicial Edges in Bipartite Graphs ⋆

Matthijs Bomhoff * Bodo Manthey

Faculty of Electrical Engineering, Mathematics and Computer Science
University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract

Bisimplicial edges in bipartite graphs are closely related to pivots in Gaussian elimination that avoid turning zeroes into non-zeroes. We present a new deterministic algorithm to find such edges in bipartite graphs. The expected time complexity of our new algorithm is $O\left(n^2 \log n\right)$ on random bipartite graphs in which each edge is present with a fixed probability p, a polynomial improvement over the fastest algorithm found in the existing literature.

Key words: bipartite graphs, random graphs, algorithms, Gaussian elimination

1 Introduction

When applying Gaussian elimination to a square $n \times n$ matrix M containing some elements with value zero, the choice of pivots can often determine the amount of zeroes turned into non-zeroes during the process, the so called *fill-in*. Some matrices even allow Gaussian elimination without any fill-in. Avoiding fill-in has the nice property of bounding the required space for intermediate results of the Gaussian elimination to the space required for storing the input matrix M. This can be important for processing very large sparse matrices. Even when fill-in cannot be completely avoided, it may still be worthwhile to avoid it for several iterations, motivating the search for pivots that avoid fill-in.

⋆ This work was supported by the Dutch Innovation Oriented Research Program "Integral Product Creation and Realisation (IOP-IPCR)" of the Dutch Ministry of Economic Affairs.

* Corresponding Author
Email addresses: m.j.bomhoff@utwente.nl (Matthijs Bomhoff), b.manthey@utwente.nl (Bodo Manthey).

If we assume subtracting a multiple of one row of M from another turns at most one non-zero into a zero, we may restrict ourselves to considering only $\{0,1\}$ matrices. Given such a square matrix M, we can construct the bipartite graph $G[M]$ with vertices corresponding to the rows and columns in M, where vertices i and j are adjacent if and only if $M_{i,j}$ is nonzero. The $\{0,1\}$ matrices that allow Gaussian elimination without fill-in correspond to the class of *perfect elimination bipartite* graphs [1]. Central to the recognition of this class of graphs is the notion of a *bisimplicial* edge that corresponds to an element of M that can be used as a pivot without causing fill-in. The fastest algorithm for finding bisimplicial edges in the existing literature has a time complexity equal to that of matrix multiplication [2,3], i.e., $O\left(n^{2.376}\right)$ [4]. We present a new deterministic algorithm for finding a bisimplicial edge in a bipartite graph, if one exists, and show that its expected time complexity on random bipartite graphs where each edge is present with some fixed probability p is $O\left(n^2 \log n\right)$.

2 Bisimplicial Edges

We denote by $\Gamma(u)$ the neighbors of a vertex u.

Definition 1 *An edge uv of a bipartite graph $G = (U, V, E)$ is called* bisimplicial*, if the induced subgraph $G[\Gamma(u) \cup \Gamma(v)]$ is a complete bipartite graph.*

Clearly, for a given edge uv we can determine in $O(|E|)$ time if it is a bisimplicial edge by simply checking all edges adjacent to it. So a simple algorithm to find a bisimplicial edge in a bipartite graph G, if one exists, takes $O\left(|E|^2\right)$ time.

Goh and Rotem [2] present a faster algorithm based on the following: A row $M_{a,*}$ is said to *majorize* a row $M_{b,*}$ if for each $1 \leq j \leq n$ we have $M_{a,j} \geq M_{b,j}$. According to this definition, every row majorizes itself.

Theorem 2 (Goh and Rotem [2]) *Let M be an $n \times n$ $\{0,1\}$ matrix representing a bipartite graph $G = (U, V, E)$. Let ℓ_i be the number of rows in M that majorize row i and let s_j be the sum of the entries in column j of M. Then $M_{i,j} = 1$ and $\ell_i = s_j$ if and only if the edge $u_i v_j$ is a bisimplicial edge of G.*

Let $Q = MM^T$, this implies ℓ_i is equal to the number of elements in the row $Q_{i,*}$ that are equal to $Q_{i,i}$ (including $Q_{i,i}$ itself). Clearly, the time complexity of finding a bisimplicial edge in $G[M]$ is bounded by the time complexity of the matrix multiplication. The fastest currently known algorithm for this has a time complexity of $O\left(n^{2.376}\right)$ [4].

To improve on this, our new approach first selects a set of candidate edges. If a bisimplicial edge exists, then one of our candidates is bisimplicial. Thus we can restrict ourselves to checking the candidate edges for bisimpliciality. By bounding the number of candidates we achieve an improved expected time complexity. The following observation is the basis of our candidate selection procedure.

Lemma 3 *If an edge uv of a bipartite graph $G = (U, V, E)$ is bisimplicial, we must have $\delta(u) = \min_{u' \in \Gamma(v)} \delta(u')$ and $\delta(v) = \min_{v' \in \Gamma(u)} \delta(v')$.*

Translated to the matrix M, this means that if $M_{i,j} = 1$, it can only correspond to a bisimplicial edge if row i has a minimal number of ones over all the rows that have a 1 in column j and column j has a minimal number of ones over all the columns having a 1 in row i. In what follows, we will call the row (column) in M with the minimal number of ones over all the rows (columns) in M the *smallest* row (column). Using this observation, we construct an algorithm to pick candidate edges that may be bisimplicial:

Algorithm 1 *(1) Determine the row and column sums (a_i and b_j) for each row and column of M.*
(2) Determine for each row i the index c_i of the smallest column with $M_{i,c_i} = 1$ (breaking ties by favoring the lowest index); or $c_i = 0$ if row i has no one.
(3) Determine for each column j the index r_j of the smallest row with $M_{r_j,j} = 1$ (breaking ties by favoring the lowest index); or $r_j = 0$ if column j has no one.
(4) Mark $M_{i,j}$ as a candidate edge if $c_i = j$ and $r_j = i$.

Clearly, all steps in the algorithm can be performed in $O(n^2)$ time. Furthermore, the last step will mark at most n candidate edges (and at least 1).

Theorem 4 *If $G = (U, V, E)$ contains a bisimplicial edge, at least one of the candidates marked by the algorithm will be bisimplicial.*

As each of the candidates can subsequently be checked for bisimpliciality in $O(n^2)$, we obtain a $O(n^3)$ algorithm that finds a single bisimplicial edge in a bipartite graph G, if one exists, without using matrix multiplication. By itself, this is not really interesting, as the worst case time complexity is not an improvement over previously known algorithms. However, for random bipartite graphs, our new algorithm performs significantly better.

3 Asymptotic Expected Behavior on Random Graphs

For a fixed value of $p \in (0,1)$, we consider random bipartite graphs from the $G_{n,n,p}$ model: i.e., we have n vertices in each vertex class, and each edge is present with probability p. Such a random graph corresponds to a stochastic $n \times n$ $\{0,1\}$ matrix M with $\mathbb{P}[M_{i,j} = 1] = p$. We denote by random variable X_i the $\{0,1\}$ vector that forms row i of M and use $|X_i|$ to denote the sum of its elements. If we order the X_i vectors according to the number of ones they contain (breaking ties by favoring lower values of i), we denote by $X_{(1)}$ the row with the least number of ones, by $X_{(2)}$ the row with the second-to-least number etc.

Lemma 5 *For any $\varepsilon > 0$ and sufficiently large n, we have*

$$\mathbb{P}\left[|X_{(1)}| < (1-\varepsilon)pn\right] \leq \frac{1}{n}.$$

Lemma 6 *For any p', k with $0 < p' < p$ and $k \geq 1$ and sufficiently large n, we have*

$$\mathbb{P}\left[\textit{Algorithm 1 selects more than } k \textit{ candidates}\right] \leq n(1-p')^k + \frac{1}{n}.$$

From this, we immediately get a bound on the expected number of candidates that our algorithm selects.

Theorem 7 *For any p' with $0 < p' < p$ and sufficiently large n, we have*

$$\mathbb{E}\left[\textit{number of candidates selected by Algorithm 1}\right] \leq 2 + 2\log_{(1-p')}\frac{1}{n}.$$

Corollary 8 *For any fixed p, the expected time complexity of finding a bisimplicial edge or deciding there is none using Algorithm 1 is $O(n^2 \log n)$.*

References

[1] Martin Charles Golumbic and Clinton F. Goss. Perfect elimination and chordal bipartite graphs. *J. Graph Theory*, 2(2):155–163, 1978.

[2] L. Goh and D. Rotem. Recognition of perfect elimination bipartite graphs. *Inform. Process. Lett.*, 15(4):179–182, 1982.

[3] Jeremy P. Spinrad. Recognizing quasi-triangulated graphs. *Discrete Appl. Math.*, 138(1-2):203–213, 2004.

[4] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.*, 9(3):251–280, 1990.

A Heuristic Algorithm for the Train-Unit Assignment Problem

V. Cacchiani * A. Caprara P. Toth

DEIS, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

Key words: Train Unit Assignment, Lagrangian relaxation, Heuristic, Assignment Problem

1 The Train-Unit Assignment Problem

We study the Train-Unit Assignment Problem (TUAP) (see e.g. [2]), calling for an optimal assignment of train units (which are self-contained trains with an engine and passenger seats) to a given set of timetabled train trips. More precisely, each trip has a departure station, an arrival station, a departure time and an arrival time, and requires a number of passenger seats. Train units can be classified in different types: each train unit has a number of available seats and can be combined with other units in order to fulfill the seat requests. For each train trip a maximum number of train units that can be combined is given. In addition, sequencing constraints between trips must be satisfied: a pair of trips can be in a sequence for a train unit if the time elapsing between the arrival of the first one and the departure of the second one is large enough to allow the corresponding train unit to travel from the arrival station of the first one to the departure station of the second one. Finally, each train unit has to undergo a maintenance operation every fixed number of days, which requires a certain amount of time, as well as the transfer to and from the maintenance station. The goal is to minimize the number of train units globally used, while satisfying the seat requests, the sequencing constraints and the maintenance constraints.

* V. Cacchiani
Email addresses: valentina.cacchiani@unibo.it (V. Cacchiani), alberto.caprara@unibo.it (A. Caprara), paolo.toth@unibo.it (P. Toth).

2 Solution method

Let n be the number of trips and p the number of train unit types. Each trip $j \in \{1, \ldots, n\}$ is defined by a request r_j, given by the required number of passenger seats, a maximum number u_j of train units that can be assigned to the trip, and its timetable. Each train unit type $k \in \{1, \ldots, p\}$ is defined by a number d^k of available train units and an associated capacity s^k, given by the number of available seats. We consider the natural and well-known Integer Linear Programming (ILP) with arc variables, based on a canonical graph representation of the problem (see e.g. [1]). Let $G = (V, A)$ be a directed acyclic multigraph, in which nodes correspond to trips, and the arc set A is partitioned into p subsets $A^1, \ldots, A^p$, where A^k is associated with train units of type k. The sequencing constraints are implicitly represented by the graph. In particular, arc $(i, j)^k$ exists whenever the time between the arrival of trip i and the departure of trip j allows a train unit of type k to travel from the arrival station of trip i to the departure station of trip j. Let us introduce an integer variable x_{ij}^k ($k \in \{1, \ldots, p\}$, $i, j \in \{1, \ldots, n\}$), that indicates the number of times that arc $(i, j)^k$ is selected in the solution, i.e., the number of train units of type k that execute trip i before trip j in the associated sequence. Moreover, let c_{ij}^k denote the cost of arc $(i, j)^k$, corresponding to the time in minutes elapsing between the departure of the starting node and the departure of the ending node. The ILP reads as follows:

$$\min \sum_{k=1}^{p} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^k x_{ij}^k, \tag{1}$$

$$\sum_{i=1}^{n} x_{ij}^k = \sum_{i=1}^{n} x_{ji}^k, \quad k \in \{1, \ldots, p\},\ j \in \{1, \ldots, n\}, \tag{2}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^k x_{ij}^k \leq 1440\ d^k, \quad k \in \{1, \ldots, p\}, \tag{3}$$

$$\sum_{k=1}^{p} \sum_{i=1}^{n} s^k x_{ij}^k \geq r_j, \quad j \in \{1, \ldots, n\}, \tag{4}$$

$$\sum_{k=1}^{p} \sum_{i=1}^{n} x_{ij}^k \leq u_j, \quad j \in \{1, \ldots, n\}, \tag{5}$$

$$x_{ij}^k \geq 0,\ \text{integer}\ , \quad k \in \{1, \ldots, p\}, i, j \in \{1, \ldots, n\}. \tag{6}$$

The objective is to minimize the overall cost of the selected arcs, equal to 1440 (the number of minutes in a day) times the number of the train units globally used. Constraints (2) express the flow conservation. Constraints (3) forbid to use more than the available train units for each type. Constraints (4) impose to cover each trip with the corresponding seat request. Constraints (5) impose a bound on the number of train units that can be used to cover each trip.

Note that the maintenance constraints are not introduced in the formulation, since they would be complex to model; we take them into account directly in the heuristic algorithm.

The approach in [1], based on solving LP relaxations by general-purpose LP software, considers an alternative equivalent ILP formulation with path variables, whose LP relaxation is much faster to solve (modulo using column generation). In this work, we consider a Lagrangian-relaxation based approach, which nicely combines with the ILP above. Specifically, we first replace constraints (5) by the weaker

$$\sum_{i=1}^{n} x_{ij}^k \leq u_j, \quad k \in \{1,\ldots,p\}, j \in \{1,\ldots,n\}, \tag{7}$$

which impose that each train unit of type k can cover each trip j at most u_j times. Then, we relax in a Lagrangian way constraints (3) and (4), by using nonnegative Lagrangian multipliers $\lambda_j, (j \in \{1,\ldots,n\})$ and $\sigma_k, (k \in \{1,\ldots,p\})$, respectively.

The resulting Lagrangian relaxed problem decomposes onto p independent subproblems, one for each train unit type. Let $\tilde{c}_{ij}^k := (c_{ij}^k - \lambda_j s^k + \sigma_k c_{ij}^k)$ be the Lagrangian cost for each arc $(i,j)^k \in A$. We impose a null Lagrangian cost ($\tilde{c}_{ii}^k := 0$) for loops $(i,i)^k$ for each vertex $i \in V$ (that had originally infinity cost). Thus, the subproblem associated with a train unit type k calls for the minimization of $\sum_{i=1}^n \sum_{j=1}^n \tilde{c}_{ij}^k x_{ij}^k$ subject to (2) and (7). Moreover, the presence of zero-cost loops allows us to replace inequality by equality in (7). It is well known that all vertices of the feasible region of this subproblem are integer for u_j integer and the associated constraint matrix is totally unimodular. Moreover, in the particular case, arising in our case study, in which u_j does not depend on j, we can replace u_j with u for $k \in \{1,\ldots,p\}$ and $j \in \{1,\ldots,n\}$. This makes the subproblem equivalent to an Assignment Problem (AP), obtained by replacing u by 1 in (7), the correspondence between solutions $\bar{x}$ of the subproblem and $\bar{y}$ of AP being given by $\bar{x} = u\bar{y}$. As is well known, the assignment problem can be solved in $O(n^3)$ time. In order to find good Lagrangian multipliers we apply a standard iterative subgradient procedure.

Besides yielding a valid lower bound on the optimal solution value, the best Lagrangian multipliers λ^*, σ^* found throughout the iterations and the corresponding reduced costs $\bar{c}_{ij}^k := c_{ij}^k - \lambda_j^* s^k + \sigma_k^* c_{ij}^k - \bar{w}_i^k - \bar{v}_j^k$, where $\bar{w}^k$ and $\bar{v}^k$ are the optimal AP dual variables associated with the assignment constraints, are used to drive the following constructive Lagrangian heuristic algorithm. We order the train unit types for decreasing capacity values, and construct the workload for each train unit, until either all the trips have been covered or all the train units have been used (in this case, we apply a local search procedure to obtain a feasible solution). The construction starts with the selection of

a trip whose departure is in the beginning of the day. Then, we choose the following trips by assigning to each one a score that takes into account the reduced costs, the original costs and how "well" the current train unit can cover the trip (taking into account which other train units are still available). In addition, we give a prize to the arcs that allow to perform maintenance, until we reach the number of necessary maintenance operations. Every time a trip is selected in the solution, we update its number of seats and the reduced cost of each arc entering the corresponding node. In order to decide how to end the workload of the current train unit type, for each selected trip in the workload we solve the relaxed problem on the residual train units and trips. The solution value of the reduced relaxed problem gives a lower bound on the global number of train units that are needed to cover all the residual trips. When no more train units are available of the current type, we end the workload with the trip giving the smallest solution value.

3 Computational Experiments

We present some preliminary computational experiments on a set of real-world instances for an operator running trains in a regional area, and compare the results with the approach presented in [1], with and without imposing the maintenance constraints. The tests were performed on a PC Pentium 4, 3.2 GHz, 2 GB RAM, and using Cplex 9.0 as an LP-solver in [1]. The results are presented in Table 1, showing that the proposed approach produces comparable results within much shorter computing time (expressed in seconds).

			Lagr. heur.		[1] heur.		Lagr. heur. maint.		[1] heur. maint.	
inst.	n	p	value	time	value	time	value	time	value	time
1	85	1	2	0	2	0	2	0	2	0
2	120	1	4	0	4	0	4	1	4	5
3	302	1	17	4	17	288	17	5	17	544
4	208	2	26	4	25	17	27	3	25	19
5	364	2	20	18	20	1912	21	20	20	3899

Table 1
Comparison on a set of real-world instances in the case without or with the maintenance constraints.

References

[1] Cacchiani, V., Caprara, A., and Toth, P., "Solving a Real-World Train Unit Assignment Problem", *Mathematical Programming Series B*, to appear (2010).

[2] Caprara, A., Kroon, L., Monaci, M., Peeters, M., and Toth, P., "Passenger Railway Optimization", in C. Barnhart and G. Laporte (eds.), *Handbooks in OR & MS*, Vol. 12, Elsevier Science (2006).

Optimization algorithms for the Max Edge Weighted Clique problem with Multiple Choice constraints

Alberto Ceselli [a,*] Roberto Cordone [a] Yari Melzani [a]
Giovanni Righini [a]

[a] *Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano*
{alberto.ceselli,roberto.cordone,giovanni.righini}@unimi.it,
yari.melzani@gmail.com

Key words: Max Clique, semidefinite programming, tabu search, branch-and-bound

1 Introduction

Many relevant problems can be described and successfully solved using graph-based models. For instance, the side-chain placement problem in biology, the choice of efficient communication protocols in management science, backbone design in telecommunication networks and data mining applications share a common structure: they all require to find a maximum weighted clique in a suitable graph [1].
We tackle the *Max Edge Weighted Clique problem with Multiple choice Constraints* (MEWCMC). Given a graph with weights on both vertices and edges, and a partition of the vertex set, the MEWCMC asks to find a maximum weight clique, choosing one vertex for each class of the partition.
The classical Max Weighted Clique problem has been studied from many aspects. State of the art algorithms include [3], [4] and [5]. The version of the problem involving multiple choice constraints, instead, has been tackled only recently [2].
In this paper we first introduce models for the MEWCMC based on Binary Quadratic Programming (BQP) and Integer Linear Programming (ILP); we also introduce a model suitable to obtain a semidefinite relaxation of the MEWCMC. Since using commercial solvers on these models allows to solve only small size instances, we present an exact algorithm exploiting a semidefinite relaxation of the MEWCMC, combinatorial bounding, rounding and problem reduction procedures in a branch-and-bound framework. We also describe a Tabu Search heuristic, able to quickly find good quality solutions.

* Corresponding author

2 Models

It is given a graph $G(V, E)$, where V is a set of n vertices and $E \subseteq V \times V$ is a set of ℓ edges. It is also given a partition $K = \{V_1, V_2, .., V_m\}$ of V. Let $w_E : E \to \mathbb{R}$ and $w_V : V \to \mathbb{R}$ be two functions mapping respectively every edge and every vertex to a weight value. The MEWCMC consists in finding a clique in G, that is finding a set of vertices $M \subseteq V$, such that $(i, j) \in E \ \forall i, j \in M$, having maximum weight $z(M) = \frac{1}{2} \sum_{v_i \in M} \sum_{v_j \in M} w_{ij}$. Furthermore, in order to fulfill multiple choice constraints, M has to include exactly 1 vertex for each class, that is $|M \cap V_k| = 1 \, \forall V_k \in K$.
By introducing a binary variable x_i for each vertex $i \in V$, such that x_i takes value 1 if vertex $i \in M$, 0 otherwise, we can formulate the MEWCMC as the following BQP problem:

$$\text{maximize} \sum_{i \in V} \sum_{j \in V} w_{ij} x_i x_j \tag{1}$$

$$\text{s.t.} \sum_{i \in V_k} x_i = 1 \qquad \forall k \in K \tag{2}$$

$$x_i \in \{0, 1\} \qquad \forall i \in V. \tag{3}$$

Multiple choice constraints (2) impose that exactly one vertex is selected for each class $V_k \in K$. This model can be either directly used by general purpose BQP solvers, or linearized using standard techniques, obtaining an ILP problem, suitable to be optimized by standard solvers. Both approaches, however, show to fail as the size of instances increases, mainly due to the poor quality of the continuous relaxation of both models.
Hence, we derive a matrix-based formulation of the problem as follows. First we define a matrix W, where each element w_{ij} with $i \neq j$ is the weight associated to edge $(i, j) \in E$ and each element w_{ii} is the weight associated to vertex $i \in V$, and we introduce a square matrix Y of variables y_{ij} of dimension n, where $y_{ij} = x_i x_j$ for each $i, j \in V$. The MEWCMC can then be stated as follows:

$$\text{maximize } W \bullet Y \tag{4}$$

$$\text{s.t. } \text{rank}(Y) = 1 \tag{5}$$

$$\sum_k S_k \bullet Y = 1 \quad k = 1, ..., m \tag{6}$$

$$Y \succeq 0 \tag{7}$$

$$y_{ij} \in \{0, 1\} \tag{8}$$

where $\bullet$ is the *Frobenius* matrix product, $rank(Y)$ is the rank of matrix Y, $Y \succeq 0$ imposes Y to be positive semidefinite, I is the identity matrix of dimension n. Constraints (5) and (7) guarantee that matrix Y can be represented as the product xx^T, and constraints (8) guarantee that each value in the matrix is binary. Inequalities (6) represent multiple choice constraints: matrices S_k are

built in such a way that

$$\sum_{i \in V_k} Y_{ii} = \sum_{i \in V} \sum_{j \in V} (S_k)_{ij} Y_{ij} = S_k \bullet Y = 1 \quad \forall V_k \in K,$$

that is, having value 1 only on some diagonal elements, and value 0 elsewhere.

3 Algorithms

We devised an exact branch-and-bound algorithm exploiting formulation (4)–(8) and combinatorial arguments. In the remainder we sketch its main components.

Dual bounds. By considering model (4)–(8) and relaxing constraints (5) and (8) we obtain a semidefinite programming problem; this is a special case of convex optimization problem, that can be solved very efficiently by special purpose algorithms [6]. The relaxed model obtained in this way is strengthened in two ways. First, since one vertex has to be selected from each class of the partition, m vertices have to be selected overall; therefore, the constraint $I \bullet Y = m$ can be added to the formulation. Second, we state constraints in stronger forms, by disaggregating and exploiting some properties of BQP problems. At the same time, we consider the following value

$$DB_c = \frac{1}{2} \sum_{V_s \in K} \max_{i \in V_s} \{ w_{ii} + \sum_{V_t \in K, t \neq s} \max_{j \in V_t} \{ \frac{w_{jj}}{m-1} + w_{ij} \} \}; \tag{9}$$

and we prove by combinatorial arguments that expression (9) gives a valid dual bound to the MEWCMC. The best of the two bounds is kept as the final dual bound.

Primal bounds. At each node of the branch-and-bound tree we try to find good feasible solutions by (a) rounding the optimal solution of the semidefinite relaxation (b) correcting the solutions given by the combinatorial bounding procedure (9). Both methods are able to find good solutions in the early nodes of the branch-and-bound tree. We also devised a Tabu Search heuristic; it works as follows. We start from a clique M either corresponding to a known feasible solution or obtained by selecting a random vertex for each class of the partition K; at each step we explore the neighborhood given by all solutions obtained by replacing one the vertices in M by a different vertex of the same class, moving to the best solution in the neighborhood. We keep two tabu lists: the first keeps track of vertices recently removed from M, that cannot be selected again; the second keeps track of vertices recently introduced in M, that cannot be removed. We stop after a fixed number of steps. Preliminary experiments shows that by setting the length of the first tabu list to 8, that of the second tabu list to 1 and the maximum number of steps to 1000, this heuristic is able to produce optimal solutions on a large set of instances; besides being useful on its own, this heuristic is used to obtain tight primal bounds at the root node of the branching tree.

Branching. The diagonal elements of Y basically represent how much any vertex is (fractionally) selected in the semidefinite relaxation solution. Therefore, when primal and dual bounds do not match, we perform binary branching by selecting the class V_k having the highest number of these fractional entries; then we try to partition the vertices of class V_k in two subsets V_k^l and V_k^r, having the most balanced sum of fractional entries. In the left branch and right branches we respectively forbid to choose in the clique any vertex of the set V_k^l, and any vertex of the set V_k^r. We visit the branching tree in a best bound order.

Performance improving techniques. We improved the performances of our algorithm by (a) switching to complete enumeration when, due to the remotion of forbidden vertices, the size of the subproblem gets small enough (b) performing problem reduction tests, that is trying to fix each vertex to be part of the solution, computing the combinatorial dual bound on the remaining subproblem and removing from the problem such a vertex if the dual bound computed in this way is worse than the best known primal bound.

4 Experimental results

We implemented our algorithms in C, we used ILOG CPLEX 11.2 as both ILP and BQP solver, and DSDP5 as semidefinite programming solver. We performed experiments on two datasets. Dataset S1 is drawn from [2]; it is composed by 168 instances of four types, having weights on vertices and edges randomly drawn in different ranges. These instances require the optimization on graphs with up to 65 vertices. Dataset S2, instead, consists of 165 new instances of three types.The number of vertices of the graphs in these instances range from 30 to 300. Our experiments ran on a Centrino Core 2 3GHz PC, equipped with 2GB of RAM, in Linux 32 bit environment. A full description of Datasets and computational experiments is available online [1].

Our experimental campaign allowed us to find that (a) the tabu search algorithm is able to find the optimal solution on 79.76% of the instances in Dataset S1, and on 88.48% of the instances in Dataset S2; (b) the BQP solver of CPLEX using formulation (1)–(3) is not competitive with other methods (c) the exact algorithm proposed in [2] is outperformed by both CPLEX and our algorithm on both datasets (d) using the best of our ILP formulations, CPLEX could solve only 44% of the instances in Dataset S2 within a time limit of one hour, while our exact algorithm solved 77% of them; when both CPLEX and our algorithm terminate within the time limit, our algorithm is up to three orders of magnitude faster, and when none of them terminates, the remaining gap between primal and dual bounds for our algorithm is a fraction of that of CPLEX.

[1] http://www.dti.unimi.it/∼ceselli/EWCMC.html

References

[1] Y. Melzani (2009) "Mathematical programming algorithms for the Max edge weighted clique problem with multiple choice constraints", Master thesis, Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano.

[2] A. Bosio (2005) "A mathematical programming algorithm for the Max edge weighted clique problem with multiple choice constraints", Degree thesis, Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano.

[3] B. Alidaee, F. Glover, G. Kochenberger, H. Wang (2007) "Solving the maximum edge weight clique problem via unconstrained quadratic programming", European Journal of Operational Research, 181(1): 592–597.

[4] M. M. Sørensen (2004) "New facets and a branch-and-cut algorithm for the weighted clique problem", European Journal of Operational Research, 154(1): 57–70.

[5] M. Hunting, U. Faigle, W. Kern (2001) "A Lagrangian Relaxation Approach to the Edge-Weighted Clique Problem", European Journal of Operational Research, 131

[6] S. J. Benson, Y. Ye (2005) "DSDP5: Software for Semidefinite Programming", Technical report, Mathematics and Computer Science Division, Argonne National Laboratory.

A branch and bound method for a clique partitioning problem

Irène Charon, [a] Olivier Hudry, [a]

[a] *Télécom ParisTech & CNRS - LTCI UMR5141, 46, rue Barrault, 75634 Paris Cedex 13, France*

Key words: Branch and bound, classification, clique partitioning of a graph, clustering, combinatorial optimization, graphs, Lagrangean relaxation, median equivalence relation, Régnier's problem, Zahn's problem

1 Introduction

We consider here the problem of the approximation of m symmetric relations defined on a same finite set X into a so-called *median equivalence relation* (see below and [1]), with in particular two special cases: the one for which the m symmetric relations are equivalence relations (Régnier's problem [4]), and the one of the approximation of only one symmetric relation ($m = 1$) by an equivalence relation (Zahn's problem [6]). These problems arise for instance from the field of classification or clustering: in this case, X is a set of entities (which can be objects, people, projects, propositions, alternatives, and so on) that we want to gather in subsets of X in such a way that the elements of any such subset can be considered as similar while the objects of different subsets can be considered as dissimilar. Each symmetric relation is associated with a criterion specifying, for any pair $\{x, y\}$ of entities, whether x and y are similar or not. Then we try to find the best compromise between all these criteria. This leads us, in Section 2, to state this problem as a graph theoretical problem, that we call CPP for *clique partitioning problem*. As this problem is NP-hard, we design in Section 3 a branch and bound algorithm to solve this problem, based on a Lagrangean relaxation method for the evaluation function.

2 The clique partitioning problem

The problem that we consider here can be mathematically described as follows. We are given a collection $\Pi = (S_1, S_2, ..., S_m)$ of m symmetric binary relations

S_k, $1 \leq k \leq m$, all defined on a same finite set X of n elements (Régnier's problem [4] corresponds to the case for which all the relations S_k are equivalence relations; Zahn's problem [6] corresponds to the case for which m is equal to 1). We consider the number $\delta(R, S)$ of disagreements between two binary relations R and S:

$$\delta(R, S) = |\{(i, j) \in X^2 \text{ with } [iRj \text{ and not } iSj] \text{ or } [iSj \text{ and not } iRj]\}|.$$

Then, for any equivalence relation E, we consider the *remoteness* $\Delta(\Pi, E) = \sum_{k=1}^{m} \delta(S_k, E)$, measuring the total number of disagreements between Π and E. Our problem thus consists in computing an equivalence relation E^*, called a *median equivalence relation* of Π, which minimizes Δ over the set $\mathcal{E}$ of all the equivalence relations defined on X:

$$\Delta(\Pi, E^*) = \min_{E \in \mathcal{E}} \Delta(\Pi, E).$$

The computation of E^* is NP-hard [5], and remains so even for Régnier's problem or for Zahn's problem.

To state this problem as a 0-1 linear programming problem, let $s^k = (s^k_{ij})_{(i,j) \in X^2}$ $(1 \leq k \leq m)$ be the binary vector defined by: $s^k_{ij} = 1$ if iS_kj (i.e. if i and j are put together by S_k), and $s^k_{ij} = 0$ otherwise. Similarly, let $(x_{ij})_{(i,j) \in X^2}$ denote the vector associated with E: $x_{ij} = 1$ if iEj, $x_{ij} = 0$ otherwise. It is easy to obtain the following:

$$\delta(S_k, E) = \sum_{(i,j) \in X^2} |s^k_{ij} - x_{ij}| = \sum_{(i,j) \in X^2} (s^k_{ij} - x_{ij})^2 = \sum_{(i,j) \in X^2} (s^k_{ij} + (1 - 2s^k_{ij})x_{ij})$$

because of the binary property of the quantities s^k_{ij} and x_{ij}. Then we obtain, for the remoteness:

$$\Delta(\Pi, E) = \sum_{k=1}^{m} \sum_{(i,j) \in X^2} s^k_{ij} + \sum_{k=1}^{m} \sum_{(i,j) \in X^2} (1 - 2s^k_{ij})x_{ij} = C + \sum_{(i,j) \in X^2} w_{ij}x_{ij}$$

where $C = \sum_{k=1}^{m} \sum_{(i,j) \in X^2} s^k_{ij}$ is a constant and with, for $(i, j) \in X^2$:

$$w_{ij} = \sum_{k=1}^{m} (1 - 2s^k_{ij}) = m - 2|\{k \text{ with } 1 \leq k \leq m \text{ and } iS_kj\}|.$$

So, minimizing $\Delta(\Pi, E)$ is the same as minimizing $\sum_{(i,j) \in X^2} w_{ij}x_{ij}$. Moreover, the constraints to state that E must belong to $\mathcal{E}$ are the following:

- symmetry: $\forall (i, j) \in X^2, x_{ij} = x_{ji}$;
- transitivity: $\forall (i, j, h) \in X^3$ with $i \neq j \neq h \neq i, x_{ij} + x_{jh} - x_{ih} \leq 1$.

If we add the binary constraints: $\forall (i, j) \in X^2, x_{ij} \in \{0, 1\}$, we obtain our 0-1 linear programming problem.

We now may state this problem as a graph theoretic one. For this, we associate the complete graph K_n to Π, and we weight every edge $\{i, j\}$ of K_n by w_{ij}. Then the variables x_{ij} equal to 1 define cliques (i.e. complete subgraphs) of K_n, and the value of $\Delta(\Pi, E)$ is equal to the sum of the weights of the edges with both extremities inside a same clique. Hence our clique partitioning problem CPP. Note that the weights of the edges can be non-positive or non-negative integers. Moreover, the number of cliques into which we want to partition K_n is not given. Finally, CPP can be stated as follows: given a complete graph $K_n = (X, A)$ whose edges $\{i, j\}$ are weighted by non-positive or non-negative integers w_{ij}, partition X into p subsets X_1, X_2, ..., X_p, where p is not given, so that $\sum_{h=1}^{p} \sum_{(i,j)\in(X_h)^2} w_{ij}$ (i.e. the sum of the weights of the edges inside the cliques) is minimum.

3 The branch and bound method

To solve CPP, we design a branch and bound method BB. We briefly depict the main ingredients of BB.

The initial bound is provided by a metaheuristic, namely the *noising methods* [2], [3]. The noising methods usually compute very good solutions, quite often optimal, though we cannot know whether these solutions are indeed optimal.

The BB-tree is built as follows. The vertices v_i of K_n are integers belonging to $\{1, 2, ..., n\}$. A partition with p subsets X_1, X_2, ..., X_p is represented as:

$$\underbrace{v_1, v_2, ..., v_{q_1}}_{X_1} \mid \underbrace{v_{q_1+1}, v_{q_1+2}, ..., v_{q_2}}_{X_2} \mid ... \mid \underbrace{v_{q_{p-1}+1}, v_{q_{p-1}+2}, ..., v_{q_p}}_{X_p}$$

With such an encoding, a partition admits several representations. To avoid this, we suppose that the vertices are ordered by increasing value within a subset and subsets are ordered according to their smallest vertices; with the above notation, it means that we have: $1 = v_1 < v_2 < ... < v_{q_1}$, $v_{q_1+1} < v_{q_1+2} < ... < v_{q_2}$, ..., $v_{q_{p-1}+1} < v_{q_{p-1}+2} < ... < v_{q_p}$, and $v_1 < v_{q_1+1} < ... < v_{q_{p-1}+1}$.

The subsets are progressively constructed. A node N of the BB-tree corresponds to the beginning of a partition encoding, something like:

$$\underbrace{v_1, v_2, ..., v_{q_1}}_{X_1} \mid \underbrace{v_{q_1+1}, v_{q_1+2}, ..., v_{q_2}}_{X_2} \mid ... \mid \underbrace{v_{q_{h-1}+1}, v_{q_{h-1}+2}, ..., v_{q_{h-1}+t}}_{X_h}$$

We extend N by at most $n - q_{h-1} - t + 1$ new branches. The first branch is obtained by closing the current subset X_h and by creating a new subset X_{h+1} which will contain at least $v_{q_{h-1}+t+1}$. The other branches correspond with the possibilities to expand the current class X_h by adding an extra vertex (greater

than $v_{q_{h-1}+t}$) to it: $v_{q_{h-1}+t+1}$, or $v_{q_{h-1}+t+2}$ but not $v_{q_{h-1}+t+1}$, or $v_{q_{h-1}+t+3}$ but neither $v_{q_{h-1}+t+1}$ nor $v_{q_{h-1}+t+2}$, and so on...

Three evaluation functions F_1, F_2, F_3 are designed to evaluate the quality of every node N of the BB-tree. They can be split into two parts. The first part is the same for the three functions: it takes into account the contribution of the vertices already dispatched inside the subsets of the partition under construction associated with N; for this, we only sum the weights of the edges with both extremities in a same subset. The second part depends on the function. For F_1, we add all the negative weights of the edges with at least one extremity greater than $v_{q_{h-1}+t}$. In F_2, we sharpen the design of F_1 by considering some triples of vertices (triangles) $\{a, b, c\}$ and by noting that if the weights of the edges between a, b and c have not the same sign, then the contribution of $\{a, b, c\}$ cannot be the sum of the negative edges, as in F_1; we design a greedy algorithm to choose these triangles in order to improve F_1 as much as possible. The last function, F_3, is the most sophisticated. It is based on the Lagrangean relaxation of the transitivity constraints (see above).

Other ingredients, not described here, allow us also to cut branches of the BB-tree. During the talk, we will discuss the efficiency of the evaluation functions and of the other ingredients, based on experiments dealing with different kinds of graphs: instances of Régnier's problem or of Zahn's problem, instances coming from the literature, random instances, or instances with special combinatorial or algorithmic properties.

References

[1] J.-P. Barthélemy, B. Monjardet: The median procedure in cluster analysis and social choice theory, Mathematical Social Sciences 1, 1981, 235-267.

[2] I. Charon, O. Hudry: Noising methods for a clique partitioning problem, Discrete Applied Mathematics 154 (5), 2006, 754-769.

[3] I. Charon, O. Hudry: Self-tuning of the noising methods, Optimization 58 (7), 2009, 1-21.

[4] S. Régnier: Sur quelques aspects mathématiques des problèmes de classification automatique, I.C.C. Bulletin 4, Rome, 1965.

[5] Y. Wakabayashi: The Complexity of Computing Medians of Relations, Resenhas, 3 (3), 1998, 323-349.

[6] C.T. Zahn: Approximating symmetric relations by equivalence relations, SIAM Journal on Applied Mathematics, 12, 1964, 840-847.

Static symmetry breaking in circle packing

Alberto Costa, Pierre Hansen *, Leo Liberti

LIX (UMR CNRS 7161), École Polytechnique, 91128 Palaiseau, France.

Key words: symmetry breaking constraints, packing of equal circles, reformulation, narrowing, nonconvex NLP.

1 Introduction

We present new Static Symmetry-Breaking Inequalities (SSBI) [11,6] for the problem of packing equal circles in a square [9]. The new SSBIs provide a marked computational improvement with respect to past work [1], though not yet at the level where a purely Mathematical Programming (MP) based spatial Branch-and-Bound (sBB) can be competitive with a Branch-and-Bound (BB) "boosted" by combinatorial and geometrical devices such as [9]. We consider the following formulation of Circle Packing in a Square (CPS) problem: given $N \in \mathbb{N}$ and $S \in \mathbb{Q}_+$, can N non-overlapping circles of unit radius be arranged in a square of side $2S$? This is equivalent to the more usual formulation where one maximizes the number of non-overlapping circles of unit radius in a square of side $2S$ with $S \in \mathbb{Q}_+$: it suffices to consider the usual correspondence (via bisection) of optimization and decision problems.

Let $\mathcal{N} = \{1, \ldots, N\}$ and $\mathcal{N}' = \{1, \ldots, N-1\}$. The CPS is formulated as the following MP:

$$\max\{\alpha \mid \forall i < j \in \mathcal{N}\ (x_i - x_j)^2 + (y_i - y_j)^2 \geq 4\alpha \wedge x, y \in [1-S, S-1]^N\} \quad (1)$$

where $(x_i, y_i) \in \mathbb{R}^2$ are the coordinates of the center of the i-th circle, for all $i \in \mathcal{N}$. For any given $N, L > 1$, if a global optimum (x^*, y^*, α^*) of (1) has $\alpha^* \geq 1$ then the CPS instance is a YES one. The CPS formulation (1) can be solved with standard off-the-shelf Mixed-Integer Nonlinear Programming (MINLP) sBB solvers such as Couenne [2]. As the instance size increases,

⋆ Financial support by grants: ANR 07-JCJC-0151 "ARS", Digiteo 2009-14D "RM-NCCO", Digiteo 2009-55D "ARM" is gratefully acknowledged.

* Also at GERAD and HEC Montreal, Canada.

Email addresses: costa@lix.polytechnique.fr (Alberto Costa), pierre.hansen@gerad.ca (Pierre Hansen), liberti@lix.polytechnique.fr (Leo Liberti).

these solvers yield search trees of disproportionate sizes. This is mostly due to the symmetries of the problem.

The concepts of *solution symmetries* and *formulation symmetries* were introduced in Constraint Programming [3] and brought to MP in the early 2000's [10,11]. If z is a solution of a problem P and πz is also a solution (where π permutes the components of z), π is a solution symmetry. A solution symmetry is a formulation symmetry if π also fixes the MP formulation of P. Most symmetry breaking techniques (including SSBIs) are based on formulation symmetries, because these are easier to detect. The formulation group of MINLPs (including nonconvex NLPs such as (1)) can be detected automatically using the method described in [6]. This method was shown in [7] to yield an interesting reformulation for another sphere packing problem, namely the Kissing Number Problem (KNP) [4]. Adjoining SSBIs to a formulation results in a reformulation of the *narrowing* type [5,8]: if Q is a narrowing of P then there is a mapping from the global optima $\mathcal{G}(Q)$ to the global optima $\mathcal{G}(P)$ — thus, if one is able to solve the simpler reformulation Q, then one can find a global optimum of P through the given mapping.

The automatic symmetry detection method of [6] was deployed in [1] on increasingly larger CPS instances to formulate the conjecture, and then prove, that the formulation group of the CPS is $C_2 \times S_N$, where C_2 (the cyclic group of order 2) refers to swapping x and y axes and S_N (the symmetric group of order N) refers to reindexing the circles in an arbitrary way. The constraints $\forall i \in \mathcal{N}'\ (x_i \leq x_{i+1})$ were shown in [1] to provide a narrowing of the CPS when adjoined to (1). In the rest of this paper we present a different narrowing of the CPS and discuss its impact on Couenne's performance.

2 New SSBI-based CPS narrowing

Let $L = \lfloor S \rfloor$, $\mathcal{N}'' = \{1, L+1, 2L+1, \ldots, (\lceil N/L \rceil - 2)L + 1\}$, and define the following constraint sets: $\mathscr{S} = \{x_i \leq x_{i+1} \mid i \in \mathcal{N}'\}$, $\mathscr{A}_i = \{x_h \leq x_{h+1} \mid h \in \mathcal{N}' \smallsetminus \{i+L-1\}\}$ and $\mathscr{C}_i = \{y_i \leq y_{i+L}\}$ for all $i \in \mathcal{N}''$. Notice that these sets contain strings belonging to the formal MP language [1]: thus, when writing $\{y_i \leq y_{i+L}\}$, for example, we do not refer to the set of all points y satisfying $y_i \leq y_{i+L}$ but rather to the singleton set containing the string "$y_i \leq y_{i+L}$" as its element. Accordingly, we consider the following MP formulations: $\text{CPS}' \equiv \text{CPS} \cup \mathscr{S}$, $\text{CPS}_i \equiv \text{CPS} \cup \mathscr{A}_i \cup \mathscr{C}_i$ for all $i \in \mathcal{N}''$ and $\text{CPS}'' \equiv \text{CPS} \cup \bigcup_{i \in \mathcal{N}''} (\mathscr{A}_i \cup \mathscr{C}_i)$, where $P \cup \mathscr{D}$ denotes the MP formulation derived by adjoining constraints in $\mathscr{D}$ to P. The formulation CPS$'$ was shown in [1] to be a narrowing of CPS.

Proposition 1 *For all $i \in \mathcal{N}''$, CPS$_i$ is a narrowing of CPS.*

Proof. Let $i \in \mathcal{N}''$ and $(\bar{x}, \bar{y}, \bar{\alpha}) \in \mathcal{G}(\text{CPS})$. For a permutation $\pi \in S_N$ we assume $\pi(\bar{x}, \bar{y}, \bar{\alpha}) = (\pi\bar{x}, \pi\bar{y}, \bar{\alpha})$ where π acts on a vector in $\mathbb{R}^N$ by permuting the indices of its components; notice that since π is simply a reindexing of the circles, $\pi(\bar{x}, \bar{y}, \bar{\alpha}) \in$

$\mathcal{G}$(CPS). Furthermore, since CPS$'$ is known to be a narrowing of CPS, we can assume WLOG that $(\bar{x}, \bar{y}, \bar{\alpha})$ satisfies $\mathscr{S}$. If $\bar{y}_i \leq \bar{y}_{i+L}$ the result holds, otherwise assume $\bar{y}_i > \bar{y}_{i+L}$. Consider the permutation $\sigma_i = \prod_{\ell=0}^{L-1}(i+\ell, i+L+\ell)$ in S_N; $\sigma_i(\bar{x}, \bar{y}, \bar{\alpha})$ has the following properties: (a) by the action of the 2-cycle $(i, i+L)$ (appearing in σ_i when $\ell = 0$) we have $\bar{y}_i < \bar{y}_{i+L}$; (b) $\forall \ell \in \{0, \ldots, L-2\}$ we have $\sigma_i \bar{x}_{i+\ell} = \bar{x}_{i+L+\ell} \leq \bar{x}_{i+L+\ell+1} = \sigma_i \bar{x}_{i+\ell+1}$ and $\sigma_i \bar{x}_{i+L+\ell} = \bar{x}_{i+\ell} \leq \bar{x}_{i+\ell+1} = \sigma_i \bar{x}_{i+L+\ell+1}$; (c) $\forall h \in \mathcal{N}'$ such that $h \notin H_i = \{i, \ldots, i+2L-1\}$ we have $\sigma_i \bar{x}_h = \bar{x}_h \leq \bar{x}_{h+1} = \sigma_i \bar{x}_{h+1}$ because σ_i fixes all $h \notin H_i$. Thus $\sigma_i(\bar{x}, \bar{y}, \bar{\alpha}) \in \mathcal{G}(CPS)$ and satisfies the constraints of CPS$_i$. □

Lemma 2 *Let $n = \lceil N/L \rceil - 1$ and $\Sigma = \{\sigma_i \mid i \in \mathcal{N}''\}$. Then $\langle \Sigma \rangle \cong S_n$.*

Proof. Notice $\mathcal{N}'' = \{(j-1)L+1 \mid 1 \leq j \leq n\}$, and define a map $\varphi((j-1)/L+1) = j$, under which $\varphi(\Sigma) = \{(1,2), (2,3), \ldots, (n-1, n)\}$. This map induces a group homomorphism $\bar{\varphi} : \langle \Sigma \rangle \to S_n$ given by $\bar{\varphi}(\sigma_i) = (\varphi(i), \varphi(i)+1)$, which can be verified to be injective and surjective. □

Similarly, for all $h < k \in \mathcal{N}''$ we have $\langle \Sigma^{hk} \rangle = \langle \{\sigma_i \mid h \leq i < k\} \rangle \cong \mathrm{Sym}(I^{hk})$, the symmetric group on the set $I^{hk} = \{\varphi(h), \ldots, \varphi(k)\}$. Thus, for all $h, k \in \mathcal{N}''$, the permutation $\tau_{hk} = \prod_{\ell=0}^{L-1}(h+\ell, k+\ell)$ can be obtained as a certain product of the σ_i's for $i \in \varphi^{-1}(I^{hk})$. More precisely, we have $\tau_{hk} = (\varphi(k)-1, \varphi(k))(\varphi(k)-2, \varphi(k)-1) \cdots (\varphi(h), \varphi(h)+1)(\varphi(h)+1, \varphi(h)+2) \cdots (\varphi(k)-1, \varphi(k))$.

Theorem 3 *CPS$''$ is a narrowing of CPS.*

Proof. Let $(\bar{x}, \bar{y}, \bar{\alpha}) \in \mathcal{G}$(CPS), and consider the set $\mathscr{V}$ of all constraints $\mathscr{C}_i \equiv \{y_i \leq y_{i+L}\}$ violated by $(\bar{x}, \bar{y}, \bar{\alpha})$. Let ψ be the (invertible) map given by $\psi(\mathscr{C}_i) = (\varphi(i), \varphi(i)+1)$; then $\psi(\mathscr{V})$ is a set of transpositions that can be partitioned into maximal non-disjoint subsets $S^{hk} = \{(\varphi(h), \varphi(h)+1), \ldots, (\varphi(k)-1, \varphi(k))\}$; let $\mathscr{T}$ be the set of pairs (h,k) for which S^{hk} is in the partition of $\psi(\mathscr{V})$. It is easy to verify that if $\pi_{hk} = \prod_{\substack{\ell \in I^{hk} \\ h+\ell L < k-\ell L}} \tau_{h+\ell L, k-\ell L}$ then $\pi_{hk}\bar{y}$ satisfies the constraints in $\psi^{-1}(S^{hk})$. Furthermore, by maximality of the S^{hk}, the permutations π_{hk} are disjoint. Now, if $\pi = \prod_{(h,k) \in \mathscr{T}} \pi_{hk}$, $\pi(\bar{x}, \bar{y}, \bar{\alpha})$ is such that $\pi\bar{y}$ satisfies all constraints in $\mathscr{V}$ and $\pi\bar{x}$ satisfies all constraints in $\bigcup_{i \in \mathcal{N}''} \mathscr{A}_i$ by Prop. 1. Thus $\pi(\bar{x}, \bar{y}, \bar{\alpha}) \in \mathcal{G}(\mathrm{CPS}'')$. □

3 Computational results

We compare Couenne's performance on formulations CPS$'$ and CPS$''$ for some "limit" instances of CPS (i.e. N circles fit in the square but $N+1$ do not). Our comparative results, shown below, have been obtained on a 2.4GHz Intel Xeon CPU

	CPS$'$			CPS$''$		
Inst.	f^*	*nodes*	*tree*	f^*	*nodes*	*tree*
16_4	0.660	2381772	642285	**1**	2795501	839240
25_5	1	461224	188835	1	521487	222846
36_6	0	49962	23784	**1**	76409	34825
49_7	0	12577	6090	**1**	21366	10136
68_8	0	4	1	**0.943**	1057	497
86_9	0	4	1	**0.640**	5	1

with 24 GB RAM running Linux. The table displays the following statistics at termination (10h of CPU time): objective function value f^* of the incumbent, number of BB nodes closed, number of BB nodes still on the tree. The best upper bound at termination was fixed at 2 (and hence the gap was always $> 100\%$) for all reformulations and instances. However, the statistics on the number of nodes show that CPS$''$ is a better reformulation than CPS$'$. The incumbent statistics also show that CPS$''$ behaves better than CPS$'$ when used to derive heuristic solutions.

References

[1] P. Hansen A. Costa and L. Liberti. Formulation symmetries in circle packing. In R. Mahjoub, editor, *ISCO 2010 Proceedings*, Electronic Notes in Discrete Mathematics, Amsterdam, accepted. Elsevier.

[2] 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.

[3] D. Cohen, P. Jeavons, C. Jefferson, K. Petrie, and B. Smith. Symmetry definitions for constraint satisfaction problems. In P. van Beek, editor, *Constraint Programming*, volume 3709 of *LNCS*. Springer, 2005.

[4] S. Kucherenko, P. Belotti, L. Liberti, and N. Maculan. New formulations for the kissing number problem. *Discrete Applied Mathematics*, 155(14):1837–1841, 2007.

[5] L. Liberti. Reformulations in mathematical programming: Definitions and systematics. *RAIRO-RO*, 43(1):55–86, 2009.

[6] L. Liberti. Reformulations in mathematical programming: Symmetry. *Mathematical Programming*, in revision.

[7] L. Liberti. Symmetry in mathematical programming. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*. IMA, Minneapolis, in revision.

[8] L. Liberti, S. Cafieri, and F. Tarissan. Reformulations in mathematical programming: A computational approach. In A. Abraham et al., editors, *Foundations of Computational Intelligence Vol. 3*, number 203 in Studies in Computational Intelligence, pages 153–234. Springer, Berlin, 2009.

[9] M. Locatelli and U. Raber. Packing equal circles in a square: a deterministic global optimization approach. *Discrete Applied Mathematics*, 122:139–166, 2002.

[10] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90, 2002.

[11] F. Margot. Symmetry in integer linear programming. In M. Jünger et al., editors, *50 Years of Integer Programming*, pages 647–681. Springer, Berlin, 2010.

Chromatic index of chordless graphs

R. Machado[a,b], C. Figueiredo[a], and N. Trotignon[c],

[a] *COPPE - Universidade Federal do Rio de Janeiro*

[b] *Instituto Nacional de Metrologia, Normalização e Qualidade Industrial*

[c] *CNRS, LIAFA, Université Paris 7*

Abstract

A graph G is said to be *chordless* if no cycle in G has a chord. Chordless graphs are exactly the graphs whose line graphs are wheel-free, which implies a connection between the study of the chromatic index of chordless graphs and the study of the chromatic number of wheel-free graphs. For example, chordless graphs of maximum degree $\Delta = 3$ are Δ-edge-colourable, and this implies the 3-vertex-colorability of {wheel,ISK_4}-free graphs [11]. In the present work we investigate the chromatic index of chordless graphs with higher degrees. We describe a decomposition result for chordless graphs and use this result to prove that every chordless graph of maximum degree $\Delta \geq 3$ has chromatic index Δ.

Key words: chordless graphs, chromatic index, edge-colouring

1 Introduction

Let $G = (V, E)$ be a simple graph. The maximum degree of a vertex in G is denoted $\Delta(G)$. A *k-edge-colouring* of G is a function $\pi : E \to \{1, 2, ..., k\}$ such that no two adjacent edges receive the same colour $c \in \{1, 2, ..., k\}$. The *chromatic index* of G, denoted by $\chi'(G)$, is the least k for which G has a k-edge-colouring. Vizing's theorem [16] states that $\chi'(G) = \Delta(G)$, and G said to be *Class 1*, or $\chi'(G) = \Delta(G) + 1$, and G said to be *Class 2*. Edge-colouring is NP-complete for regular graphs [8,10] of degree $\Delta \geq 3$. The problem is NP-complete also for the following classes [4]: comparability (hence perfect) graphs, line graphs of bipartite graphs (hence line graphs and clique graphs), {induced k-cycle}-free graphs ($k \geq 3$), cubic graphs of girth k ($k \geq 4$). Graph

classes for which edge-colouring is polynomially solvable include the following: bipartite graphs [9], split-indifference graphs [12], series-parallel graphs (hence outerplanar) [9], k-outerplanar graphs [2] ($k \geq 1$). The complexity of edge-colouring is unknown for several well-studied strong structured graph classes, for which only partial results have been reported, such as cographs [1], join graphs [7], planar graphs [14], chordal graphs, and several subclasses of chordal graphs such as indifference graphs [6], split graphs [5] and interval graphs [3].

Lêvéque, Maffray and Trotignon [11] studied the class of chordless graphs, which are the graphs whose cycles are all chordless. There are two main motivations to study the class. The first motivation is its relation with the the class of wheel-free graphs: chordless graphs are exactly the graphs whose line graphs are wheel-free. Hence, the study of the chromatic index of chordless graphs has importance to the study of the (vertex) chromatic number of wheel-free graphs and subclasses. For example, the 3-vertex-colourability of {wheel, ISK_4}-free graphs is consequence [11] of the fact that chordless graphs of maximum degree 3 are Class 1. The second motivation for the study of the chromatic index of chordless graph is the fact that they are a subclass of the class of graphs that do not contain, as induced subgraph, a cycle with unique chord, called *unichord-free* graphs. The edge-colouring problem is NP-complete when restricted to unichord-fre graphs [13]; hence, it is of interest to determine subclasses of unichord-free graphs for which ege-colouring is polynomial.

2 Structure of chordless graphs

We describe a decomposition result for chordless graphs. This result is used in Section 3 to determine the chromatic index of chordless graphs.

A graph is *strongly 2-bipartite* if it is square-free and bipartite with bipartition (X, Y) where every vertex in X has degree 2 and every vertex in Y has degree at least 3. A graph is *sparse* if every edge is incident to a vertex of degree at most 2. A *cutset* of a graph G is a set of vertices whose exclusion disconnects G. The following cutsets are used in the known decomposition theorems of the class of chordless graphs:

- A *1-cutset* of a connected graph $G = (V, E)$ is a node v such that V can be partitioned into sets X, Y and $\{v\}$, so that there is no edge between X and Y. We say that (X, Y, v) is a *split* of this 1-cutset.
- A *proper 2-cutset* of a connected graph $G = (V, E)$ is a pair of non-adjacent nodes a, b, both of degree at least three, such that V can be partitioned into sets X, Y and $\{a, b\}$ so that: $|X| \geq 2$, $|Y| \geq 2$; there is no edge between X and Y, and both $G[X \cup \{a, b\}]$ and $G[Y \cup \{a, b\}]$ contain an ab-path. We say that (X, Y, a, b) is a *split* of this proper 2-cutset.

A graph is called *basic* if: (1) is cycle with at least four vertices or strongly 2-bipartite; and (2) has no decomposition by 1-cutset or proper 2-cutset.

The *block* G_X (resp. G_Y) of a graph G with respect to a 1-cutset with split (X, Y, v) is $G[X \cup \{v\}]$ (resp. $G[Y \cup \{v\}]$). The *blocks* G_X and G_Y of a graph G with respect to a proper 2-cutset with split (X, Y, a, b) are defined as follows. If a node c has only neighbors a and b in G, then $G_X := G[X \cup \{a, b, c\}]$ (resp. $G_Y := G[Y \cup \{a, b, c\}]$). Otherwise, G_X (resp. G_Y) is obtained from $G[X \cup \{a, b\}]$ (resp. $G[Y \cup \{a, b\}]$) by adding new node c adjacent to a and b.

Theorem 1 [11] describes a decomposition of chordless graphs into sparse graphs. As we state in Theorem 2, sparse graphs can be decomposed by proper 2-cutsets. From Theorems 1 and 2, we state the new decomposition result of Theorem 3. Finally, Theorem 4 proves that a non-basic chordless graph has a decomposition in which (at least) one of the blocks is basic.

Theorem 1 (Lévêque, Maffray and Trotignon [11]) *If G is a chordless graph, then either G is sparse or G admits a 1-cutset or G admits a proper 2-cutset.*

Theorem 2 *If G is sparse graph with $\Delta(G) \geq 3$ then either G is strongly 2-bipartite or G admits 1-cutset or G admits proper 2-cutset.*

Theorem 3 *If G is chordless graph then either G is strongly 2-bipartite or G is cycle on at least 4 vertices or G admits 1-cutset or G admits proper 2-cutset.*

Theorem 4 *If G is a biconnected non-basic chordless graph then G admits proper 2-cutset with split (X, Y, a, b) such that G_X is basic.*

3 Chromatic index of chordless graphs

We apply the structure results of Section 2 to show that every chordless graph of maximum degree at least 3 is Class 1. We show how to $\Delta(G)$-edge-colour a graph $G \in \mathcal{C}'$ by combining $\Delta(G)$-edge-colourings of its blocks with respect to a decomposition by proper 2-cutset.

Lemma 5 *Let G be a chordless graph of maximum degree $\Delta \geq 3$ and let (X, Y, a, b) be a split of proper 2-cutset, in such a way that G_X is basic. If G_Y is Δ-edge-colourable, then G is Δ-edge-colourable.*

Theorem 6 *Every chordless graph of maximum degree $\Delta \geq 3$ is Class 1.*

PROOF. Sketch. Assume G biconnected. If G is basic, then G is strongly 2-bipartite, hence Class 1. If G is not basic, then G has proper 2-cutset with

split (X, Y, a, b) such that G_X is basic. Assume, as induction hypothesis, that G_Y is Δ-edge-colourable. By Lemma 5, graph G is Δ-edge-colourable.

References

[1] M. M. Barbosa, C. P. de Mello, and J. Meidanis. *Local conditions for edge-colouring of cographs*. Congr. Numer. **133** (1998) 45–55.

[2] H. L. Bodlaender. *Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees*. J. Algorithms **11** (1990) 631–643.

[3] V. A. Bojarshinov. *Edge and total colouring of interval graphs*. Discrete Appl. Math. **114** (2001) 23–28.

[4] L. Cai and J. A. Ellis. *NP-Completeness of edge-colouring some restricted graphs*. Discrete Appl. Math. **30** (1991) 15–27.

[5] B. L. Chen, H.-L. Fu, and M. T. Ko. *Total chromatic number and chromatic index of split graphs*. J. Combin. Math. Combin. Comput. **17** (1995) 137–146.

[6] C. M. H. de Figueiredo, J. Meidanis, C. P. de Mello, and C. Ortiz. *Decompositions for the edge colouring of reduced indifference graphs*. Theoret. Comput. Sci. **297** (2003) 145–155.

[7] C. De Simone and C. P. de Mello. *Edge-colouring of join graphs*. Theoret. Comput. Sci. **355** (2006) 364–370.

[8] I. Holyer. *The NP-completeness of edge-colouring*. SIAM J. Comput. **10** (1982) 718–720.

[9] D. S. Johnson. *The NP-completeness column: an ongoing guide*. J. Algorithms **6** (1985) 434–451.

[10] D. Leven and Z. Galil. *NP-completeness of finding the chromatic index of regular graphs*. J. Algorithms **4** (1983) 35–44.

[11] B. Lévêque, F. Maffray, and N. Trotignon. *On graphs with no induced subdivision of K_4*. Preprint available at `www.liafa.jussieu.fr/ trot`.

[12] C. Ortiz, N. Maculan, and J. L. Szwarcfiter. *Characterizing and edge-colouring split-indifference graphs*. Discrete Appl. Math. **82** (1998) 209–217.

[13] R. Machado, C. Figueiredo, and K. Vušković. *Chromatic index of graphs with no cycle with unique chord*. Theoret. Comput. Sci. **411** (2010) 1221–1234.

[14] D. P. Sanders and Y. Zhao. *Planar graphs of maximum degree seven are class I*. J. Combin. Theory Ser. B **83** (2001) 201–212.

[15] N. Trotignon and K. Vušković. *A structure theorem for graphs with no cycle with a unique chord and its consequences*. J. Graph Theory **63** (2010) 31–67.

[16] V. G. Vizing. *On an estimate of the chromatic class of a p-graph* (in russian). Diskret. Analiz **3** (1964) 25–30.

Lattice Polyhedra and Submodular Flows

Satoru Fujishige and Britta Peis

Key words: distributive lattices, Edmonds-Giles polyhedra, submodularity

---

Lattice polyhedra were introduced by Hoffman and Schwartz as a common framework for various discrete optimization problems. They are specified by a ternary matrix whose row set forms a *consecutive, supermodular* lattice and some submodular rank function (the terms "sub"-and "supermodular" can also be interchanged). Though lattice polyhedra are known to be integral, so far no combinatorial algorithm could have been found for the corresponding linear optimization problem. We show that the important class of distributive lattice polyhedra in which the underlying lattice is both, sub-and supermodular can be reduced to Edmonds-Giles polyhedra. Thus, submodular flow algorithms can be applied to this class of lattice polyhedra.

1 Introduction

A large class of discrete optimization problems allow a formulation as integer linear program with underlying ternary matrix: given a matrix $A \in \{-1, 0, 1\}^{L \times E}$, some weight function $w \in \mathbb{R}^E$, lower and upper bounds $c, d \in \mathbb{R}^E$ and some "rank" function $f \in \mathbb{R}^L$ find an integral solution of

$$(LP) \quad \max_{x \in \mathbb{R}^E} \{w^T x \mid Ax \leq f,\ c \leq x \leq d\}.$$

This problem is easily seen to be $\mathcal{NP}$-hard even if restricted to binary matrices. Therefore, we are looking for more special structures of the polyhedron

$$\mathbb{P}(A, f) = \{x \in \mathbb{R}^E \mid Ax \leq f,\ c \leq x \leq d\}.$$

A promising class is that of *lattice polyhedra* which were introduced by Hoffman and Schwartz [HS78] and shown to be integral. The name comes from a certain, very general, lattice structure on A on which f is submodular.

Definition 1 (Lattice polyhedra) *Let $A \in \{-1, 0, 1\}^{L \times E}$ be a matrix with entries $\chi(i, e)$ for $i \in L$ and $e \in E$, and let $c, d \in \mathbb{R}^E$ and $f \in \mathbb{R}^L$. Then the polyhedron*

$$\mathbb{P}(A, f) = \{x \in \mathbb{R}^E \mid Ax \leq f,\ c \leq x \leq d\}$$

is called a lattice polyhedron *if the row index set L forms a lattice $L = (L, \preceq, \wedge, \vee)$ on which f is submodular, i.e., f satisfies*

$$f(i) + f(j) \geq f(i \wedge j) + f(i \vee j) \quad \forall i, j \in L,$$

and where for all $i, j, k \in L$ and all $e \in E$ the following three hold:

(C1) *if $i \prec j \prec k$ and $\chi(i, e) = \chi(k, e) = t \neq 0$, then $\chi(j, e) = t$,*

(C2) *if $i \prec j$, then $\chi(i, e) \cdot \chi(j, e) \geq 0$, and*

(C3) $\chi(i, e) + \chi(j, e) \leq \chi(i \vee j, e) + \chi(i \wedge j, e)$.

Analogously, if, in the above definition, function f is supermodular and (C3) is replaced by

(C3′) $\chi(i, e) + \chi(j, e) \geq \chi(i \vee j, e) + \chi(i \wedge j, e)$,

the polyhedron

$$\mathbb{P}'(A, f) = \{x \in \mathbb{R}^E \mid Ax \geq f,\ c \leq x \leq d\}$$

is also called a lattice polyhedron. The lattice L is called *consecutive* if properties (C1) and (C2) are satisfied. If L satisfies (C3) (or (C3′)), we call it *supermodular* (or *submodular*).

Lattice polyhedra form a common framework for various combinatorial structures such as polymatroids, the intersection of polymatroids, and Edmonds-Giles polyhedra. Several min-max results for combinatorial structures can be derived from the following theorem:

Theorem 1 ([HS78], [H82]) *If f, c and d are integral, then all vertices of lattice polyhedra are integral.*

However, this integrality result is only a structural existence theorem without algorithmic foundation. While several greedy-type algorithms have been developed for special instances of lattice polyhedra in the last decades (see e.g. [**?**], [FP08], [FK96], [FK00], [2], [E70], [3], [DH03], up to now no combinatorial algorithm could have been found for lattice polyhedra in its general form, even if the polyhedra are restricted to binary matrices.

A very important class of lattice polyhedra is that of *distributive lattice polyhedra*, in which the lattice $(L, \preceq, \wedge, \vee)$ is distributive and (C3) is satisfied with equality.

Let us first recall some basic facts about distributive lattices: A lattice $(L, \preceq, \wedge, \vee)$ is called *distributive* if the binary operators $\wedge, \vee$ satisfy the distributive laws. Alternatively, distributive lattices can be characterized by the exclusion of the sublattices N_5 and M_3. By a theorem of Birkhoff, a distributive lattice L is isomorphic to the

lattice $\mathcal{D}(P)$ of all ideals [1] of poset $(P, \preceq)$ on the set P of join-irreducible elements [2] of L. (For more details about lattices the reader is referred to [B91].)

Beside classical examples of combinatorial structures such as polymatroids, the intersection of polymatroids, or submodular systems, distributive lattice polyhedra also cover Edmonds-Giles polyhedra (see below). Furthermore, we show in Theorem 2 below that a large class of lattice polyhedra is in fact distributive. Finally, we show in Theorem 3 that distributive lattice polyhedra can in fact be reduced to Edmonds-Giles polyhedra for which several efficient algorithms exist.

Theorem 2 *Let $\mathbb{P}(A, f)$ be a lattice polyhedron in which any two rows of A are distinct and property (C3) is satisfied with equality. Then the underlying lattice $(L, \preceq, \wedge, \vee)$ is distributive.*

Proof: For the sake of contradiction, assume that L is not distributive, i.e., that it contains an N_5- or an M_3-sublattice. Then there exist five distinct elements $i, j, k, l, m \in L$ such that

$$l = i \wedge j = i \wedge k \quad \text{and} \quad m = i \vee j = i \vee k.$$

Since $\chi(j) \neq \chi(k)$ by the assumption, choose some element $e \in E$ with $\chi(j, e) \neq \chi(k, e)$. Since (C3) is satisfied with equality, it follows that

$$\chi(l, e) + \chi(m, e) = \chi(i, e) + \chi(j, e) = \chi(i, e) + \chi(k, e),$$

which implies $\chi(j, e) = \chi(k, e)$, a contradiction. □

Edmonds-Giles polyhedra. Let $G = (V, E)$ be a connected directed graph and $\mathcal{F} \subseteq 2^V$ be a ring family of subsets of vertex set V, (i.e., $\mathcal{F}$ is union-and intersection-closed). Given a submodular function $f : \mathcal{F} \to \mathbb{R}$ and lower and upper bounds on the edges $c, d : E \to \mathbb{R}$, the Edmonds-Giles polyhedron is

$$\mathbb{P}(G, \mathcal{F}, f) = \{x \in \mathbb{R}^E \mid x(\Delta^+(S)) - x(\Delta^-(S)) \leq r(S)\ \forall S \in \mathcal{F}, c \leq x \leq d\},$$

where $\Delta^+(S)$ and $\Delta^-(S)$ denote, respectively, the sets of arcs leaving S and of entering S. (In the original definition, $\mathcal{F}$ is a crossing family on which f is crossing submodular. However, it suffices to consider the case of ring families with submodular f, as the more general crossing case can be reduced to it using the Dilworth truncation (see e.g. [Fuj91]).) Edmonds and Giles [EG77] proved that $\mathbb{P}(G, \mathcal{F}, f)$ is integral, and several algorithms for the corresponding linear optimization problem, called the *submodular flow problem*, have been established (see e.g., the survey paper [FI00]. Almost all submodular flow algorithms are based on generalizations of different min-cost-flow algorithms).

Also Edmonds-Giles polyhedra turn out to be distributive lattice polyhedra: given an Edmonds-Giles polyhedron $\mathbb{P}(G, \mathcal{F}, f)$ consider the collection of ordered pairs

$$L = \{(\Delta^+(S), \Delta^-(S)) \mid S \in \mathcal{F}\} \subseteq 3^E$$

[1] A subset $I \subseteq P$ is an *ideal* of poset $(P, \preceq)$ if $i \preceq j$ and $j \in P$ implies $i \in P$

[2] An element $i \in L$ is *join-irreducible* if $i = j \vee k$ implies $i = j$ or $i = k$

partially ordered by

$$(\Delta^+(S), \Delta^-(S)) \preceq (\Delta^+(T), \Delta^-(T)) \iff S \subseteq T$$

and with join- and meet-operations

$$\begin{aligned}(\Delta^+(S), \Delta^-(S)) \vee (\Delta^+(S), \Delta^-(S)) &= (\Delta^+(S \cup T), \Delta^-(S \cup T)) \\ (\Delta^+(S), \Delta^-(S)) \wedge (\Delta^+(T), \Delta^-(T)) &= (\Delta^+(S \cap T), \Delta^-(S \cap T)).\end{aligned}$$

Also for all $S \subseteq V$ and $e \in E$ define

$$\chi(S, e) = \begin{cases} 1 & e \in \Delta^+(S) \\ -1 & e \in \Delta^-(S) \\ 0 & \text{otherwise} \end{cases}$$

Then $(L, \preceq, \wedge, \vee)$ with such a χ is a consecutive, sub- and supermodular lattice.

While it seems that the Edmonds-Giles polyhedra form a special class of distributive lattice polyhedra, we will see that they are in fact equivalent, i.e., we show that any distributive lattice polyhedron can be reduced to some Edmonds-Giles polyhedron. For this, we construct an auxiliary digraph G whose vertices correspond to the join-irreducible elements P of L and whose edges correspond to the elements in E. We then show that the lattice polyhedron is equivalent to the Edmonds-Giles polyhedron $\mathbb{P}(G, \mathcal{D}(P), f)$, i.e., we show (in the appendix)

Theorem 3 *If $\mathbb{P}(A, f)$ is a distributive lattice polyhedron, then there exists an auxilary digraph $G = (P, E)$ whose vertices correspond to the join-irreducible elements of L such that*

$$\begin{aligned}\mathbb{P}(A, f) &= \{x \in \mathbb{R}^{E'} \mid \forall I \in \mathcal{D}(P) : x(\Delta^+(I)) - x(\Delta^-(I)) \leq f(I),\ c \leq x \leq d\} \\ &= \mathbb{P}(G, \mathcal{D}(P), f)\end{aligned}$$

References

[B91] G.Birkhoff: *Lattice Theory*. Amer. Math. Soc. **91** (1991).

[DH03] B.L. Dietrich and A.J. Hoffman: *On greedy algorithms, partially ordered sets, and submodular functions*. IBM J. Res. & Dev. 47 (2003), 25-30.

[E70] J. Edmonds: *Submodular functions, matroids, and certain polyhedra*, in: *Combinatorial Structures and Their Applications*, R. Guy *et al.* eds., Gordon and Breach, New York, 1970, 69-87.

[FK96] U. Faigle and W. Kern: *Submodular linear programs on forests*. Math. Programming 72 (1996), 195-206.

[FK00] U. Faigle and W. Kern: *On the core of ordered submodular cost games*. Math. Programming 87 (2000), 483-489.

[FP08] U.Faigle and B. Peis, *Two-phase greedy algorithms for some classes of combinatorial linear programs*, In SODA, 2008, 161-166.

[1] A. Frank: *Increasing the rooted-connectivity of a digraph by one*. Math. Programming 84 (1999), 565-576.

[H82] A.J. Hoffman: *Ordered sets and linear programming*. In: Ordered Sets (editor: I. Rival), D. Reidel Publishing company (1982), 619-654.

[HS78] A. Hoffman and D.E. Schwartz, *On lattice polyhedra*, Proceedings of Fifth Hungarian Combinatorial Coll. (A. Hajnal and V.T. Sos, eds.), North-Holland, Amsterdam, 1978, pp. 593–598.

[EG77] J. Edmonds and R. Giles: *A min-max relation for submodular functions on graphs*. Ann. Discrete Math., **1** (1977), 185-204.

[FI00] S. Fujishige and S. Iwata: *Algorithms for submodular flows*. Special Issue on Algorithm Engineering: Surveys. IEICE Transactions on Informations and Systems, Vol.E83-D, No.3 (2000).

[Fuj91] S. Fujishige: *Submodular Functions and Optimization*. Annals of Discrete Math., **47**, (1991).

[2] U. Krüger: *Structural aspects of ordered polymatroids*. Discr. Appl. Math. 99 (2000), 125-148.

[3] M. Queyranne, F. Spieksma, and F. Tardella: *A general class of greedily solvable linear programs*. Math. Oper. Res. 23 (1998), 892-908.

On the partition dimension of Cartesian product graphs

Ismael G. Yero [a], Juan A. Rodríguez-Velázquez [a] and Magdalena Lemańska [b]

[a] *Department of Computer Engineering and Mathematics Rovira i Virgili University, Av. Països Catalans 26, 43007 Tarragona, Spain*

[b] *Department of Technical Physics and Applied Mathematics Gdansk University of Technology, ul. Narutowicza 11/12 80-233 Gdansk, Poland*

Abstract

Let $G = (V, E)$ be a connected graph. The distance between two vertices $u, v \in V$, denoted by $d(u, v)$, is the length of a shortest $u - v$ path in G. The distance between a vertex $v \in V$ and a subset $P \subset V$ is defined as $min\{d(v, x) : x \in P\}$, and it is denoted by $d(v, P)$. An ordered partition $\{P_1, P_2, ..., P_t\}$ of vertices of a graph G, is a *resolving partition* of G, if all the distance vectors $(d(v, P_1), d(v, P_2), ..., d(v, P_t))$ are different. The *partition dimension* of G, denoted by $pd(G)$, is the minimum number of sets in any resolving partition of G. In this article we show that for all pair of connected graphs G, H, $pd(G \times H) \leq pd(G) + pd(H)$ and $pd(G \times H) \leq pd(G) + dim(H)$. Consequently, we show that $pd(G \times H) \leq dim(G) + dim(H) + 1$.

Key words: Resolving sets, resolving partition, partition dimension, Cartesian product.

1 Introduction

The concepts of resolvability and location in graphs were described independently by Harary and Melter [9] and Slater [16], to define the same structure in a graph. After these papers were published several authors developed diverse theoretical works about this topic [2–8,14]. Also, Slater described the usefulness of these ideas into long range aids to navigation [16]. Recently, these concepts were used by a pharmacy company while attempting to develop a capability of large datasets of chemical graphs [12,13]. Other applications of this concept to navigation of robots in networks and other areas appear in [5,11,14]. Some variations on resolvability or location have been appearing in the literature, like those about conditional resolvability [15], locating domination [10], resolving domination [1] and resolving partitions [4,7,8].

Email address: ismael.gonzalez@urv.cat (Ismael G. Yero).

Given a graph $G = (V, E)$ and a set of vertices $S = \{v_1, v_2, ..., v_k\}$ of G, the *metric representation* of a vertex $v \in V$ with respect to S is the vector $r(v|S) = (d(v, v_1), d(v, v_2), ..., d(v, v_k))$, where $d(v, v_i)$, with $1 \leq i \leq k$, denotes the distance between the vertices v and v_i. We say that S is a *resolving set* of G if for every pair of vertices $u, v \in V$, $r(u|S) \neq r(v|S)$. The *metric dimension*[1] of G is the minimum cardinality of any resolving set of G, and it is denoted by $dim(G)$. The metric dimension of graphs is studied in [2–6,17]. Given an ordered partition $\Pi = \{P_1, P_2, ..., P_t\}$ of the vertices of G, the *partition representation* of a vertex $v \in V$ with respect to the partition Π is the vector $r(v|\Pi) = (d(v, P_1), d(v, P_2), ..., d(v, P_t))$, where $d(v, P_i)$, with $1 \leq i \leq t$, represents the distance between the vertex v and the set P_i, that is $d(v, P_i) = \min_{u \in P_i}\{d(v, u)\}$. We say that Π is a *resolving partition* of G if for every pair of vertices $u, v \in V$, $r(u|\Pi) \neq r(v|\Pi)$. The *partition dimension* of G is the minimum number of sets in any resolving partition of G and it is denoted by $pd(G)$. The partition dimension of graphs is studied in [4,7,8,17]. It is natural to think that the partition dimension and metric dimension are related; in [7] it was shown that for any nontrivial connected graph G we have

$$pd(G) \leq dim(G) + 1. \tag{1}$$

The study of relationships between invariants of Cartesian product graphs and invariants of its factors appears frequently in research about graph theory. In the case of resolvability, the relationships between the metric dimension of the Cartesian product graphs and the metric dimension of its factors was studied in [2,3]. An open problem on the dimension of Cartesian product graphs is to prove (or finding a counterexample) that for all pair of graphs G, H; $dim(G \times H) \leq dim(G) + dim(H)$. In the present paper we study the case of resolving partition in Cartesian product graphs, by giving some relationships between the partition dimension of Cartesian product graphs and the partition dimension of its factors. More precisely, we show that for all pair of connected graphs G, H; $pd(G \times H) \leq pd(G) + pd(H)$ and $pd(G \times H) \leq pd(G) + dim(H)$. Consequently, we show that $pd(G \times H) \leq dim(G) + dim(H) + 1$.

2 Results

Theorem 1 *For any connected graphs G_1 and G_2,*

$$pd(G_1 \times G_2) \leq pd(G_1) + pd(G_2).$$

By (1) we obtain the following direct consequence of Theorem 1.

Corollary 2 *For any connected graphs G_1 and G_2,*

$$pd(G_1 \times G_2) \leq pd(G_1) + dim(G_2) + 1.$$

As we can see below, the above relationship can be improved.

[1] Also called locating number.

Theorem 3 *For any connected graphs G_1 and G_2,*

$$pd(G_1 \times G_2) \leq pd(G_1) + dim(G_2).$$

We note that there are graphs for which Theorem 1 estimates $pd(G_1 \times G_2)$ better than Theorem 3 and vice versa. For example Theorem 1 leads to $pd(K_n \times P_n) \leq n + 2$ while Theorem 3 gives $pd(K_n \times P_n) \leq n + 1$. On the contrary, if G denotes the graph in Figure 1, Theorem 1 leads to $pd(G \times G) \leq 8$ while Theorem 3 gives $pd(G \times G) \leq 13$.

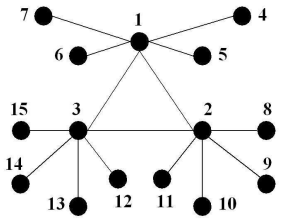


Fig. 1. $\{\{1, 4, 8, 12\}, \{2, 5, 9, 13\}, \{3, 6, 10, 14\}, \{7, 11, 15\}\}$ is a resolving partition of G and $\{4, 5, 6, 8, 9, 10, 12, 13, 14\}$ is a resolving set of G.

As a direct consequence of above theorem and (1) we deduce the following interesting result.

Corollary 4 *For any connected graphs G_1 and G_2,*

$$pd(G_1 \times G_2) \leq dim(G_1) + dim(G_2) + 1.$$

One example of graphs for which the equality holds in Corollary 4 (and also in Corollary 5 (ii)) are the graphs belonging to the family of grid graphs: $pd(P_r \times P_t) = 3$.

Corollary 5 *For any connected graph G,*

(i) $pd(G \times K_n) \leq pd(G) + n - 1$.
(ii) $pd(G \times P_n) \leq pd(G) + 1$.
(iii) $pd(G \times C_n) \leq pd(G) + 2$.
(iv) $pd(G \times K_{1,n}) \leq pd(G) + n - 1$.

Acknowledgments: This work was partly supported by the Spanish Ministry of Science and Innovation through projects TSI2007-65406-C03-01 "E-AEGIS", CONSOLIDER INGENIO 2010 CSD2007-0004 "ARES".

References

[1] R. C. Brigham, G. Chartrand, R. D. Dutton, P. Zhang, Resolving domination in graphs, *Mathematica Bohemica* **128** (1) (2003) 25–36.

[2] J. Caceres, C. Hernando, M. Mora, I. M. Pelayo, M. L. Puertas, C. Seara, D. R. Wood, On the metric dimension of Cartesian product of graphs, *SIAM Journal of Discrete Mathematics* **21** (2) (2007) 273–302.

[3] J. Caceres, C. Hernando, M. Mora, I. M. Pelayo, M. L. Puertas, C. Seara, On the metric dimension of some families of graphs, *Electronic Notes in Discrete Mathematics* **22** (2005) 129–133.

[4] G. Chappell, J. Gimbel, C. Hartman, Bounds on the metric and partition dimensions of a graph, *Ars Combinatoria* **88** (2008) 349–366.

[5] G. Chartrand, L. Eroh, M. A. Jhonson, O. R. Oellermann, Resolvabiblity in graphs ans the metric dimension of a graph, *Discrete Applied Mathematics* **105** (2000) 99–113.

[6] G. Chartrand, C. Poisson, P. Zhang, Resolvability and the upper dimension of graphs, *Computer and Mathematics with Appplications* **39** (2000) 19–28.

[7] G. Chartrand, E. Salehi, P. Zhang, The partition dimension of a graph, *Aequationes Mathematicae* **59** (2000) 45–54.

[8] M. Fehr, S. Gosselin, O. R. Oellermann, The partition dimension of Cayley digraphs *Aequationes Mathematicae* **71** (2006) 1–18.

[9] F. Harary, R. A. Melter, On the metric dimension of a graph, *Ars Combinatoria* **2** (1976) 191–195.

[10] T. W. Haynes, M. Henning, J. Howard, Locating and total dominating sets in trees, *Discrete Applied Mathematics* **154** (2006) 1293–1300.

[11] B. L. Hulme, A. W. Shiver, P. J. Slater, A Boolean algebraic analysis of fire protection, *Algebraic and Combinatorial Methods in Operations Research* **95** (1984) 215–227.

[12] M. A. Johnson, Structure-activity maps for visualizing the graph variables arising in drug design, *Journal of Biopharmaceutical Statististics* **3** (1993) 203–236.

[13] M. A. Johnson, Browsable structure-activity datasets, *Advances in Molecular Similarity* (R. Carb–Dorca and P. Mezey, eds.) JAI Press Connecticut (1998) 153–170.

[14] S. Khuller, B. Raghavachari, A. Rosenfeld, Landmarks in graphs, *Discrete Applied Mathematics* **70** (1996) 217–229.

[15] V. Saenpholphat, P. Zhang, Conditional resolvability in graphs: a survey, *International Journal of Mathematics and Mathematical Sciences* **38** (2004) 1997–2017.

[16] P. J. Slater, Leaves of trees, Proc. 6th Southeastern Conference on Combinatorics, Graph Theory, and Computing, *Congressus Numerantium* **14** (1975) 549–559.

[17] I. Tomescu, Discrepancies between metric and partition dimension of a connected graph, *Discrete Mathematics* **308** (2008) 5026–5031.

On the Design of the Fiber To The Home Networks

Stefano Gualandi, Federico Malucelli, Domenico L. Sozzi

Politecnico di Milano, Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{gualandi,malucell,sozzi}@elet.polimi.it

Key words: LP-based Randomized Rounding, Local Search, Network Design

1 Introduction

Telecommunications network design is the source of many interesting challenges in combinatorial optimization. Among the more recent ones there is the design of the Next Generation Access Networks completely based on fiber cable technology that, in certain cases, may reach single users and for this reason are called Fiber To The Home networks (FTTH). These networks are organized into two levels. In the first level few *central offices* are connected with high capacity fiber cables to *splicing cabinets* usually located at street intersections. Cabinets are then connected with users or houses. The fiber technology allows to have very long connection cables thus few central offices suffice to serve many more users with respect to traditional copper based networks. The new network characteristics and the incumbent deployment, that requires a great extent of investments, motivate the investigations on quantitative optimization models and algorithms for the planning that can help investors to decide which type of fiber network to select and how to operationally implement it. For a review on technical aspects refer to [2].

2 Problem Statement and Formulation

Planning a FTTH network can be seen as a particular case of facility location problem where facilities belong to two levels. Given a set of candidate sites O for central offices, a set of candidate sites C for cabinets and the set of homes to be served S, the problem consists in deciding in which candidates sites install central offices and cabinets, and connect users to central offices

passing through a cabinet. In addition to these decisions the problem considers also the multiplexing capability of cabinets. Depending on the type of device installed in the cabinet, several signals transmitted on fibers to the users can be groomed into a single fiber to the central office thus allowing for a capacity saving in the leg central office-cabinet. The additional decision level is thus the type of multiplexing technology to be installed in each cabinet. Decisions must consider central office and cabinet installation costs, multiplexing technology costs, and cable deployment costs.

Let s^1_i and M^1_i be the cost and the capacity (in terms of number of fibers) of central office i. Let T be the types of technologies that can be installed in cabinets. Multiplexing technology t in a cabinet allows to send m_t channels on a single fiber towards the central office. Let s^2_{jt} be the installation cost of cabinet j with technology t, and M^2_j its maximum capacity in terms of number of fibers coming from the users. With d_{ij} we indicate the known distance (computed on the street graph) between any two sites i and j.

A possible formulation of the problem introduces two sets of binary variables: $y^1_i, i \in O$ whose value is 1 if a central office is activated in site i, and $y^2_{jt}, j \in C, t \in T$ if a cabinet with multiplexing technology t is activated in site j. We need another set of binary variables x^2_{jl} whose value is 1 if basement l is assigned to cabinet j. Integer variables x^1_{ij} give the number of fibers connecting central office i with cabinet j. The last two sets of variables are defined for all pairs i, j and j, l such that the distance between the corresponding sites is less than or equal to the maximum allowed distance. In order to consider only pairs of sites within a feasible distance, we introduce a set E of pairs i, j with $i \in O$ and $j \in C$ such that $d_{ij} \leq L^1$, and a set F of pairs j, l with $j \in C$ and $l \in S$ such that $d_{jl} \leq L^2$.

The Integer Programming model is as follows:

$$\min \quad \sum_{i \in O} s^1_i y^1_i + \sum_{j \in C} \sum_{t \in T} s^2_{jt} y^2_{jt} + \sum_{ij \in E} c^1_{ij} x^1_{ij} + \sum_{jl \in F} c^2_{jl} x^2_{jl} \tag{1}$$

$$\text{s.t.} \quad \sum_{jl \in E} x^2_{jl} = 1, \qquad \forall l \in S, \tag{2}$$

$$\sum_{ij \in E} x^1_{ij} \leq M^1_i y^1_i, \qquad \forall i \in O, \tag{3}$$

$$\sum_{t \in T} y^2_{jt} \leq 1, \qquad \forall j \in C, \tag{4}$$

$$\sum_{jl \in F} x^2_{jl} \leq M^2_j \sum_{t \in T} y^2_{jt}, \qquad \forall j \in C, \tag{5}$$

$$m_t \sum_{ij \in E} x^1_{ij} \geq \sum_{jl \in F} x^2_{jl} - M^2_j (1 - y^2_{jt}), \qquad \forall j \in C, \forall t \in T, \tag{6}$$

$$y^1_i \in \{0,1\}, \forall i \in O, \quad y^2_{jt} \in \{0,1\}, \forall j \in C, \forall t \in T, \tag{7}$$

$$x^1_{ij} \in \mathbb{Z}_+, \forall ij \in E, \quad x^2_{jl} \in \{0,1\}, \forall jl \in F. \tag{8}$$

Constraints (2) state that each user must be connected to a cabinet. Constraints (3) are twofold: they force the activation of central office i (i.e. it sets variable y_i to 1) if at least one cabinet j is assigned to it, and they limit the number of cabinets assigned to i according to the capacity. Constraints (4) determine that either a cabinet is not active (when the left hand side is equal to 0) or at most a multiplexing technology is assigned to it. While constraints (6) relate the number of incoming fibers in a cabinet from users with the number of outgoing fibers towards the central office. This number must account for the multiplexing factor installed in the cabinet. The objective function (1) sums up the cost s_i^1 of each selected central office, the cost s_{jt}^2 for installing the technology t in cabinet j and the connection costs for the fibers between central offices and cabinets and between cabinets and the users.

In order to improve the linear relaxation, we introduce the following constraint:

$$\sum_{t \in T} y_{jt}^2 \leq \sum_{ij \in E} x_{ij}^1, \forall j \in C. \tag{9}$$

that states that if a cabinet is activated it must be connected to a central office. Though the improvement on the lower bound is modest, this constraint does have an impact on our LP-based randomized rounding algorithm.

3 Solution Approaches and Computational Results

We have developed two approaches to solve the FTTH problem. The first approach is a LP-based Randomized Rounding (LP-RR) algorithm, the second is a Constraint-Based Local Search (CBLS) algorithm. Both approaches are implemented exploiting features of the COMET constraint language [3]. For the lack of space, we just briefly sketch the two approaches.

Our LP-RR algorithm, motivated by the results in [1], is based on the observation that once we have decided which central offices and which cabinets are opened, that is, the variables y^1 and y^2 have been fixed to either 1 or 0, the remaining problem is reduced to a generalized minimum cost flow problem on a tripartite graph. So we first randomly round the variables y^1 and y^2, and only then, the x^1 and x^2 variables.

The proposed CBLS approach relies on the use of *invariants* (see [4]) to incrementally maintain the necessary information to guide the search procedure. Once a greedy procedure has computed a feasible solution, we execute a local search algorithm based on a simple move: select the basement l connected to a cabinet j, and select a different open cabinet $j' \neq j$ that is not saturated (it has some capacity left) such that *moving* l from j to j' gives, after the propagation of the new assignments, the best improvement in the objective

Table 1
LP-based Randomized Rounding (LP-RR) versus Constraint-based Local Search (CBLS). Cost and Time (in seconds). Standard deviations omitted.

			LP-RR			CBLS		
$\|O\|$	$\|C\|$	$\|S\|$	Cost	Time	Best-Cost	Cost	Time	Best-Cost
3	10	100	2383	31	**2383**	**2383**	0.6	**2383**
10	35	400	6979	716	6966	6864	1.2	**6860**
15	65	841	13630	1735	13599	13349	44.6	**13306**
20	100	1521	25499	2465	25427	24850	316	**24752**
25	120	3025	55073	4768	55052	51752	330	**51646**
30	140	6084	121794	7705	121974	118224	1105	**118135**
35	150	10000	239668	26915	239668	229677	1817	**229244**

Table 2
Solving big Rome instances with the CBLS approach: gaps computed with respect to the linear relaxation (P).

$\|O\|$	$\|C\|$	$\|S\|$	Cost	(stdev)	Time	(stdev)	Best-Cost	LP-Gap
30	140	5982	4561215	(0.01%)	1803.6	(0.14%)	4560780	0.9%
30	140	5995	4164941	(0.01%)	2168.7	(0.69%)	4164724	1.1%
30	140	6014	3462920	(0.01%)	1426.9	(0.35%)	3462857	1.4%
35	150	10020	3126763	(0.02%)	2511.8	(0.44%)	3126385	2.4%
35	150	10040	5937585	(0.01%)	3484.7	(0.55%)	5936733	1.1%
35	150	10072	6663950	(0.01%)	1183.6	(0.54%)	6663481	0.9%

function. After this move, we possibly increase the number of fibers outgoing cabinet j'.

Tables 1 shows a comparison of the two approaches, reporting computational results averaged over 5 runs for each problem instance. Even if both approaches are interesting, the CBLS outperforms the LP-RR both in quality and computation time. Table 2 reports the results for a set of realistic instances based on the street graph of the city of Rome. Note that CBLS computes near-optimal solution in short time.

References

[1] Barahona, F., Chudak, F.: Near-optimal solutions to large-scale facility location problems. Discrete Optimization **2**(1) (2005) 35–50

[2] Kramer, G., Pesavento, G.: Ethernet Passive Optical Network (EPON): building a next-generation optical access network. IEEE Communications Magazine **40**(2) (2002) 66–73

[3] Van Hentenryck, P., Michel, L.: Control abstractions for local search. Constraints **10**(2) (2005) 137–157

[4] Van Hentenryck, P., Michel, L.: Differentiable invariants. In: Frédéric Benhamou (Ed.), Proc CP-06 LNCS-4204. Springer, (2006) 604–619

A Branch-and-Price Approach to the k-Clustering Minimum Biclique Completion Problem

Stefano Gualandi [a,*] Francesco Maffioli [a] Claudio Magni [b]

[a] *Politecnico di Milano, Dipartimento di Elettronica e Informazione*
Piazza Leonardo da Vinci 32, 20133 Milano, Italy

[b] *SAS Institute, Analytic Innovation Center*
via Darwin 20-22, 20143, Milano

Key words: Biclique, Branch-and-Price, Meta-heuristic

1 Introduction

Given a bipartite graph $G = (S, T, E)$, the k-clustering Minimum Biclique Completion Problem (k-MinBCP) consists of finding k bipartite subgraphs (clusters), such that each vertex i of S appears in exactly one subgraph, every vertex j in T appears in each cluster in which at least one of its neighbors appears, and the total number of edges that would complete each subgraph into a complete bipartite subgraph, i.e., a biclique, is minimized. This problem was introduced in [1], as an application of the problem of bundling channels in multicast transmissions. k-MinBCP is NP-Hard, and its approximability, to the best of our knowledge, remains unknown. In the literature, k-MinBCP is tackled with two approaches: in [1], it is solved with an Integer Programming approach, a Bilinear Programming formulation and its standard linearization; and in [2], it is solved with an hybrid Constraint Programming–Semidefinite programming approach.

In this work, we present a Branch-and-Price algorithm that embeds a new meta-heuristic to find integer solutions, and a non-trivial branching rule. Computational results show that our algorithm outpeforms the state-of-the-art approaches to this problem.

* Corresponding author.
Email addresses: {maffioli,gualandi}@elet.polimi.it, claudio.magni@ita.sas.com (Claudio Magni).

2 Problem Formulation

In this work, we use a Column Generation formulation that is similar to the one proposed in [1]: the master problem is a set partitioning problem where each column represents the subset of vertices of S that induces a cluster t. The cost c_t of each cluster t is equal to the number of edges that would complete the corresponding subgraph into a biclique. Let λ_t be a 0–1 variable, equal to 1 if the cluster t is part of the solution, and 0 otherwise. Let c_t be the cost of the t-th cluster. Let $\mathcal{T}$ be the collection of every possible cluster, and let S_t be the subset of vertices of S that form the t-th cluster. The master problem formulation is as follows:

$$\min \quad \sum_{t \in \mathcal{T}} c_t \lambda_t \tag{1}$$

$$\text{s.t.} \quad \sum_{t \in \mathcal{T} | i \in S_t} \lambda_t = 1 \qquad \forall\, i \in S \tag{2}$$

$$\sum_{t \in \mathcal{T}} \lambda_t = k \tag{3}$$

$$\lambda_t \in \{0, 1\} \qquad \forall\, t \in \mathcal{T} \tag{4}$$

Constraints (2) is the partitioning constraints, one for each vertex in S. Constraint (3) is the cardinality constraint on the number of cluster to be selected. Let π_i and ν be the dual multipliers of constraints (2) and (3), respectively. Then, the pricing subproblem is the problem of finding a vertex– and edge–weighted biclique of negative reduced cost. The vertex weights are given by the dual multipliers π_i, while the edge weights gives the number of edges that would complete the subgraph into a biclique. The pricing subproblem is as follows:

$$\min \quad \sum_{\{i,j\} \in \bar{E}} z_{ij} - \sum_{i \in I} \pi_i x_i - \nu \tag{5}$$

$$\text{s.t.} \quad z_{ij} \geq x_i + x_l - 1 \quad \forall \{i, j\} \in \bar{E},\, \forall \{l, j\} \in E \tag{6}$$

$$x_i, z_{ij} \in \{0, 1\} \qquad \forall i \in I,\, \forall \{i, j\} \in \bar{E} \tag{7}$$

Constraints (6) force the binary variable z_{ij} to be 1 if the corresponding edge is part of the biclique, and 0 otherwise. Note that an edge belongs to a biclique if it exists at least a pair of vertices i and l both in S such that a vertex $j \in T$ exists with $(i, j) \in \bar{E}$ and $(l, j) \in E$.

3 Branch-and-Price Implementation

Differently from [1], that uses the column generation only to compute lower bounds, we have embedded the column generation into a branch-and-price

exact algorithm. The branch-and-price we have implemented is based on three key feature: (i) a Variable Neighborhood Search (VNS) heuristic that computes very efficient primal solution, providing tight upper bounds; (ii) a slightly different VNS heuristic that compute nearly-optimal solutions for the pricing subproblem, and (iii) a non-trivial branching rule.

The VNS heuristic used to find integer solutions, hence yielding upper bounds, explores basically three different neighborhoods: (i) moving a single vertex from one subgraph to another subgraph, (ii) swapping two vertices in two different subgraphs, and (iii) selecting two vertices in two different subgraphs and moving them into new subgraphs. In addition, after each cycle of VNS, we perform a search in the space of the unfeasible solutions, by augmenting by one the number of clusters. Then, a greedy procedure is used to recover a feasible solution. Although the search in the unfeasible space is very simple, it does improve the performance of our meta-heuristic.

In our Column Generation approach to k-MinBCP the bottleneck is the solution of the pricing subproblem. We have implemented two methods for solving the pricing problem. The first method is again a VNS heuristic very similar to the heuristic used to find integer solutions to k-MinBCP: we basically look for a single subgraph with negative reduced cost. Whenever the heuristic is unable to find a negative reduced cost solution, we use an integer programming approach to find any solution of negative reduced cost, not necessarily the solution of minimum cost.

The meta-heuristic and the Column Generation are used to obtain upper and lower bounds within our branch-and-price algorithm. Though many instances are solved at the root node (the upper bounds obtained with our VNS heuristic are equal to the lower bounds obtained by Column Generation), this is not always the case. Therefore, we have devised a branching rule that exploits the problem structure. Once a pair of vertices i and j of S appearing in a fractional solution of the restricted master problem are selected, the algorithm adds two branching constraints: either i and j must appear in the same cluster, or they cannot. In the first branch, we merge the two nodes in a single new node, obtaining a new instance of the same problem. In the second branch, in order to force two vertices to appear in different clusters, we add the corresponding constraints to the pricing subproblem.

4 Computational Results

We tested our branch-and-price algorithm on two classes of instances: the first set of instances consists of random bipartite graphs, and the second set of instances extracted by the MovieLens data set (`http://movielens.umn.edu`).

Table 1

|I|	**|J|**	**k**	**UB**	**LB**	**Opt**	**N**	**Time**	**SCIP**
15	15	3	**72**	**72**	**72**	0	**6.38**s	11s
15	15	4	**59**	**59**	**59**	0	**2.92**s	370s
15	15	5	**50**	**50**	**50**	0	**4.14**s	–
18	18	3	**109**	**109**	**109**	0	**26**s	137s
18	18	4	**96**	**96**	**96**	0	**20**s	–
18	18	5	**86**	85	**86**	8	**47**s	–
20	20	3	**156**	154	**156**	12	**275**s	–
20	20	4	*139*	137	**138**	8	**206**s	–
20	20	5	**123**	121	**123**	38	**175**s	–

The branch-and-price algorithm is implemented in COMET [4], using `lp_solve` as linear solver. Extensive computational results are reported in [3].

Table 1 shows a summary of the comparison of the computational results obtained with our branch-and-price algorithm and with the ILP solver SCIP. The table gives the results for the most challenging instances, that are instances randomly generated with $|S| = |T|$. The first three columns give the $|S|$, $|T|$, and the number of required cluster k. Then, the table reports the **UB** obtained with our VNS heuristic at the root node, the lower bound **LB** obtained via column generation at the root node, the optimal solution **Opt** obtained via Branch-and-Price, the number of branching nodes **N**, and the computation time in seconds. For the SCIP solver, we just report the computation time in seconds. Note that our branch-and-price algorithm is able to solve instances of dimension up to 20×20. In addition, we remark that previous work solved only instances up to 10×10 in [1] and 12×12 in [2].

References

[1] Faure, N., Chrétienne, P., Gourdin, E., Sourd, F.: Biclique completion problems for multicast network design. Discr. Optim. **4**(3) (2007) 360–377

[2] Stefano Gualandi. k-clustering minimum biclique completion via a hybrid CP and SDP approach. In *Proc Integration of AI and OR Techniques in CP for Combinatorial Optimization*, LNCS 5547, pages 87–101. Springer, 2009.

[3] Claudio Magni. Biclique completion problem: models and algorithms. Master Project, Politecnico di Milano, September 2009.

[4] Comet web site, Brown University, `http://comet-online.org`

The game chromatic number of 1-caterpillars

Adrien Guignard [1]

Université de Bordeaux
LaBRI UMR 5800
351, cours de la Libération
F-33405 Talence Cedex, France

Abstract

The game chromatic number of a graph is defined using a two players game. In 1993, Faigle et al. proved that the game chromatic number of trees is at most four. In this paper we investigate the problem of characterizing those trees with game chromatic number three, and setttle this problem for 1-caterpillars.

Key words: game chromatic number, caterpillar, tree, leaf.

1 Introduction

The game chromatic number of a graph G, denoted by $\chi_g(G)$, is defined through a *coloring game* with two players Alice and Bob and a set of k colors. Each move by either player consists of coloring an uncolored node of G with a color i of the set. Adjacent vertices must be colored by distinct colors. The game ends if no more vertices can be colored. Alice wins the game if all vertices are colored. Otherwise, Bob wins.
The game chromatic number $\chi_g(G)$ is the least number of colors for which Alice has a winning strategy in this game. This parameter was introduced by Bodlaender [1] (see also [3] for a recent survey). Since then the problem has attracted considerable attention and has been studied for various classes of graphs [4] [5]. For instance, it is proved by Zhu [6] that if P is a planar graph then $\chi_g(P) \leq 17$. Faigle et al. [2] proved that the game chromatic number of every tree is at most four. A natural question in this framework is to characterize the trees with given game chromatic number k, for $1 \leq k \leq 4$. Since the answer is obvious for $k = 1, 2$, our aim is to characterize the set of trees

[1] Email:guignard@labri.fr

with game chromatic number three. The general characterization seems to be a difficult problem. Therefore, we restrict ourselves to the set of caterpillars and settle the problem for 1-caterpillars.

2 Definitions and properties

A tree is called a *caterpillar* if a path remains after the removal of all its leaves. This path is called the *spine* of the caterpillar. A *1-caterpillar* is a caterpillar such that every vertex of the spine is a neighbor of exactly one leaf, except for the two extremities of the spine that have two leaves.

Since the game chromatic number of a caterpillar C is at most 4, we will play with three colors and determine whether Alice can complete the coloration of C or not. If it is possible, we will say that Alice wins and otherwise that Bob wins. We denote by $[C, c]$ a caterpillar C equipped with a partial coloring c of its vertices. Such a caterpillar will be simply denoted by C whenever the partial coloring c is clear from the context. To compute the game chromatic number of caterpillars, we have to know for any $[C, c]$ not only whoever wins if Alice begins, but also if Bob begins.

We thus define the *outcome* of a partially colored caterpillar C, denoted by $o(C)$, as follows:

(1) $o(C) = \mathcal{B}$ if **Bob** wins whoever starts the game;
(2) $o(C) = \mathcal{P}$ if the next player loses (so the **Previous** one wins);
(3) $o(C) = \mathcal{N}$ if the **Next** player wins;
(4) $o(C) = \mathcal{A}$ if **Alice** wins whoever starts the game.

We denote by $Opt(C)$ the set of options of C, that is the set of partially colored caterpillars that can be obtained from C after one move. If C has an option with outcome $\mathcal{X}$, we say that C has an $\mathcal{X}$*-option*. We extend the definition of outcome to a set of caterpillars $\mathcal{C}$: $o(\mathcal{C}) = \{o(C),\ C \in \mathcal{C}\}$.

Proposition 2.1 *Let C be a not totally colored caterpillar.*

(1) $o(C) = \mathcal{B} \Leftrightarrow o(Opt(C)) \in \Big\{\{\mathcal{B}\}\ ,\ \{\mathcal{N}, \mathcal{B}\}\Big\}$;
(2) $o(C) = \mathcal{P} \Leftrightarrow o(Opt(C)) = \{\mathcal{N}\}$;
(3) $o(C) = \mathcal{N} \Leftrightarrow$ $o(Opt(C))$ *contains* $\mathcal{P}$*, or contains* $\mathcal{A}$ *and* $\mathcal{B}$;
(4) $o(C) = \mathcal{A} \Leftrightarrow o(Opt(C)) \in \Big\{\{\mathcal{A}\}\ ,\ \{\mathcal{A}, \mathcal{N}\}\Big\}$.

Since we will need to consider outcomes of disjoint unions of caterpillars, we need some properties to compute $o(C_1 \cup C_2)$ according to $o(C_1)$ and $o(C_2)$

(where $C_1 \cup C_2$ denotes the disjoint union of C_1 and C_2).
We call the *signed outcome* of C the outcome $\mathcal{X}$ of C signed by the parity of the number of uncolored nodes of C, denoted by $o'(C) = \mathcal{X}_0$ or $\mathcal{X}_1$.

Proposition 2.2 *Firstly, if $o(C_1) = \mathcal{B}$ or $o(C_2) = \mathcal{B}$ then $o(C_1 \cup C_2) = \mathcal{B}$. Otherwise, we define an addition function of signed outcomes, denoted by "$\oplus$", that satisfies $o'(C_1 \cup C_2) = o'(C_1) \oplus o'(C_2)$, and is given by the following table:*

$\oplus$	$\mathcal{P}_0$	$\mathcal{P}_1$	$\mathcal{N}_0$	$\mathcal{N}_1$	$\mathcal{A}_0$	$\mathcal{A}_1$
$\mathcal{P}_0$	$\mathcal{P}_0$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{N}_1$	$\mathcal{P}_0$	$\mathcal{N}_1$
$\mathcal{P}_1$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{B}_1$	$\mathcal{N}_0$	$\mathcal{P}_1$	$\mathcal{N}_0$
$\mathcal{N}_0$	$\mathcal{B}_0$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{B}_1$	$\mathcal{N}_0$	$\mathcal{N}_1$
$\mathcal{N}_1$	$\mathcal{N}_1$	$\mathcal{N}_0$	$\mathcal{B}_1$	$\mathcal{B}_0$	$\mathcal{N}_1$	$\mathcal{N}_0$
$\mathcal{A}_0$	$\mathcal{P}_0$	$\mathcal{P}_1$	$\mathcal{N}_0$	$\mathcal{N}_1$	$\mathcal{A}_0$	$\mathcal{A}_1$
$\mathcal{A}_1$	$\mathcal{N}_1$	$\mathcal{N}_0$	$\mathcal{N}_1$	$\mathcal{N}_0$	$\mathcal{A}_1$	$\mathcal{A}_0$

We note $\preceq$ the relation on the set of outcomes $\{\mathcal{B}, \mathcal{P}, \mathcal{N}, \mathcal{A}\}$ such that:

(1) $\mathcal{A} \preceq \mathcal{N} \preceq \mathcal{B}$
(2) $\mathcal{P}$ and every another outcome $\mathcal{X}$ are incomparables.

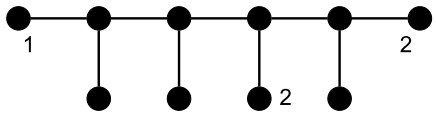
If T and U are two partially colored caterpillars, we say that T is a subgraph of U and note $T \subseteq U$ if and only if $V(T) \subseteq V(U)$, $E(T) \subseteq E(U)$ and $\forall v \in V(T)$, $c_T(v) = c_U(v)$ (where $c_T(v)$ is the color of v in T).

Proposition 2.3 *Let T and U be two partially colored caterpillars. If $T \subseteq U$ then $o(T) \preceq o(U)$*

Let C be a partially colored caterpillar. Observe that if C has an uncolored vertex v having three neighbours colored with distinct colors, then the outcome is $\mathcal{B}$ (since v cannot be colored). Similary, if C has a colored node v with degree $k \geq 2$, the forest C' obtained from C by splitting v into k colored leaves with the same color than v, each linked to a neighbour of v (thus creating k connected components) is equivalent to C (we mean $o'(C') = o'(C)$). Finally, if C has an uncolored node v with two leaves colored with the same color, the caterpillar C' obtained from C by deleting one of these two leaves is equivalent to C.

3 The family of 1-caterpillars

Let C be a partially colored 1-caterpillar whose spine $s_1 s_2 \dots s_\ell$ contains no colored vertices. Moreover let s_0 (resp. $s_{\ell+1}$) be one of the two leaves connected to s_1 (resp. s_ℓ). The leaves connected to s_1 or s_ℓ are the *ends* of C. We associate with C a word $w(C) = w_0 \dots w_{\ell+1}$ on the alphabet $\{z, 1, 2, 3\}$ defined as follows: w_0 is the color of s_0 if it is colored or z otherwise (the same is true of $w_{\ell+1}$), and for every i, $1 \leq i \leq \ell$, w_i is the color of the leaf connected to s_i, or z if this leaf is not colored. For instance, the following caterpillar is associated with the word $1zz2z2$.



Using properties of outcomes and subgraphs, we prove the following results.

Thorme 3.1 *Let C be a 1-catepillar with $w(C) = az^n b$ and $a, b \in \{1, 2, 3\}$. The outcome of C is given by the following table:*

n	1	2	3	4	5	6	7	8	9
$o(C)$	$\mathcal{AN}$	$\mathcal{A}$	$\mathcal{N}$	$\mathcal{A}$	$\mathcal{N}$	$\mathcal{A}$	$\mathcal{N}$	$\mathcal{AN}$	$\mathcal{N}$
n	10	11	12	13	14	15	16	17	≥ 18
$o(C)$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{N}$	$\mathcal{NB}$	$\mathcal{N}$	$\mathcal{B}$	$\mathcal{NB}$	$\mathcal{B}$

where $\mathcal{XY}$ stands for $\mathcal{X}$ if $a = b$ and $\mathcal{Y}$ otherwise.

The proof relies on several lemmas which consider the outcomes of some 1-caterpillars with particular partial colorings. We proceed by studying a family $\mathcal{F}$ of caterpillars (for instance $az^{2n}bb$ with $n \geq 0$) and compute the value n defined as the smallest size of a caterpillar with outcome $\mathcal{N}$ (resp. $\mathcal{B}$), so that every smaller caterpillar in $\mathcal{F}$ has outcome $\mathcal{A}$ (resp. $\mathcal{A}$ or $\mathcal{N}$). We also have to check that no 1-caterpillar of this family has outcome $\mathcal{P}$.

Thorme 3.2 *Let C an uncolored 1-caterpillar with n nodes of degree 3.*

(1) If $n \geq 28$ then $o(C) = \mathcal{B}$
(2) If $23 \leq n \leq 27$ then $o(C) = \mathcal{N}$
(3) Otherwise $o(C) = \mathcal{A}$

References

[1] H. L. Boadlaender, *On the complexity of some coloring games*, Int. J. Found Comput. Sci. **2**(2) (1991), 133–147.

[2] U. Faigle, U. Kern, H. A. Kierstead and W. T. Trotter, *On the game chromatic number of some classes of graphs*, Ars Combin **35** (1993), 143–150.

[3] T. Bartnicky, J. Grytczuk, H. A. Kierstead and X. Zhu, *The map coloring game*, Am. Math. Monthly, **114** (2007), 793–803.

[4] H. A. Kierstead and W. T. Trotter, *Planar graph coloring with an uncooperative partner*, J. Graph Theory **18**(6) (1994), 569-584.

[5] D. Guan and X. Zhu, *The game chromatic number of outerplanar graphs*, J. Graph Theory **30**(1999), 67–70.

[6] X. Zhu, *Refined activation strategy for the marking game*, J. Combin. Theory Ser. B **98**(1) (2008), 1–18

On Hamiltonian cycles through prescribed edges of a planar graph

Jochen Harant

Department of Mathematics, Technical University of Ilmenau, D-98684 Ilmenau, Germany

Key words: planar graph, hamiltonian cycle, prescribed edges

We use [3] for terminology and notation not defined here and consider finite simple graphs only.
The first major result on the existence of hamiltonian cycles in graphs embeddable in surfaces was by H. Whitney [12] in 1931, who proved that 4-connected maximal planar graphs are hamiltonian. In 1956, W.T. Tutte [10,11] generalized Whitney's result from maximal planar graphs to arbitrary 4-connected planar graphs. Actually, Tutte proved that a 4-connected planar graph G has a hamiltonian cycle through any two edges of a given face of G. Moreover, in [7,8] it is proved that a 4-connected planar graph G has a hamiltonian cycle through any three edges of a given face of G or that face is a 3-gon.
Improving a result of C. Thomassen [9], in 1997, D.P. Sanders [7] proved the following:

Theorem 1 *([7]) Every 4-connected planar graph on at least three vertices has a hamiltonian cycle through any two of its edges.*

In [6] the connectivity of a subset X of the vertex set of a graph G is defined as follows. Let G be a graph, $X \subseteq V(G)$, and $G[X]$ be the subgraph of G induced by X. A set $V \subset V(G)$ *splits* X if the graph $G - V$ obtained from G by removing V contains at least two components each containing a vertex of X. Let $\kappa(X)$ be infinity if $G[X]$ is complete, or the minimum cardinality of a set $V \subset V(G)$ splitting X. Let us remark that G is k-connected if and only if $\kappa(V(G)) \geq k$.

Theorem 2 is a local version of Theorem 1 and in case $X = V(G)$, Theorem 1 follows from Theorem 2. It is proven in [6].

Theorem 2 *([6]) If G is a planar graph, $X \subseteq V(G)$, $|X| \geq 3$, $\kappa(X) \geq 4$, $E \subset E(G[X])$, and $|E| \leq 2$, then G contains a cycle C with $X \subseteq V(C)$ and $E \subset E(C)$.*

The following theorem is proven in [4] and, unlike Theorem 2, it is appropriable if $|E| \geq 3$.

Theorem 3 *([4], Theorem 6) If G is a graph, $X \subseteq V(G)$, $E \neq \emptyset$ is a set of independent edges of $G[X]$, $|X| \geq 2|E| + 1$, and $\kappa(X) \geq |X| - |E|$, then G contains a cycle C with $X \subseteq V(C)$ and $E \subset E(C)$.*

Note that Theorem 3 holds for arbitrary graphs. Obviously $|X| \geq 2|E|$ since E is a set of independent edges of $G[X]$.[1] If $X = V(G)$, $|X| - |E| \geq 6$, and G is planar then Theorem 3 cannot be used since a planar graph is at most 5-connected.

We call a maximal planar graph G a plane triangulation if G is embedded into the plane. In [1] it was shown that there are 4-connected plane triangulations containing seven faces of arbitrary distance apart such that each hamiltonian cycle of that graph misses at least one of these faces, i.e. seven edges cannot be guaranteed to belong to a hamiltonian cycle of a 4-connected planar graph even if their pairwise distance is large. Theorem 1 is best possible in the sense that even three prescribed edges need not belong to a hamiltonian cycle of a 4-connected maximal planar graph. Given two edges xy and uv of a graph G, the number of edges of a shortest path in G connecting a vertex of $\{x, y\}$ and a vertex of $\{u, v\}$ is called the *distance* of xy and uv.

Theorem 4 *([5]) There is a 4-connected plane triangulation G containing $E \subseteq E(G)$ with $3|E| = |E(G)|$ such that each hamiltonian cycle of G contains exactly two edges of E. Moreover, for given integer $k \geq 1$, G and E can be chosen such that E contains three edges of pairwise distance at least k.*

The situation changes in comparison with Theorem 4, if the connectivity of G is increased and the pairwise distance of the edges in the set E is at least three. In this case it is even possible to forbid edges of E to belong to a hamiltonian cycle as described in the following Theorem 5. A proof is given in [2].

Theorem 5 *([2]) Let G be a 5-connected plane triangulation and E be a set of edges of G such that the distance between any two edges of E is at least three. Furthermore, let $E = E_1 \cup E_2$ with $E_1 \cap E_2 = \emptyset$. Then G has a hamiltonian cycle C with $E_1 \subset E(C)$ and $E_2 \cap E(C) = \emptyset$.*

Theorem 5 does not hold if the 5-connected plane graph G is not a triangulation. Moreover, the existence of a cycle satisfying the assertion of Theorem

[1] The inequality $|X| \geq 2|E| + 1$ is needed in the proof of Theorem 3 because of technical reasons. Probably Theorem 3 also holds in case $|X| \geq 2|E|$. If in Theorem 3, additionally, G is assumed to be planar, then it remains an open problem whether the inequality $\kappa(X) \geq |X| - |E|$ can be weakened.

5 cannot be guaranteed in the case $|E| \geq 5$ and $E_2 = \emptyset$, because for each integer k there is a 5-connected plane graph G containing a set $E \subset E(G)$ with $|E| = 5$ such that any two edges of E have distance at least k and there is no cycle of G containing E.

For three edges of a 5-connected plane triangulation the distance condition in Theorem 5 can be omitted as follows.

Theorem 6 *([5]) Let G be a 5-connected plane triangulation and E be a set of three edges of G such that E does not form a facial cycle and there is no vertex incident with all edges of E. Then G has a hamiltonian cycle containing E.*

Considering 5-connected maximal planar graphs, the following theorem is an analogue to Theorem 4.

Theorem 7 *([5]) Let G be a 5-connected plane triangulation containing an independent set V of vertices such that each face of G is incident with exactly one vertex of V and $H = G - V$ is 3-regular. Then each hamiltonian cycle of G contains exactly $\frac{1}{3}|E(H)| - 2 = \frac{1}{3}(|V(G)| - 8)$ edges of H.*

Let e_k be the smallest integer l such that there is a 5-connected plane triangulation G containing l edges of pairwise distance at least k such that there is no hamiltonian cycle of G containing all these l edges. If e_k does not exist then we write $e_k = \infty$.

Theorem 5 implies that $e_k = \infty$ if $k \geq 3$.

Theorem 8 *([5]) For e_1 the inequalities $4 \leq e_1 \leq 9$ hold.*
Moreover, there are infinitely many 5-connected maximal planar graphs G containing a set E of $\frac{1}{3}|E(G)|$ independent edges such that each hamiltonian cycle of G misses two edges of E.

It remains open whether e_2 is finite or not.

References

[1] T. Böhme, J. Harant, On Hamiltonian Cycles in 4- and 5-connected Plane Triangulations, *Discrete Mathematics*, 191(1998),25-30.

[2] T. Böhme, J. Harant, M. Tkáč, On Certain Hamiltonian Cycles in Planar Graphs, *Journal of Graph Theory* 32(1999)81-96.

[3] R. Diestel, Graph Theory, Springer, Graduate Texts in Mathematics 173(2000).

[4] T. Gerlach, J. Harant, On a Cycle through a Specified Linear Forest of a Graph, *Discrete Mathematics*, 307(2007)892-895.

[5] F. Göring, J.Harant, Hamiltonian cycles through prescribed edges of 4-connected maximal planar graphs, *Discrete Mathematics* 310(2010) 1491-1494.

[6] J. Harant, S. Senitsch, A Generalization of Tutte's Theorem on Hamiltonian Cycles in Planar Graphs, *Discrete Mathematics* 309(2009)4949-4951.

[7] D.P. Sanders, On Paths in Planar Graphs, *Journal of Graph Theory* 24(1997)341-345.

[8] R. Thomas, X. Yu, Projective Planar Graphs are Hamiltonian, *J. Combin. Theory* Ser. B 62(1994)114-132.

[9] C. Thomassen, A Theorem on Paths in Planar Graphs, *Journal of Graph Theory* 7(1983)169-176.

[10] W.T. Tutte, A Theorem on Planar Graphs, *Trans. Amer. Math. Soc.* 82(1956)99-116.

[11] W.T. Tutte, Bridges and Hamiltonian Circuits in Planar Graphs, *Aequationes Math.*15(1977)1-33.

[12] H. Whitney, A Theorem on Graphs, *Ann.Math.* 32(1931)378-390.

Some bounds on alliances in trees

Ararat Harutyunyan

Department of Mathematics, Simon Fraser University
8888 University Drive, Burnaby, BC, Canada
aha43@sfu.ca

Key words: alliances, global defensive alliance, global offensive alliance, trees, complete k-ary trees

1 Introduction

The study of alliances in graphs was first introduced by Hedetniemi, Hedetniemi and Kristiansen [4]. They introduced the concepts of defensive and offensive alliances, global offensive and global defensive alliances and studied alliance numbers of a class of graphs such as cycles, wheels, grids and complete graphs. Haynes et al. [2] studied the global defensive alliance numbers of different classes of graphs. They gave lower bounds for general graphs, bipartite graphs and trees, and upper bounds for general graphs and trees. Rodriquez-Velazquez and Sigarreta [8] studied the defensive alliance number and the global defensive alliance number of line graphs. A characterization of trees with equal domination and global strong defensive alliance numbers was given by Haynes, Hedetniemi and Henning [3]. Rodriguez-Velazquez and Sigarreta [5] gave bounds for the defensive, offensive, global defensive, global offensive alliance numbers in terms of the algebraic connectivity, the spectral radius, and the Laplacian spectral radius of a graph. They also gave bounds on the global offensive alliance number of cubic graphs in [6] and the global offensive alliance number for general graphs in [7].

Balakrishnan et al. [1] studied the complexity of global alliances. They showed that the decision problems for global defensive and global offensive alliances are both NP-complete for general graphs.

Given a simple graph $G = (V, E)$ and a vertex $v \in V$, the *open neighborhood* of v, $N(v)$, is defined as $N(v) = \{u : (u, v) \in E\}$. The *closed neighborhood* of v, denoted by $N[v]$, is $N[v] = N(v) \cup \{v\}$.

Definition 1 *A set $S \subset V$ is a* defensive alliance *if for every $v \in S$, $|N[v] \cap$*

$S| \geq |N(v) \cap (V - S)|$. *A defensive alliance S is called a* global defensive alliance *if S is also a dominating set.*

Definition 2 *A set $S \subset V$ is an* offensive alliance *if for every $v \in V - S$, $|N[v] \cap S| \geq |N[v] - S|$. An offensive alliance S is called a* global offensive alliance *if S is also a dominating set.*

Definition 3 *The* global defensive(offensive) alliance number *of G is the cardinality of a minimum size global defensive(offensive) alliance in G, and is denoted by $\gamma_a(G)(\gamma_o(G))$. A minimum size global defensive(offensive) alliance is called a $\gamma_a(G)$-set ($\gamma_o(G)$-set).*

In this paper, we study the global defensive and global offensive alliance numbers of trees. We find the asymptotic order of global defensive alliance number of complete k-ary trees, and compute exactly the global offensive alliance number. We also give a sharp bound on the difference between the global offensive and global defensive alliance numbers for a general tree.

The rest of the paper is organized as follows. In Section 2, we find the global defensive alliance number of complete binary and complete ternary trees. We also find tight bounds for the global defensive alliance number of complete k-ary trees, and determine the asymptotic order. In Section 3, we find the global offensive alliance number of complete k-ary trees. We also compare the global offensive and global defensive alliance numbers of a general tree.

2 Defensive Alliances in Complete k-ary Trees

A *k-ary tree* is a rooted tree where each node has at most k children. A *complete k-ary tree* is a k-ary tree in which all the leaves have the same depth and all the nodes except the leaves have k children. We let $T_{k,d}$ be the complete k-ary tree with depth/height d. The proofs of the following theorems are omitted.

Theorem 1 *Let n be the order of $T_{2,d}$. Then $\gamma_a(T_{2,d}) = \lceil \frac{2}{5}n \rceil$ for any d.*

Corollary 1 *If $d \equiv 2(mod\ 4)$ or $d \equiv 3(mod\ 4)$ then there is a unique $\gamma_a(T_{2,d})$-set. If $d \equiv 0(mod\ 4)$ or $d \equiv 1(mod\ 4)$ then there are exactly two $\gamma_a(T_{2,d})$-sets.*

Theorem 2 *If $d \geq 4$ then $\gamma_a(T_{3,d}) = \lfloor \frac{19}{36}n \rfloor$ if d is odd and $\gamma_a(T_{3,d}) = \lceil \frac{19}{36}n \rceil$ if d is even.*

Theorem 3 $\frac{27}{64}n - 2 \leq \gamma_a(T_{4,d}) \leq \frac{27}{64}n + 2$.

When k is large, the methods used to prove the above theorems are difficult to apply. Therefore, for general k, we give upper and lower bounds for $\gamma_a(T_{k,d})$.

Theorem 4 *For $d \geq 2$, and $k \geq 2$,*

$$k^{d-1}\left\lfloor \frac{k-1}{2} \right\rfloor + k^{d-1} + k^{d-2} \leq \gamma_a(T_{k,d}) \leq k^{d-1}\left\lfloor \frac{k-1}{2} \right\rfloor + k^{d-1} + k^{d-2} + k^{d-3}.$$

It follows that $\gamma_a(T_{k,d}) \sim k^{d-1}\left\lfloor \frac{k-1}{2} \right\rfloor$, where the asymptotics is taken to be in terms of k. Since the number of vertices of $T_{k,d}$ is $n = \frac{k^{d+1}-1}{k-1}$ we get $\gamma_a(T_{k,d}) \sim \frac{n}{2}$ when k tends to infinity. For offensive alliances we have the following result, the proof of which is omitted.

3 Offensive Alliances vs. Defensive Alliances in general trees

Theorem 5 *Let $T_{k,d}$ be the complete k-ary tree with depth $d \geq 1$. Then,* $\gamma_o(T_{k,d}) = \left\lfloor \frac{n}{k+1} \right\rfloor$.

Note that $\gamma_o(T_{k,d}) \sim \frac{n}{k}$ with respect to k. As k becomes very large the difference between $\gamma_a(T_{k,d})$ and $\gamma_o(T_{k,d})$ approaches $n/2$. In general, we are interested if this difference can be larger for other trees. In fact, we have the following theorem.

Theorem 6 *For any tree T of order n, $\gamma_a(T) \leq \gamma_o(T) + \frac{n}{2}$.*

Proof 1 *Root the tree T at a vertex of largest* eccentricity *(the eccentricity of a vertex x is equal to $max_{y \in V(G)} d(x,y)$). Let T have a depth d, and let v be a vertex at depth $d-2$. Let u be v's parent. We are going to proceed by induction on n. We may assume that $diam(T) \geq 3$. Otherwise, T is a star and the theorem holds (this also establishes the base case).*
Let T_v be the subtree of T rooted at vertex v. Let $T' = T - T_v$ be the subtree of T obtained by removing all the vertices of T_v, and let $|T'| = n'$. Define P to be the set of children of v in T which are support vertices. Denote by L the set of children of v which are leaves. By assumption on the diameter of T, $|P| \geq 1$. Let y_i denote the number of children of each vertex in P, $1 \leq i \leq |P|$. The proofs of the following two claims are omitted due to space restrictions.

Claim 1 $\gamma_o(T') \leq \gamma_o(T) - |P|$.

Claim 2 $\gamma_a(T) \leq \gamma_a(T') + k$ *where* $k = 1 + |P| + max\left(\lceil \frac{|L|-|P|}{2} \rceil, 0\right) + \sum_{i=1}^{|P|} \lfloor \frac{y_i - 1}{2} \rfloor$.

By the last claim and the induction hypothesis we have

$$\gamma_a(T) \leq \gamma_a(T') + k \leq \gamma_o(T') + \frac{n'}{2} + k.$$

What is left to prove is that $\gamma_o(T') + \frac{n'}{2} + k \leq \gamma_o(T) + \frac{n}{2}$. *By the first claim, it is sufficient to prove that* $k - \lfloor \frac{n-n'}{2} \rfloor \leq |P|$. *Since* $|P| \geq 1$, *we have that*

$$1+max\left(\left\lceil \frac{|L|-|P|}{2} \right\rceil, 0\right)+\sum_{i=1}^{|P|}\left\lfloor \frac{y_i-1}{2} \right\rfloor \leq 1+\left\lfloor \frac{|L|}{2} \right\rfloor+\sum_{i=1}^{|P|}\left\lfloor \frac{y_i-1}{2} \right\rfloor \leq \left\lfloor \frac{n-n'}{2} \right\rfloor,$$

as required.

The above bound is best possible. Consider $K_{1,n-1}$ where n is odd. Then $\gamma_o(K_{1,n-1}) = 1$ and $\gamma_a(K_{1,n-1}) = 1 + \frac{n-1}{2}$.

In a bipartite graph, each partite set forms a global offensive alliance. It follows that $\gamma_o(T) \leq \frac{n}{2}$ for any tree T. Therefore, $|\gamma_a(T) - \gamma_o(T)| \leq \frac{n}{2}$. However, we believe the following stronger result is true: for any n-vertex tree T, $\gamma_o(T) \leq \gamma_a(T) + \frac{n}{6}$. This conjecture, if true, is essentially best possible because of the following theorem, the proof of which we omit.

Theorem 7 *For any constant* $C > 0$, *there exists an* n*-vertex tree* T *with* $\gamma_o(T) \geq \gamma_a(T) + \frac{n}{6} - C$.

References

[1] H. Balakrishnan, A. Cami, N. Deo, and R. D. Dutton, On the complexity of finding optimal global alliances, J. Combinatorial Mathematics and Combinatorial Computing, Volume 58 (2006), 23-31.

[2] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning, Global defensive allliances in graphs, Electronic Journal of Combinatorics 10 (2003), no. 1, R47.

[3] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning, A characterization of trees with equal domination and global strong alliance numbers, Utilitas Mathematica, Volume 66(2004), 105-119.

[4] S. M. Hedetniemi, S. T. Hedetniemi, and P. Kristiansen, Alliances in graphs, Journal of Combinatorial Mathematics and Combinatorial Computing, Volume 48 (2004), 157-177.

[5] J. A. Rodriguez-Velazquez, J.M. Sigarreta, Spectal study of alliances in graphs, Discussiones Mathematicae Graph Theory 27 (1) (2007) 143-157.

[6] J. A. Rodriquez-Velazquez and J. M. Sigarreta, Offensive alliances in cubic graphs, International Mathematical Forum Volume 1 (2006), no. 36, 1773-1782.

[7] J. A. Rodriguez-Velazquez, J.M. Sigarreta, Global Offensive Alliances in Graphs, Electronic Notes in Discrete Mathematics, Volume 25 (2006), 157-164.

[8] J. A. Rodriguez-Velazquez, J. M. Sigarreta, On defensive alliances and line graphs. Applied Mathematics Letters, Volume 19 (12) (2006) 1345-1350.

The Inverse 1-Median Problem in $\mathbb{R}^d$ with the Chebyshev-Norm *

Johannes Hatzl

Institute of Optimization and Discrete Mathematics
Graz University of Technology
Steyrergasse 30, 8010 Graz, Austria

Key words: 1-median problem, inverse location problem, fractional b-matching problem, parametric flow problem

1 Introduction

This paper focuses on the weighted 1-median problem in $\mathbb{R}^d$ where the distance of two points is measured by the Chebyshev-norm. So far this problem is only well understood for $d = 2$. In this case, a linear time algorithm is given in Hamacher [2]. In this note, we give the first combinatorial algorithm for $d \geq 3$. Furthermore, we discuss an optimality criterion for the d-dimensional case which is based on linear programming. Using this optimality criterion we are able to solve the inverse location problem. In the inverse problem the facility is already given and the task is to modify the weights of the points at minimum cost such that the given facility is a 1-median with respect to the new weights.

2 The 1-median problem

The 1-median problem discussed in this note is defined as follows: Given n points $P_1, \ldots, P_n$ with $P_i = (x_1^i, \ldots, x_d^i) \in \mathbb{R}^d$ for $i = 1, \ldots, n$ and associated non-negative weights w_i the task is to find a point $P^* = (x_1^*, \ldots, x_d^*) \in \mathbb{R}^d$

* This research has been supported by the Austrian Science Fund (FWF) Project P18918-N18

Email address: hatzl@opt.math.tugraz.at (Johannes Hatzl).

such that

$$\sum_{i=1}^{n} w_i \, \|P_i - P\|_\infty \geq \sum_{i=1}^{n} w_i \, \|P_i - P^*\|_\infty$$

for all $P \in \mathbb{R}^d$. Note that using some straightforward techniques the problem $\min_{P=(y_1,\ldots,y_d)\in\mathbb{R}^d} \sum_{i=1}^{n} w_i \, \|P_i - P\|_\infty$ can be written as a linear programming problem in the following form:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} w_i \, z_i & & & (1)\\
\text{s.t.} \quad & z_i + y_j \geq x_j^i & i = 1,\ldots,n, \;\; & j = 1,\ldots,d & (2)\\
& z_i - y_j \geq -x_j^i & i = 1,\ldots,n, \;\; & j = 1,\ldots,d & (3)\\
& y_j \in \mathbb{R}, \;\; z_i \in \mathbb{R} & j = 1,\ldots,d, \;\; & i = 1,\ldots,n. & (4)
\end{aligned}$$

Due to the fact that the variables y_j do not appear in the objective function, we use the well known Fourier-Motzkin elimination to get rid of these variables in the constraints. We obtain the following equivalent problem

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} w_i \, z_i & & \\
\text{s.t.} \quad & z_i + z_k \geq d^{ik} & i = 1,\ldots,n \;\; & k = 1,\ldots,n\\
& z_i \geq 0 & i = 1,\ldots,n &
\end{aligned}$$

where $d^{ik} := \max_j |x_j^i - x_j^k|$. It is easy to see that the corresponding dual problem is the linear relaxation of the maximum-weight-b-matching problem on a complete graph. It is shown in Antsee [1] that this graphtheoretical problem can be solved by a min-cost-flow problem in a bipartite graph.

3 The Inverse Problem

An instance of the inverse problem is given by a set of n points $P_1, \ldots, P_n \in \mathbb{R}^d$ with corresponding non-negative weights $w_i \geq 0$ and a point P_0 (which may coincide with a given point). The task is to find new weights $\tilde{w}_i \geq 0$ such that P_0 is a 1-median with respect to $\tilde{w}_i$ and $\|w - \tilde{w}\|_1$ is minimized. The problem can be formulated in a compact form as follows:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} |w_i - \tilde{w}_i| & & \\
\text{s.t.} \quad & \sum_{i=1}^{n} \tilde{w}_i \, \|P_i - P\|_\infty \geq \sum_{i=1}^{n} \tilde{w}_i \, \|P_i - P_0\|_\infty & \forall P \in \mathbb{R}^d & \\
& \tilde{w}_i \geq 0 & & i = 1,\ldots,n.
\end{aligned}$$

Before we give a combinatorial algorithm we state the following lemma.

Lemma 1 *There exists an optimal solution w^* of the inverse location problem such that $w_i \geq w_i^*$ holds for all $i = 1, \ldots, n$.*

Let us consider the dual linear programming problem of (1)—(4). We introduce non-negative dual variables $u_{i,j}$ for the constraints (2) and non-negative dual variables $v_{i,j}$ for the constraints (3) and obtain

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} \sum_{j=1}^{d} x_j^i \left(u_{i,j} - v_{i,j}\right) & \\
\text{s.t.} \quad & \sum_{j=1}^{d} \left(u_{i,j} + v_{i,j}\right) = w_i & i = 1, \ldots, n \\
& \sum_{i=1}^{n} u_{i,j} = \sum_{i=1}^{n} v_{i,j} & j = 1, \ldots, d \\
& u_{i,j} \geq 0, \;\; v_{i,j} \geq 0 & i = 1, \ldots, n, \;\; j = 1, \ldots, d.
\end{aligned}$$

In order to interpret the dual problem let us construct the following flow problem: Consider the bipartite graph $G = (V_1 \cup V_2, E)$ where the set V_1 consists of n vertices, one for each point P_i of the 1-median problem. The set V_2 has exactly $2d$ vertices representing the closed cones

$$Q_j^{\geq} := \{x \in \mathbb{R}^d : x_j \geq 0 \text{ and } |x_j| \geq |x_k| \, \forall k = 1, \ldots, d\}$$

and

$$Q_j^{\leq} := \{x \in \mathbb{R}^d : x_j \leq 0 \text{ and } |x_j| \geq |x_k| \, \forall k = 1, \ldots, d\}$$

for $j = 1, \ldots, d$. We have an edge $(P_i, Q_j^{\sim})$ if P_i is in the cone $Q_j^{\sim}$ ($\sim \in \{\leq, \geq\}$). Moreover, we set the capacity $u(e)$ of the edges of the bipartite graph to infinity. Furthermore, we add a source s and the edges (s, P_i) for all $i = 1, \ldots, n$ with $u(s, P_i) = w_i$. Finally, we introduce a sink t and an edge from each vertex in V_2 to t with infinite capacity. We denote this network by $I(P_1, \ldots, P_n, w)$. Furthermore, a flow in $I(P_1, \ldots, P_n, w)$ is called perfect if the flow on the edges $(Q_j^{\geq}, t)$ and $(Q_j^{\leq}, t)$ is equal for all j. The value of a flow is denoted by $v(f)$. Now we can state the following theorem.

Theorem 2 *Suppose we are given n points $P_1, \ldots, P_n \in \mathbb{R}^d$ with non-negative weights $w_i \geq 0$. Then, the origin $P^* = (0, \ldots, 0)$ is an optimal solution of the 1-median problem if and only if there exists a perfect flow f in $I(P_1, \ldots, P_n, w)$ such that $v(f) = \sum_{i=1}^{n} w_i$.*

Example 3 *Suppose we are given the following points: $P_1 = (-1, 3)$, $P_2 = (2, 2)$, $P_3 = (4, -1)$, $P_4 = (2, -2)$, $P_5 = (0, -2)$ and $P_6 = (-4, 1)$ with the weights $w_1 = 3$, $w_2 = 1$, $w_3 = 1$, $w_4 = 2$, $w_5 = 2$ and $w_6 = 3$. Then, the corresponding instance $I(P_1, \ldots, P_n, w)$ of the balancing flow problem admits a perfect flow f with $v(f) = \sum_{i=1}^{n} w_i$ (see Figure 1). Thus, we can conclude that the origin is a 1-median.*

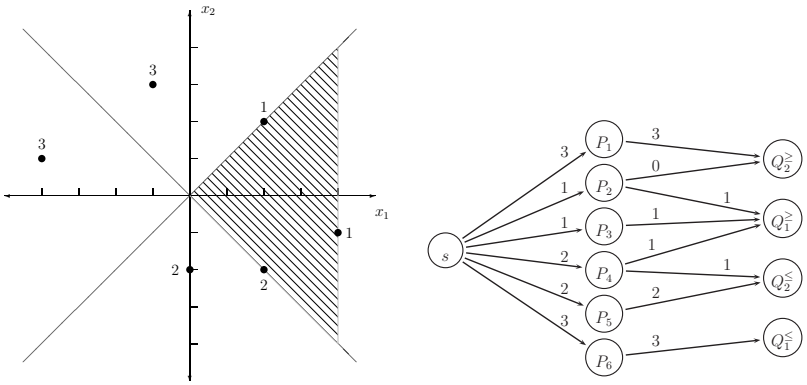


Fig. 1. On the left hand side we are given an instance of the 1-median problem where the cone $Q_1^{\geq}$ is highlighted. The right hand side shows the graph of the balancing flow problem (without the supersink t). On the edges a perfect flow is given.

It is easy to see that the inverse problem can be solved by the following algorithm:

Algorithm 1 *Step 1: Construct the instance* $I(P_1, \ldots, P_n, w)$
Step 2: Find a perfect flow f *in* $I(P_1, \ldots, P_n, w)$ *that maximizes* $v(f)$
Step 3: The new weights are given by $w_i^* = f(s, P_i)$

The main step is obviously the computation of a maximum perfect flow in Step 2. The maximum perfect flow problem can be reformulated as a parametric flow problem, where capacities on the edges $(Q_j^{\geq}, t)$ and $(Q_j^{\leq}, t)$ are given by a parameter λ_j. If we maximize

$$2\sum_{j=1}^{d} \lambda_j$$

such that there exists a flow f that saturates all the edges entering the supersink t, the flow f is a maximum perfect flow. This maximization problem is closely related to parametric flow problems discussed in McCormick [3] and can indeed be solved in polynomial time.

References

[1] R. Antsee, A polynomial algorithm for b-matchings: an alternative approach. *Inform. Process. Lett.* (1987), 153–157.

[2] H. Hamacher, Mathematische Lösungsverfahren für planare Standortprobleme. Braunschweig/Wiesbaden: Vieweg (1995).

[3] T. McCormick, Fast algorithms for parametric scheduling come from extensions to parametric maximum flow. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (1996), 319–328.

Graph Models and their Efficient Implementation for Sparse Jacobian Matrix Determination ⋆

Shahadat Hossain [a,*] and Trond Steihaug [b]

[a] *Department of Mathematics and Computer Science, University of Lethbridge, Canada*

[b] *Department of Informatics, University of Bergen, Norway*

Abstract

Algorithms for solving large-scale combinatorial scientific computing problems arising in sparse or otherwise structured matrix computation are often represented by appropriate graph models and sometimes the same problem can be formulated in more than one graph models with similar asymptotic computational complexity. The relative merits of different graph models for the same problem can then be expressed in terms of factors such as generality of the model and ease of computer implementation. In this note we briefly review the contemporary graph formulations for large-scale sparse Jacobian matrix determination problem (JMDP) and suggest the pattern graph model which can be viewed as a unifying framework for the unidirectional and bidirectional approaches for JMDP. Due to the irregular memory access pattern combined with low floating point calculations relative to the volume of data movements the actual running time of sparse matrix and graph algorithms may achieve only a small fraction of the theoretical performance. We proffer the use of array-based data structures as the basic building-blocks for efficient implementation of fundamental graph algorithms on modern cache-based computer architectures. Numerical results comparing our implementation (DSJM toolkit) with ColPack [4] is given.

Key words: Sparse Matrix Data Structures, Intersection Graph, Bipartite Graph, Hypergraph, Pattern Graph

1 Introduction

We consider the problem of determining the Jacobian matrix $F'(x)$ of a mapping $F : \Re^n \mapsto \Re^m$. In this paper graphs are undirected. The colon notation of [6] is used to specify sections of a matrix. The (i, j)th entry is denoted by

⋆ This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Research Council of Norway (NFR).

* Corresponding author.

$A(i,j)$ or a_{ij}. When a matrix is displayed the nonzero entries are explicitly shown while a blank or a '0' marks a zero entry. We assume that the sparsity pattern of matrix $F'(x)$ is known a priori and that it is computationally more economical to compute the entire column of $F'(x)$ than computing individual entries. Using differences the jth column of the Jacobian matrix may be approximated as

$$\left.\frac{\partial F(x+ts)}{\partial t}\right|_{t=0} = F'(x)s \approx As = \frac{1}{\varepsilon}[F(x+\varepsilon s) - F(x)] \equiv b \tag{1}$$

with one extra function evaluation. Also Algorithmic Differentiation (AD) [7] forward mode gives $b = F'(x)s$ at a cost which is a small multiple of the cost of one function evaluation. The Jacobian matrix determination problem can be stated as below.

Problem 1 (JMDP) *Obtain vectors $s_i \in \Re^n, i = 1, \ldots, p$ and $w_j \in \Re^m, j = 1, \ldots, q$ 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.*

In absence of any sparsity information, one may use the Cartesian basis vectors $e_i, i = 1, \ldots, n$ in (1) using n extra function evaluations. However, if the columns are structurally orthogonal i.e. no two columns have nonzero entries in the same row position only one extra function evaluation

$$F'_j + F'_k \approx A(:,j) + A(:,k) = \tfrac{1}{\varepsilon}[F(x + \varepsilon(e_j + e_k)) - F(x)]$$

is sufficient to read off the nonzero entries from the product $b = As$. If the columns can be partitioned into p structurally orthogonal groups then the Jacobian matrix is *directly determined* from the compressed matrix $B = AS$. Similarly, the rows can be partitioned into q structurally orthogonal groups and the Jacobian can be directly determined from $C^T = W^T A$ using the reverse mode of AD. The problem of finding minimum cardinality orthogonal column (or row) partition of matrix A can be solved as different vertex coloring problems (which in general are NP-hard and therefore are solved by heuristics) of suitable graph(s) associated with A.

2 Graph Models and Computer Implementation

With regard to the choice of the graph model for the partitioning problem we expect that the graph formulation

(1) retains exploitable matrix structures,
(2) enables efficient implementation of pertinent algorithms, and
(3) generic enough to encapsulate the combined row-and-column determination as in Problem JMDP.

The *intersection graph* of the columns of A is denoted by $G(A) = (V, E)$ where corresponding to $A(:, j), j = 1, 2, \ldots, n$, there is a vertex $v_j \in V$ and $\{v_j, v_l\} \in E$ if and only if $A(:, j)$ and $A(:, l), l \neq j$ have nonzero elements in the same row position. Then an orthogonal partition of the columns of A is equivalent to a coloring ϕ of the vertices of $G(A)$ such that $\phi(u) \neq \phi(v)$ if and only if $\{u, v\} \in E$ [1]. Unfortunately, the column intersection graph of A or A^T is unable to expose all the exploitable matrix sparsity. Alternative formulations define graphs based on column segments to allow for better utilization of available structure or sparsity [8]. However, these alternative formulations apply to either column direction or row direction but not to a combination of row and column directions – henceforth *bidirectional determination*.

The *bipartite graph* associated with matrix A is denoted by $G_b(A) = (V_c \cup V_r, E)$ where corresponding to $A(:, j), j = 1, 2, \ldots, n$, there is a column vertex $v_j \in V_c$ and corresponding to $A(i, :), i = 1, 2, \ldots, m$, there is a row vertex $v_i \in V_r$ and $\{v_i, v_j\} \in E$ if and only if $a_{ij} \neq 0, i = 1, 2, \ldots, m, j = 1, 2, \ldots, m$. The bipartite graph model has been proposed independently by Coleman and Verma [2] and Hossain and Steihaug [9] in connection with bidirectional determination of Jacobian matrices. In [3] the bipartite graph model has been considered for column partitioning. The zero-nonzero structure of the underlying matrix is accurately represented in its bipartite graph making it a natural logical data structure for bidirectional determination. On the other hand, for the unidirectional determination the model is "asymmetrical" in the sense that it contains extraneous information. This difficulty is manifested in [3] where the unidirectional determination posed as distance-2 coloring needed the qualification "partial" as only one set of vertices are colored.

A *hypergraph* is a graph in which edges are generalized as *hyperedges* where a hyperedge may connect more than vertices. The hypergraph model presented here is more general than the ones considered in [3]. The hypergraph $H(A) = (V, E)$ associated with the matrix A has the vertex set

$$V = \{v_i | \exists k \text{ for which } a_{ik} \neq 0, i = 1, 2, \ldots, m\} \cup \{v_j | \exists k \text{ for which } a_{kj} \neq 0, j = 1, 2, \ldots, n, \}$$

and corresponding to each row i and each column j the hyperedges $e_i \in E$ and $e_j \in E$, respectively, are defined by

$$e_i = \{v_k | a_{ik} \neq 0\} \text{ and } e_j = \{v_k | a_{kj} \neq 0\}.$$

A *lateral neighbor* of $a_{ij} \neq 0$ is a nonzero $a_{ij'} \neq 0$ in row i of A such that $j' - j$ is the smallest if $j' > j$ or such that $j - j'$ is the smallest if $j > j'$ among all such indices j' in row i. Vertical neighbors can be interpreted in an analogous way with the roles of i and j interchanged. The *sparsity-pattern graph* (or simply the pattern graph) associated with A is $G_{\mathcal{P}}(A) = (V, E)$, where

$$V = \{v_{ij} : a_{ij} \neq 0, i = 1, 2, \ldots, m, j = 1, 2, \ldots, n\}$$

and

$$\{v_{ij}, v_{i'j'}\} \in E \text{ if } a_{ij} \text{ and } a_{i'j'} \text{ are lateral or vertical neighbors.}$$

An unknown a_{ij} is said to be *covered* (by S or W) if it can be uniquely solved in

$$\hat{S_i}^T A(i, \mathcal{J}_i)^T = B(i,:)^T \text{ or } A(\mathcal{I}_j, j)\hat{W_j} = C(:, j).$$

where $\hat{S}_i(\hat{W}_j)$ is the submatrix of $S(W)$ corresponding to the nonzero entries in row i (column j) indicated by $\mathcal{J}_i(\mathcal{I}_j)$. The matrices S and W are said to constitute a cover for A if each $a_{ij} \neq 0$ is covered. A cover is a *direct cover* if A can be determined directly from the cover. Given a mapping $\Phi : V \mapsto \mathcal{S} \cup \mathcal{W}$ where $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_p\}, \mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_q\}$ we define matrices $S \in \{0,1\}^{n \times p}$ and $W \in \{0,1\}^{m \times q}$,

$$S(:, k) = \sum_j e_j, \mathcal{S}_k = \Phi(v_{ij}) \text{ and } W(:, l) = \sum_i e_i, \mathcal{W}_l = \Phi(i).$$

The mapping $\Phi : V \mapsto \mathcal{S} \cup \mathcal{W}$ is said to yield a cover for A if the matrices S and W constitute a cover for A. Denote by $u_{ij} \overset{\geq 1}{\approx} u_{i'j'}$ a path of length at least 1.

Theorem 2 *Let $G_{\mathcal{P}}(A) = (V, E)$ be the pattern graph associated with A. Define the mapping $\Phi : V \mapsto \mathcal{S} \cup \mathcal{W}$ where $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_p\}, \mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_q\}$ such that for each $v_{ij} \in V$,*

EITHER

(1) (a) $v_{ij} \overset{\geq 1}{\approx} v_{ij'}, j \neq j'$ implies $k \neq k'$ where $\Phi(v_{ij}) = \mathcal{S}_k, \Phi(v_{ij'}) = \mathcal{S}_{k'}$ and

(b) $v_{ij} \overset{\geq 1}{\approx} v_{ij'} \overset{\geq 1}{\approx} v_{i'j'}, i \neq i', j \neq j'$ implies $k \neq k'$ where $\Phi(v_{ij}) = \mathcal{S}_k, \Phi(v_{i'j'}) = \mathcal{S}_{k'}$

OR

(2) (a) $v_{ij} \overset{\geq 1}{\approx} v_{i'j}, i \neq i'$ implies $k \neq k'$ where $\Phi(v_{ij}) = \mathcal{W}_k, \Phi(v_{i'j}) = \mathcal{W}_{k'}$ and

(b) $v_{ij} \overset{\geq 1}{\approx} v_{i'j} \overset{\geq 1}{\approx} v_{i'j'}, i \neq i', j \neq j'$ implies $k \neq k'$ where $\Phi(v_{ij}) = \mathcal{W}_k, \Phi(v_{i'j'}) = \mathcal{W}_{k'}$.

Then matrices S and W constitute a direct cover for A.

One of the strengths of the pattern graph model is that the result given above can be specialized to unidirectional and column segments determinations without changing the graph.

The sparsity pattern of matrix A can be efficiently represented using two arrays: array `colind` that stores the column indices of the nonzero entries row-by-row, and array `rowptr` that contains the index of the first nonzero element of each row of the sparse matrix stored in `colind` array. For easy access to the adjacent vertices sparsity pattern of A^T is explicitly stored using analogous arrays `rowind` and `colptr` . This storage is of order $\Theta(\max(n, m, nnz))$ which meets the design strategies for sparse linear algebra implementation [5]. Next we consider the computations on this data structure relevant to algorithms for Problem JMDP. In many graph coloring heuristics on static graphs a frequently executed operation is to find the neighbors of a given vertex v_j.

Assuming v_j a column vertex this information is obtained easily as

$$\{\{v_j, v_i\}|i = \text{ rowind(k), k = colptr(j) : colptr(j+1)-1 }\}.$$

This computation can be performed independent of the graph models discussed above. Further, the array representation ensures better cache performance and the computational cost of the operation is proportional to the size of the data accessed and the number of nonzero arithmetic operations. Clearly, it is not necessary to compute the intersection graphs explicitly as advocated in [3]. Indeed, almost all the well-known coloring heuristics for JMDP can be implemented in time proportional to $\sum_{i=1}^{m} \rho_i^2$ where ρ_i denotes the number of nonzero entries in row i of A. As an indication of the efficiency of the data structure proposed here we report in the table below the timing experiments for incidence degree order (IDO) coloring where `ot` and `ct` represent ordering and coloring times, respectively. `ColPack` implements IDO coloring with partial distance-2 coloring scheme on bipartite graph while `DSJM` implements IDO coloring using the data structure described here using column intersection graph. The times reported here do not include data structure set up times. The sparse matrices are obtained from University of Florida Sparse Matrix Collection.

Matrix	m	n	nnz	ColPack [4]		DSJM	
				ot	ct	ot	ct
lpcreb	9648	77137	260785	13.9	1.49	2.46	1
lpcred	8926	73948	246614	14.79	1.48	2.46	1.01
lpfit2d	25	10524	129042	64.19	24.94	16.32	17.2
lpken18	105127	154699	358171	15.92	0.63	1.47	0.48
lposa07	1118	25067	144812	161.76	28.34	40.84	18.35

References

[1] T. F. Coleman and J. J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM J. Numer. Anal.*, 20(1):187–209, 1983.

[2] T. F. Coleman and A. Verma. The efficient computation of sparse Jacobian matrices using automatic differentiation. *SIAM J. Sci. Comput.*, 19(4):1210–1233, 1998.

[3] A. H. Gebremedhin, F. Manne, and A. Pothen. What Color Is Your Jacobian? Graph Coloring For Computing Derivatives. *SIAM Review*, 47(4):629–705.

[4] A. H. Gebremedhin, A. Tarafdar, D. Nguyen, and A. Pothen. ColPack. http://www.cs.odu.edu/~dnguyen/dox/colpack/html/ (accessed May 2009)

[5] J. R. Gilbert, S. Reinhardt, and V. Shah. High-performance graph algorithms from parallel sparse matrices. In B. Kågström, E. Elmroth, J. Dongarra, and J. Wasniewski, editors, *PARA*, volume 4699 of *Lecture Notes in Computer Science*, pages 260–269. Springer, 2006.

[6] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[7] A. Griewank, Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, Penn., 2008.

[8] Shahadat Hossain and Trond Steihaug. Graph coloring in the estimation of sparse derivative matrices: Instances and applications, *Discrete Applied Mathematics*, 156(2):280–288, 2008.

[9] A. S. Hossain and T. Steihaug. Computing a sparse Jacobian matrix by rows and columns. *Optimization Methods and Software*, 10:33–48, 1998.

G. Jäger

Computer Science Institute, University of Kiel, D-24118 Kiel, Germany

An Effective SAT Encoding for Magic Labeling

Key words: Magic Labeling, Boolean Satisfiability, Backtracking.

1 Introduction

This work presents a Boolean satisfiability (SAT) encoding for a special problem from combinatorial optimization. In the last years much progress has been made in the optimization of practical SAT solvers (see the SAT competition [5]). This has made SAT encodings for combinatorial problems highly attractive. In this work we propose an encoding for the combinatorial problem *Magic Labeling* which has important applications in the field of wireless networks [4]. It is defined as follows. Let an undirected, unweighted graph $G = (V, E)$ be given with vertex set V and edge set E, where $|V| = n$ and $|E| = m$. A *labeling* is a one-to-one mapping $\lambda : V \cup E \to \{1, 2, \ldots, m+n\}$. Define the *weight* $\omega(e)$ of an edge $e \in E$ as the sum of the label of e and of the labels of its two endpoints. An *edge-magic total labeling (EMTL)* is a labeling λ for which a constant $h \in \mathbb{N}$ exists such that $\omega(e) = h$ for each edge $e \in E$. Similarly, define the *weight* $\omega(v)$ of a vertex $v \in V$ as the sum of the label of v and of the labels of all edges incident to v. A *vertex-magic total labeling (VMTL)* is a labeling λ for which a constant $k \in \mathbb{N}$ exists such that $\omega(v) = k$ for each vertex $v \in V$. Finally, a *totally magic labeling (TML)* is a labeling λ for which (not necessarily equal) constants $h, k \in \mathbb{N}$ exist such that λ is edge-magic with constant h and vertex-magic with constant k. h and k are called *magic constants*. A vertex $v \in V$ and an edge $e \in E$ are denoted *neighboring*, if e is incident to v. Note that different EMTLs exist for the same graph, and the same holds for VMTLs. Surveys of results for magic graphs are given in [4]. We consider the following three problems for a given graph:
(1) Does an EMTL exist with given magic constant $h \in \mathbb{N}$?
(2) Does a VMTL exist with given magic constant $k \in \mathbb{N}$?
(3) Does a TML exist with given magic constants $h, k \in \mathbb{N}$?

The obvious method for these problems would be to use a backtracking approach. In Section 2 we propose a general algorithm based on a SAT encoding and extend

Email address: gej@informatik.uni-kiel.de (G. Jäger).

the encoding and the resulted algorithm to the following problems:
(4) Enumerate all EMTLs with given magic constant $h \in \mathbb{N}$?
(5) Enumerate all VMTLs with given magic constant $k \in \mathbb{N}$?
(6) Enumerate all TMLs with given magic constants $h, k \in \mathbb{N}$?
In Section 3 we compare the performance of this SAT based algorithm and of a backtracking algorithm for the problems (1) and (2).

2 SAT Encoding for Magic Labeling

In this section we consider the problems (1) to (3). Let $G = (V, E)$ with $|V| = n$, $|E| = m$ and $r := n + m$. For our convenience, we define a fixed ordering on the set $V \cup E$ by the numbers $1, 2, \ldots, n + m$, i.e., each number of $\{1, 2, \ldots, n + m\}$ represents an edge or a vertex of the graph. For the encoding we use r^2 Boolean variables $x_{i,j}$ with $1 \leq i, j \leq r$, where we set $x_{i,j} = \begin{cases} \textsc{True}, & \text{if edge/vertex } i \text{ receives label } j \\ \textsc{False}, & \text{if edge/vertex } i \text{ does not receive label } j \end{cases}$
Labeling Clauses: For receiving a feasible labeling we need the following conditions. First each edge/vertex needs to have exactly one label. This leads to the condition that for $i = 1, 2, \ldots, r$ exactly one $j \in \{1, 2, \ldots, r\}$ exists with $x_{i,j} = \textsc{True}$. Second each label has to be used by exactly one edge/vertex. This leads to the condition that for $j = 1, 2, \ldots, r$ exactly one $i \in \{1, 2, \ldots, r\}$ exists with $x_{i,j} = \textsc{True}$. All $2r$ restrictions have the same structure, namely that exactly one of the r involved Boolean variables is set to True and the rest to False. To represent this, we introduce $2r^2$ auxiliary variables $y_1, y_2, \ldots, y_{2r^2}$, with r y's for one restriction. W.l.o.g., consider the first restriction, which contains the Boolean variables $x_{1,1}, x_{1,2}, \ldots, x_{1,r}$, and the corresponding auxiliary variables $y_1, y_2, \ldots, y_r$. For $1 \leq k \leq r$ we use y_k to represent that at least one of $x_{1,1}, x_{1,2}, \ldots, x_{1,k}$ is True. Precisely, the y variables are defined as $y_1 = x_{1,1}$ or equivalently $(\neg x_{1,1} \vee y_1) \wedge (x_{1,1} \vee \neg y_1)$, and $y_k = x_{1,k} \vee y_{k-1}$ or equivalently $(y_k \vee \neg x_{1,k}) \wedge (y_k \vee \neg y_{k-1}) \wedge (\neg y_k \vee x_{1,k} \vee y_{k-1})$ for $k = 2, 3, \ldots, r$. In addition, we need to enforce that only one $x_{1,i}$ with $1 \leq i \leq r$ can be True. This means, if $x_{1,k}$ is True, none of the x_{1i} for $1 \leq i < k \leq r$ can be True. This is formulated as $\neg y_{k-1} \vee \neg x_{1k}$ for $k = 2, \ldots, r$. Finally y_r must be True.
Magic Clauses: Furthermore we have to add clauses which ensure that the conditions of EMTL/ VMTL/TML are fulfilled. The following two conditions occur:
EMTL/TML: Set $l := 2$. For given $h \in \mathbb{N}$ and a given edge the sum of $l+1$ labels (namely the label of the edge and of its l endpoint vertices) equals h.
VMTL/TML: For given $k \in \mathbb{N}$ and a given vertex with degree $l \in \mathbb{N}$ the sum of $l + 1$ labels (namely the label of the vertex and of its l incident edges) equals k.
Observe that both conditions have the following structure: For given constants $c, l \in \mathbb{N}$ the sum of $l+1$ labels equals c. For $l \in \mathbb{N}$ let W be the set containing all possible l-tuples $\overrightarrow{w} = (w_1, w_2, \ldots, w_l)$ with $w_i \in \{1, 2, \ldots, r\}$ for $1 \leq i \leq l$ and $w_i \neq w_j$ for $1 \leq i < j \leq l$. Now let a constant $c \in \mathbb{N}$ be given and an edge or

vertex f with corresponding $l \in \mathbb{N}$, i.e., if f is an edge, then $l = 2$, and otherwise l is the degree of f. We want to fulfill the magic condition for f. This means that the sum of the label of f and of its neighboring elements is c. Let $f_1, f_2, \ldots, f_l$ be the neighboring elements of f. For this l compute the set W (which is easy for small l) and choose an arbitrary element $\overrightarrow{w} \in W$ with $w := \sum_{i=1}^{l} w_i$. Then label $f_1, f_2, \ldots, f_l$ by $w_1, w_2, \ldots, w_l$, and consider the four cases:
Case 1**:** $i \in \{1, 2, \ldots, l\}$ exists with $c-w = w_i$; **Case** 2**:** $w \geq c$; **Case** 3**:** $w < c-r$; **Case** 4**:** Otherwise.
As all labels are different and are contained in the set $\{1, 2, \ldots, r\}$, it is clear that for the Cases 1, 2, or 3 no labeling of f exists such that the sum of the labels of $f, f_1, f_2, \ldots, f_l$ is c. In these cases we add the clause $\neg x_{f_1,w_1} \vee \neg x_{f_2,w_2} \vee \cdots \vee \neg x_{f_l,w_l}$ meaning that labeling $f_1, f_2, \ldots, f_l$ by $w_1, w_2, \ldots, w_l$ is not possible. For Case 4 such a labeling is possible, but only if f is labeled with $c - w$. This leads to the clause $\neg x_{f_1,w_1} \vee \neg x_{f_2,w_2} \vee \cdots \vee \neg x_{f_l,w_l} \vee x_{f,c-w}$. Thus for each $\overrightarrow{w} \in W$ we have an additional clause. Clearly, the number of possible sums and therefore the number of magic clauses becomes rather large, if we consider VMTLs or TML for dense graphs. In these cases the resulted algorithm has bad performance (see Section 3).
Enumerating All Magic Labelings: The SAT based representation allows us to enumerate all magic labelings using a technique of Jin, Han, Somenzi [3], which is applicable to general SAT instances. The main idea of this technique is to add new clauses to a SAT model with purpose to enumerate all SAT solutions. In our case we start with the presented SAT encoding. If this SAT encoding is satisfiable, we receive a first magic labeling $\lambda : \{1, 2, \ldots, r\} \to \{1, 2, \ldots, r\}$. This means that in the SAT solution exactly the Boolean variables $x_{1,\lambda(1)}, x_{2,\lambda(2)}, \ldots, x_{r,\lambda(r)}$ are set to TRUE. Then we explicitly forbid this magic labeling by adding the new clause $\neg x_{1,\lambda(1)} \vee \neg x_{2,\lambda(2)} \vee \cdots \vee \neg x_{r,\lambda(r)}$ to the current SAT instance. For the updated SAT instance there are two possibilities. If the instance is satisfiable, this leads to another magic labeling, as the first one is not allowed. If not, the first magic labeling was the only one. This process can be iterated, until all or a determined number of magic labelings has been found.

3 Experimental Results

In this section we compare our algorithm (called SAT-MAGIC) with a natural backtracking algorithm (called BACK-MAGIC). In BACK-MAGIC all vertices and edges are labeled in a fixed order, and if a partial labeling makes a magic labeling impossible, then a backtracking step occurs, i.e., a previous labeling of a edge/vertex is changed, and the search continues at this step. All algorithms have been implemented in C++, where we make use of an effective SAT solver implemented by Eén and Sörensson, called MINISAT [2]. The experiments were carried out on a PC with an Athlon 1900MP CPU with 2GB of memory. We test random graphs with size $n = 10, 15$, where $p = 10\%, 20\%, 30\%, 40\%$ edges are

Size n	10								15							
Type	EMTL				VMTL				EMTL				VMTL			
Density p (%)	10	20	30	40	10	20	30	40	10	20	30	40	10	20	30	40
Suc. Back (%)	40	20	10	0	100	0	0	0	0	0	0	0	0	0	0	0
Suc. Sat (%)	100	100	100	80	100	100	0	0	80	60	50	0	100	0	0	0

Table 1
Comparison of Sat Magic and Back Magic

chosen randomly and uniformly distributed from all possible $n \cdot (n-1)/2$ ones. As only 3 connected TMLs are known [1], we do not consider TMLs, but only EMTLs and VMTLs. Note that for each single instance we can easily compute a lower bound $lb \in \mathbb{N}$ and an upper bound $ub \in \mathbb{N}$ for possible magic constants. Then we choose $\min\{10, ub - lb + 1\}$ values of this interval $[lb, ub]$ and for each value we receive a single instance of the form (1) or (2). Thus we have $16 = 2 \cdot 2 \cdot 4$ test classes, where each test class consists of up to 10 single instances.
In Table 3 for both algorithms and for each test class a percentage value is given describing how many instances of this test class can be solved in 600 seconds. The results clearly demonstrate the superiority of Sat-Magic in comparison to Back-Magic. As expected, Sat-Magic behaves rather bad for VMTLs with large density.

References

[1] A. Baker, J. Sawada: *Magic Labelings on Cycles and Wheels.* In B. Yang, D.-Z. Du, C.A. Wang (Eds.): Proc. 2nd Annual International Conference on Combinatorial Optimization and Applications (COCOA). Lecture Notes in Comput. Sci. **5165**, 361-373, 200

[2] N. Eén, N. Sörensson: *An Extensible SAT-Solver.* In E. Giunchiglia, A. Tacchella (Eds.): Proc. 6th International Conference on Theory and Applications of Satisfiability Testing (SAT). Lecture Notes in Comput. Sci. **2919**, 502-518, 2003.

[3] H. Jin, H. Han, F. Somenzi: *Efficient Conflict Analysis for Finding All Satisfying Assignments of a Boolean Circuit.* In N. Halbwachs, L.D. Zuck (Eds.): Proc. 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Comput. Sci. **3440**, 287-300, 2005.

[4] W.D. Wallis: *Magic Graphs.* Birkhäuser, Boston, 2001.

[5] International SAT Competition: "http://www.satcompetition.org/".

New Fully Polynomial Time Approximation Scheme for the makespan minimization with positive tails on a single machine with a fixed non-availability interval

Imed Kacem

LITA, Universit Paul Verlaine–Metz, Ile du Saulcy. Metz 57000 cedex, France.
(Email address: kacem@univ-metz.fr).

Key words:
scheduling, non-availability constraint, approximation, makespan

1 Introduction

The studied problem $(\mathcal{P})$ can be formulated as follows. We have to schedule a set J of n jobs on a single machine, where every job j has a processing time p_j and a tail q_j. The machine can process at most one job at a time and it is unavailable between T_1 and T_2 (i.e., $[T_1, T_2)$ is a forbidden interval). Preemption of jobs is not allowed (jobs have to be performed under the non-resumable scenario). All jobs are ready to be performed at time 0. With no loss of generality, we consider that all data are integers and that jobs are indexed according to Jackson's rule [1] (i.e., jobs are indexed in nonincreasing order of tails). Therefore, we assume that $q_1 \geq q_2 \geq ... \geq q_n$. The consideration of tails is motivated by the large set of scheduling problems such that jobs have delivery times after their processing. As an example, it is well-known that the minimization of makespan with tails is equivalent to the minimization of the maximum lateness with due dates [2]. Let $C_j(S)$ denote the completion time of job j in a feasible schedule S for the problem and let $\varphi_S(\mathcal{P})$ be the makespan yielded by schedule S for instance $\mathcal{I}$ of $(\mathcal{P})$:

$$\varphi_S(\mathcal{I}) = \max_{1 \leq j \leq n} \left(C_j(S) + q_j\right) \tag{1}$$

The aim is to find a feasible schedule S by minimizing the makespan. Due to the dominance of Jackson's order, an optimal schedule is composed of two sequences of jobs scheduled in nondecreasing order of their indexes.

If all the jobs can be inserted before T_1, the instance studied ($\mathcal{I}$) has obviously a trivial optimal solution obtained by Jackson's rule. We therefore consider only the problems in which all the jobs cannot be scheduled before T_1.

In the remainder of this paper $\varphi^*(\mathcal{I})$ denotes the minimal makespan for instance $\mathcal{I}$.

This type of problems has been studied in the literature under various criteria (a sample of these works includes Lee [7], Kacem [4], Kubzin and Strusevich [6], Qi *et al.* [8]-[9], Schmidt [10], He *et al.* [3]). However, few papers studied the problem we consider in this paper. Lee [7] explored the Jackson's sequence JS and proved that its deviation to the optimal makespan cannot exceed $\max_{1\leq j\leq n}(p_j)$, which is equivalent to state that $\varphi_{JS}(\mathcal{I}) \leq 2\varphi^*(\mathcal{I})$. Recently, Yuan et al. developed an interesting PTAS for the studied problem [11]**.** That is why this paper is a good attempt to design more efficient approximation heuristics and approximation schemes to solve the studied problem.

2 New FPTAS

Now, let describe our FPTAS. It uses a simplification technique based on merging small jobs [5]. First, we simplify the instance $\mathcal{I}$ as follows. Given an arbitrary $\varepsilon > 0$, we split the interval $[0, \max_{j\in J}\{q_j\}]$ in $1/\varepsilon$ equal lenght intervals and we round up every tail q_j to the next multiple of $\varepsilon\overline{q}$ ($\overline{q} = \max_{j\in J}\{q_j\}$). Then, we obtain a new instance $\mathcal{I}'$ with no $(1+\varepsilon)$-loss. Thus, J can be divided into $1/\varepsilon$ subsets $J(k)$ $(1 \leq k \leq 1/\varepsilon)$ where jobs in $J(k)$ have identical tails of $k\varepsilon\overline{q}$. The second modification consists in reducing the number of small jobs in every subset $J(k)$. Small jobs are those having processing times $< \varepsilon P/2$ where $P = p_1 + p_2 + ... + p_n$. The reduction is done by merging the small jobs in each $J(k)$ so that we obtain new greater jobs having processing times between $\varepsilon P/2$ and εP. At most, for every subset $J(k)$, a single small job remains. We show that this reduction cannot increase the optimal solution value by more than $(1+\varepsilon)$-factor. We re-index jobs according to nondecreasing order of their tails. The new instance we obtain is denoted as $\mathcal{I}''$. Clearly, the number of jobs remaining in the simplified instance $\mathcal{I}''$ is less than $3/\varepsilon$.

Our FPTAS is based on two steps. First, we use the Jackson's sequence JS obtained for the initial instance $\mathcal{I}$. Then, we apply a modified dynamic programming algorithm APS'_ε on instance $\mathcal{I}''$. The main idea of APS'_ε is to remove a special part of the states generated by a dynamic programming algorithm (See Kacem [4]). Therefore, the modified algorithm becomes faster and yields an approximate solution instead of the optimal schedule.

Given an arbitrary $\varepsilon > 0$, we define $\overline{n} = \min\{n, 3/\varepsilon\}$, $\omega_1 = \left\lceil \frac{4\overline{n}}{\varepsilon} \right\rceil$, $\omega_2 = \left\lceil \frac{2\overline{n}^2}{\varepsilon} \right\rceil$,

$\delta_1 = \frac{\varphi_{JS}(\mathcal{I})}{\omega_1}$and $\delta_2 = \frac{T_1}{\omega_2}$.

We split $[0, \varphi_{JS}(\mathcal{I}))$ into ω_1 equal subintervals $I_m^1 = [(m-1)\delta_1, m\delta_1)_{1 \leq m \leq \omega_1}$. We also split $[0, T_1)$ into ω_2 equal subintervals $I_s^2 = [(s-1)\delta_2, s\delta_2)_{1 \leq s \leq \omega_2}$ of length δ_2. Moreover, we define the two singletons $I_{\omega_1+1}^1 = \{\varphi_{JS}(\mathcal{I})\}$ and $I_{\omega_2+1}^2 = \{T_1\}$. Our algorithm APS'_ε generates reduced sets $\mathcal{X}_k^\#$ of states $[t, f]$ where t is the total processing time of jobs assigned before T_1 in the associated partial schedule and f is the makespan of the same partial schedule. It can be described as follows:

Algorithm APS'_ε

(i). Set $\mathcal{X}_0^\# = \{[0, 0]\}$.

(ii). For $k \in \{1, 2, 3, ..., \overline{n}\}$,

For every state $[t, f]$ in $\mathcal{X}_{k-1}^\#$:

1) Put $\left[t, \max\left(f, T_2 + \sum_{i=1}^k p_i - t + q_k\right)\right]$ in $\mathcal{X}_k^\#$

2) Put $[t + p_k, \max(f, t + p_k + q_k)]$ in $\mathcal{X}_k^\#$ if $t + p_k \leq T_1$

Remove $\mathcal{X}_{k-1}^\#$

Let $[t, f]_{m,s}$ be the state in $\mathcal{X}_k^\#$ such that $f \in I_m^1$ and $t \in I_s^2$ with the smallest possible t (ties are broken by choosing the state of the smallest f).

Set $\mathcal{X}_k^\# = \left\{[t, f]_{m,s} \,|1 \leq m \leq \omega_1 + 1, 1 \leq s \leq \omega_2 + 1\right\}$.

(iii). $\varphi_{APS'_\varepsilon}(\mathcal{I}) = \min_{[t,f] \in \mathcal{X}_n^\#}\{f\}$.

Theorem 1 *Given an arbitrary $\varepsilon > 0$, algorithm APS'_ε yields an output $\varphi_{APS'_\varepsilon}(\mathcal{I}'')$ such that:*

$$\varphi_{APS'_\varepsilon}(\mathcal{I}'') - \varphi^*(\mathcal{I}'') \leq \varepsilon\varphi^*(\mathcal{I}''). \tag{2}$$

The proof will be presented at the conference.

Lemma 2 *Given an arbitrary $\varepsilon > 0$, algorithm APS'_ε can be implemented in $O(n \log n + \min\{n, 1/\varepsilon\}^4/\varepsilon^2)$ time.*

The schedule obtained by APS'_ε for instance $\mathcal{I}''$ can be easily converted into a feasible one for instance $\mathcal{I}$. This can be done in $O(n)$ time. ¿From the previous lemma and theorem, the main result is proved and the following corollary holds.

Corollary 3 *Algorithm APS'_ε is an FPTAS and it can be implemented in $O(n \log n + \min\{n, 1/\varepsilon\}^4/\varepsilon^2)$ time.*

3 Conclusion

In this paper, we considered the non-resumable case of the single machine scheduling problem with a fixed non-availability interval. Our aim is to minimize the makespan when every job has a positive tail. We showed that the problem has an FPTAS (Fully Polynomial Time Approximation Scheme). Such an FPTAS is strongly polynomial. The obtained results outperform the previous polynomial approximation algorithms for this problem.

References

[1] Carlier, J., 1982. The one-machine sequencing problem. *European Journal of Operational Research* 11, 42-47.

[2] Dessouky, M.I., Margenthaler, C.R., 1972. The one-machine sequencing problem with early starts and due dates. *AIIE Transactions* 4:3, 214-222.

[3] He, Y., Zhong, W., Gu, H., 2006. Improved algorithms for two single machine scheduling problems. *Theoretical Computer Science* 363, 257-265.

[4] Kacem, I., 2009. Approximation algorithms for the makespan minimization with positive tails on a single machine with a fixed non-availability interval. *Journal of Combinatorial Optimization* 17:2, 117-133.

[5] Kacem, I., Kellerer, H., 2010. No-Wait Scheduling of a Single-Machine to Minimize the Maximum Lateness, *Proceedings of the 24th Annual Conference of the Belgian Operational Society*, Liège, January 28-29.

[6] Kubzin, M.A., Strusevich, V.A., 2006. Planning machine maintenance in two machine shop scheduling. *Operations Research* 54, 789-800.

[7] Lee, C.Y., 1996. Machine scheduling with an availability constraints. *Journal of Global Optimization* 9, 363-384.

[8] Qi, X., 2007. A note on worst-case performance of heuristics for maintenance scheduling problems. *Discrete Applied Mathematics* 155, 416-422.

[9] Qi, X., Chen, T., Tu, F., 1999. Scheduling the maintenance on a single machine. *Journal of the Operational Research Society* 50, 1071-1078.

[10] Schmidt, G., 2000. Scheduling with limited machine availability. *European Journal of Operational Research* 121, 1-15.

[11] Yuan, J.J., Shi, L., and Ou, J.W., 2008. Single machine scheduling with forbidden intervals and job delivery times. *Asia-Pacific Journal of Operational Research* 25:3, 317-325.

On Enumerating All Maximal Bicliques of Bipartite Graphs

Enver Kayaaslan

enver@cs.bilkent.edu.tr

Key words: bipartite graph, biclique, maximal biclique

1 Introduction

Enumerating all maximal bicliques of a bipartite graph has two main contributions to literature: First, we can use all maximal bicliques to find minimum number of bicliques covering a subset of edges or vertices for given bipartite graph. This problem is referred as Minimum Biclique Cover (MBC) Problem and dates back to the study of Orlin [5]. A recent work by Cornaz and Fonlupt deals with MBC problem for general graphs and useful references can be found therein [3]. Secondly, Agarwal et al. show that extracting all maximal bicliques can be used for compression [1].

In this work, we focus on finding all maximal bicliques in general bipartite graphs. In this point of view, this work can be considered as the specified version of study by Alexe et. al. [2] where the authors try to solve maximal biclique generation problem (MBGP) which is the problem of enumerating all maximal bicliques in simple graphs. Here, we present some theoretical results including a sufficient and necessary condition for a maximal biclique in a bipartite graph as well as a practical algorithm for generating all maximal bicliques of a given bipartite graph.

2 Theoritical Results

A bipartite graph $\mathcal{B} = (X, Y, \mathcal{E})$ is a graph, where the vertices can be divided into two disjoint sets X and Y such that every edge $e_{ij} \in \mathcal{E}$ connectes a vertex in X to one in Y. A biclique (S_x, S_y) of a bipartite graph $\mathcal{B}$ is a complete bipartite subgraph of $\mathcal{B}$ induced by vertex set $S_x \cup S_y$. The neighbor set $N_y(x_i)$ of a vertex $x_i \in X$ is defined as the set of vertices y_j such that

there is an edge $e_{ij} \in \mathcal{E}$, i.e., $N_y(x_i) = \{y_j \in Y : (x_i, y_j) \in \mathcal{E}\}$. Similarly, $N_x(y_j) = \{x_i \in X : (x_i, y_j) \in \mathcal{E}\}$. The empty biclique $(\emptyset, \emptyset)$ is simply denoted as $\emptyset$.

Definition 1 (Maximal Biclique) *Given a bipartite graph $\mathcal{B} = (X, Y, \mathcal{E})$, a biclique (S_x, S_y) is a maximal biclique of $\mathcal{B}$ if no proper superset of (S_x, S_y) is a biclique, i.e., there exists no biclique $(S'_x, S'_y) \neq (S_x, S_y)$ such that $S_x \subseteq S'_x$ and $S_y \subseteq S'_y$.*

For the sake of simplicity, pair of sets $(S_x, \{y_j\})$ and $(\{x_i\}, S_y)$ are denoted as (S_x, y_j) and (x_i, S_y), respectively.

Definition 2 (Consensus Set) *Given a bipartite graph $\mathcal{B} = (X, Y, \mathcal{E})$, for a subset $S_x \subseteq X$ ($S_y \subseteq Y$) the consensus set $P_y(S_x)$ ($P_x(S_y)$) is defined as the intersection of neighbor set of each vertex $x_i \in S_x$ ($y_j \in S_y$),i.e,*

$$P_y(S_x) = \bigcap_{x_i \in S_x} N_y(x_i) \tag{1}$$

By definition, $P_x(\emptyset) = P_y(\emptyset) = \emptyset$.

Note that this is equaivalent to $P_y(S_x) = \{y_j \in Y : (S_x, y_j)$ is biclique$\}$ and similarly $P_x(S_y) = \{x_i \in X : (x_i, S_y)$ is biclique $\}$.

Theorem 1 *$(S_x, S_y) \neq \emptyset$ is a maximal biclique $\Leftrightarrow$ $S_y = P_y(S_x)$ and $S_x = P_x(S_y)$.*

Proof: Let $(S_x, S_y) \neq \emptyset$ be a biclique and so $S_x \subseteq P_x(S_y)$ and $S_y \subseteq P_y(S_x)$. By defintion of maximality, S_x, S_y is a maximal biclique if and only if there exists no $y_j \in Y/S_y$ such that (S_x, y_j) is a biclique and similarly there exists no $x_i \in X/S_x$ such that (x_i, S_y) is biclique. This is equivalent to that there exists no $y_j \in Y/S_y$ such that $y_j \in P_y(S_x)$ and there exists no $x_i \in X/S_x$ such that $x_i \in P_x(S_y)$. Equivalently $S_y = P_y(S_x)$ and $S_x = P_x(S_y)$. $\square$.

Lemma 1 *For any $S_y \subseteq Y$, $S_y \subseteq P_y(P_x(S_y))$. Similarly, for any $S_x \subseteq X$, $S_x \subseteq P_x(P_y(S_x))$.*

Proof: Consider a vertex $y_j \in S_y$. Since $(P_x(S_y), S_y)$ is biclique, $(P_x(S_y), y_j)$ is also biclique. Thus, $y_j \in P_y(P_x(S_y))$ which concludes that $S_y \subseteq P_y(P_x(S_y))$. Similar proof can be conducted for $S_x \subseteq P_x(P_y(S_x))$. $\square$

Theorem 2 *For any $S_y \subseteq Y$ such that $P_x(S_y) \neq \emptyset$, $(P_x(S_y), P_y(P_x(S_y)))$ is a maximal biclique. Similarly, for any $S_x \subseteq X$ such that $P_y(S_x) \neq \emptyset$, $(P_x(P_y(S_x)), P_y(S_x))$ is a maximal biclique.*

Proof: Let S_y be a subset of Y such that $P_x(S_y)$ and let $\tilde{S}_y$ denote $P_y(P_x(S_y))$. Consider a vertex $x_i \in P_x(\tilde{S}_y)$. Then, $(x_i, \tilde{S}_y)$ is biclique. Since $S_y \subseteq \tilde{S}_y$,

(x_i, S_y) is also biclique. Thus $x_i \in P_x(S_y)$ which concludes that $P_x(S_y) \supseteq P_x(\tilde{S}_y)$. Since $P_x(S_y) \subseteq P_x(\tilde{S}_y)$, $P_x(S_y) = P_x(\tilde{S}_y)$. Since both $P_y(P_x(S_y)) = \tilde{S}_y$ and $P_x(S_y) = P_x(\tilde{S}_y)$, $(P_x(S_y), \tilde{S}_y)$ is a maximal biclique. Similar proof can be conducted for maximality of biclique $(P_x(P_y(S_x)), P_y(S_x))$. □

3 Algorithm

Theoretical results suggest that it is sufficient to enumerate on the consensus sets, in order to find all maximal bicliques. Therefore we find all consensus X subsets of bipartite graph $\mathcal{B} = (X, Y, \mathcal{E})$. The algorithm requires only the bipartite graph $\mathcal{B}$. Then we initialize a set of consensus X subsets $\mathcal{S}$ with neighbor sets of every vertex $y_j \in Y$. Concurrently, we hold a priority queue Q which works in a FIFO manner, and it is also initialized by the same set of consensus X subsets. We iteratively grow the set $\mathcal{S}$ as follows. At each iteration, we select an unselected consensus set S_x from queue Q. For each vertex $y_j \in Y$ which is not in the consensus set of S_x, we construct a set S_{new} by the intersection of S_x and neighbor set $N(y_j)$. S_{new} corresponds to the consensus set of $P_y(S_x) \cup \{y_j\}$. We do not consider a vertex in $P_y(S_x)$, because for such a vertex, the intersection will result again S_x which wouldn't be a new consensus set in $\mathcal{S}$. If S_{new} is a new consensus set in $\mathcal{S}$, we insert S_{new} to $\mathcal{S}$. We also enqueue it to priority queue Q in order to expand new consensus sets based on S_{new}. The iterations terminate whenever there remains no consensus set to generate new ones. By the termination, we compute the maximal bicliques by taking pairs $(S_x, P_y(S_x))$ for each subset $S_x \in \mathcal{S}$.

Algorithm 1 FIND-ALL-MAXIMAL Algorithm

Require: Bipartite graph $\mathcal{B} = (X, Y, \mathcal{E})$
  $\mathcal{S} \leftarrow \{N_x(y_j) : y_j \in Y\}$
  $Q \leftarrow \mathcal{S}$
  while $Q \neq \emptyset$ **do**
    $S_x \leftarrow$ DEQUEUE(Q)
    for each $y_j \in Y / P_y(S_x)$ **do**
      $S_{new} \leftarrow S_x \cap N(y_j)$
      if $S_{new} \notin \mathcal{S}$ **then**
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{S_{new}\}$
        ENQUEUE(Q, S_{new})
      end if
    end for
  end while
  $\mathcal{C}_{max} = \{(S_x, P_y(S_x)) : S_x \in \mathcal{S}\}$
  return $\mathcal{C}_{max}$

For each maximal biclique we check for at most $|Y|$ new bicliques. With a naive

implementation, the checking procedure can be done in polynomial time on number of total maximal bicliques and number of vertices. Thus, the whole procedure runs in polynomial-time in total of input and output size with a naive implementation which conludes that FIND-ALL-MAXIMAL is a *total polynomial* algorithm [4] for the problem of enumerating all maximal bicliques of a bipartite graph.

4 Conclusion

In this work, we study the problem of enumerating all maximal bicliques in a given bipartite graphs. We devised a necessary and sufficient condition for a maximal biclique in bipartite graph which happened to be useful for enumeration. As well as, we constructed an algorithm for generating all maximal bicliques of a bipartite graph which runs in *total polynomial* time.

References

[1] P. K. Agarwal, N. Alon, B. Aranov, and S. Suri, *Can visibility graphs be represented compactly?*, Discrete and Computational Geometry, 12 (1994), pp. 347 – 365.

[2] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. L. Hammer, and B. Simeone, *Consensus algorithms for the generation of all maximal bicliques*, Discrete Applied Mathematics, 145 (2004), pp. 11 – 21. Graph Optimization IV.

[3] D. Cornaz and J. Fonlupt, *Chromatic characterization of biclique covers*, Discrete Mathematics, 306 (2006), pp. 495 – 507.

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

[5] J. Orlin, *Contentment in graph theory: Covering graphs with cliques*, Proceedings of the Koninklijke Nederlandse, (1977), pp. 406–424.

A tight analysis of Brown-Baker-Katseff sequences for online strip packing

W. Kern and J.J. Paulus

1 Abstract

In the two-dimensional strip packing problem a number of rectangles have to be packed without rotation or overlap into a strip such that the height of the strip used is minimum. The width of the rectangles is bounded by 1 and the strip has width 1 and infinite height. Baker, Coffman and Rivest [1] show that this problem is NP-hard.

We study the online version of this packing problem. In the online version the rectangles are given to the online algorithm one by one from a list, and the next rectangle is given as soon as the current rectangle is irrevocably placed into the strip. To evaluate the performance of an online algorithm we employ competitive analysis. For a list of rectangles L, the height of a strip used by online algorithm A and by the optimal solution is denoted by $A(L)$ and $OPT(L)$, respectively. The optimal solution is not restricted in any way by the ordering of the rectangles in the list. Competitive analysis measures the absolute worst-case performance of online algorithm A by its competitive ratio $\sup_L\{A(L)/OPT(L)\}$.

Regarding the upper bound on the competitive ratio for online strip packing, recent advances have been made by Ye, Han and Zhang [6] and Hurink and Paulus [4]. Independently they present an online algorithm with competitive ratio $7/2+\sqrt{10} \approx 6.6623$, that is a modification of the well known shelf algorithm. We refer to these two papers for a more extensive overview of the literature.

In the early 80's, Brown, Baker and Katseff [2] derived a lower bound $\rho \geq 2$ on the competitive ratio of any online algorithm by constructing certain (adversary) sequences in a fairly straightforward way. These sequences were further studied by Johannes [5] and Hurink and Paulus [3], who derived improved lower bounds of 2.25 and 2.43, *resp.* (Both results are computer aided and presented in terms of online parallel machine scheduling, a closely related problem.) The paper of Hurink and Paulus [3] also presents an upper bound of $\rho \leq 2.5$ for packing such "Brown-Baker-Katseff sequences" . The purpose of our present paper is to propose a potential function approach that allows us to close the gap between 2.43 and 2.5. We present a tight analysis, showing that Brown-Baker-Katseff sequences

can be packed online with competitive ratio $\rho = 3/2 + \sqrt{33}/6$ and that this is best possible. As a byproduct we obtain a new lower bound $\rho \approx 2.457$ for online strip packing.

Acknowledgment

Part of this research has been funded by the Dutch BSIK/BRICKS project.

References

[1] Baker B.S., Coffman E.G. and Rivest R.L. (1980). *Orthogonal packings in two-dimensions.* SIAM Journal on Computing 9:846-855.

[2] Brown D.J., Baker B.S. and Katseff H.P. (1982). *Lower bounds for on-line two-dimensional packing algorithms.* Acta Informatica 18:207-225.

[3] Hurink J.L. and Paulus J.J. (2008). *Online scheduling of parallel jobs on two machines is 2-competitive.* Operations Research Letters 36:51-56.

[4] Hurink J.L. and Paulus J.J. (2008). *Online algorithm for parallel job scheduling and strip packing.* Lecture Notes in Computer Science (WAOA 2007) 4927:67-74.

[5] Johannes B. (2006) *Scheduling parallel jobs to minimize the makespan.* Journal of Scheduling 9:433-452.

[6] Ye D., Han X. and Zhang G. (2009) *A note on online strip packing.* Journal of Combinatorial Optimization, in press, doi:10.1007/s10878-007-9125-x.

On a Stochastic Knapsack Problem

Stefanie Kosuch and Marc Letournel and Abdel Lisser

Laboratoire de recherche en Informatique, Université Paris Sud
91405 Orsay Cedex, France

Key words: stochastic knapsack, expectation constraint, stochastic gradient method, Arrow-Hurwicz

1 Introduction

The deterministic knapsack problem is a well known and well studied NP-hard combinatorial optimization problem. It consists in filling a knapsack with items out of a given set such that the weight capacity of the knapsack is respected and the total reward maximized. For a review of references on the stochastic knapsack problem, stochastic gradient algorithms and branch-and-bound methods see [4]. In the deterministic problem, all parameters (item weights, rewards, knapsack capacity) are known (*deterministic*). In the stochastic counterpart, some (or all) of these parameters are assumed to be random, i.e. not known at the moment the decision has to be made.
In this paper, we study the stochastic knapsack problem with expectation constraint. The item weights are assumed to be independently normally distributed. We solve the relaxed version of this problem using a stochastic gradient algorithm in order to provide upper bounds for a branch-and-bound framework. Two approaches to estimate the needed gradients are applied, one based on Integration by Parts and one using Finite Differences. Finite Differences is a robust and simple approach with efficient results despite the fact that the estimated gradients are biased, meanwhile Integration by Parts is based upon a more theoretical analysis and permits to enlarge the field of applications.

2 Mathematical formulations

We consider a stochastic knapsack problem of the following form: Given a set of n items. Each item has a weight that is not known in advance and the

decision of which items to choose has to be made without the exact knowledge of their weights. Therefore, we handle the weights as random variables and assume that weight χ_i of item i is independently normally distributed with mean $\mu_i > 0$ and standard deviation σ_i. Furthermore, each item has a fix reward per weight unit $r_i > 0$. We denote by χ, μ, σ and r the corresponding n-dimensional vectors. The aim is to maximize the expected total gain $\mathbb{E}[\sum_{i=1}^{n} r_i \chi_i x_i]$. In addition, we assume that the knapsack problem has a fixed weight capacity $c > 0$. In this paper, we solve the following expectation constrained knapsack problem:

Expectation Constrained Knapsack Problem ($ECKP$)

$$\max_{x \in \{0,1\}^n} \mathbb{E}\left[\sum_{i=1}^{n} r_i \chi_i x_i\right] \tag{1}$$

$$\text{s.t.} \quad \mathbb{E}\left[\mathbb{H}_{\mathbb{R}^+}(c - g(x, \chi))\right] \geq p \tag{2}$$

where $\mathbb{E}[\cdot]$ denotes the expectation, $g(x, \chi) = \sum_{i=1}^{n} \chi_i x_i$ is the total weight of the chosen items, $\mathbb{H}_{\mathbb{R}^+}$ denotes the indicator function of the positive real interval - the Heaviside function, and $p \in (0.5, 1]$ is the prescribed probability. We refer to the function inside the expectation of the constraint function as θ, i.e. $\theta(x, \chi) = \mathbb{H}_{\mathbb{R}^+}(c - g(x, \chi))$.

3 Problem solving method

Due to its combinatorial nature, $ECKP$ can be solved using a branch-and-bound framework as presented in [4]. To obtain upper bounds, the authors propose to solve the corresponding continuous optimization problem using a stochastic gradient type algorithm. A stochastic gradient algorithm is an algorithm that combines both Monte-Carlo techniques and the deterministic gradient method. More precisely, instead of computing the gradient of the objective funtion (that is a function in expectation) to determine the direction of descent, one uses the gradient of the function insight the expectation. By drawing independent samples of the random variables at each iteration, one approximates the expectation.
Applying a gradient method to solve the relaxed $ECKP$ is promising as its objective function is concave and, in addition, constraint (2) defines a convex feasible set due to the assumption that the weights are independently normally distributed.
The particular stochastic gradient algorithm used in this work is the Stochastic Arrow-Hurwicz algorithm (hereafter called SAH-algorithm) that uses Lagrangian multipliers to deal with the expectation constraint (for further details see [3]).

However, to use such an algorithm for $ECKP$, one has to estimate the gradient of the indicator function $\mathbb{H}_{\mathbb{R}^+}(\cdot)$. In this paper, we apply two different approaches: the first one is a non-biased estimator based on Integration by Parts (called hereafter *IP-method*) proposed in [1] to solve continuous stochastic optimization problems. The second approach is a Finite Differences estimator (*FD-method*) presented in [2]. Unlike the IP-method method, the FD-method provides a biased estimator of the gradient.
In subsection 3.0.1 we present the two methods. Subsection 3.0.2 gives a first insight in the convergence analysis we conducted.

3.0.1 Gradient computation methods

In the FD-method, the h-th component of the gradient of θ is approximated by the corresponding difference quotient

$$\frac{\theta(x+\delta\nu^h,\chi)-\theta(x-\delta\nu^h,\chi)}{2\delta}$$

where $\delta > 0$ and $\nu^h \in \{0,1\}^n$ such that $\nu^h_h = 1$ and $\nu^h_i = 0$ for $i \neq h$.
The basic idea of the IP-method consists in using Integration by Parts to reformulate $\mathbb{E}[\theta(x,\chi)]$ which gives rise to a function in expectation $\mathbb{E}[\widetilde{\theta}(x,\chi)]$ s.t. $\mathbb{E}[\widetilde{\theta}(x,\chi)] = \mathbb{E}[\theta(x,\chi)]$. $\widetilde{\theta}$ is differentiable and the idea is to use the gradient of $\widetilde{\theta}$ in the SAH-algorithm. Andrieu et al. presented how to compute such a $\widetilde{\theta}(x,\chi)$ using Integration by Parts (see Theorem 5.5 in [1]). We state and proof their theorem for the case of $ECKP$ with normally distributed weights.

3.0.2 Convergence analysis

When using the IP-method, main adaptations have been made to correctly check all hypotheses of convergence. Instead of replacing $\{0,1\}^n$ by $[0,1]^n$ when relaxing $ECKP$, the theoretical analysis compels us to consider a complementary set of a neighborhood of $0_{[0,1]^n}$. However, assuming that an empty knapsack is not an optimal solution, it is convenient to consider that the optimal solution vector of the continuous problem contains at least one component x_κ with $x_\kappa \geq 1/n$. We are thus allowed to replace $[0,1]^n$ by $\{x \in [0,1]^n \,|\, ||x||_\infty \geq 1/n\} = X_{cont}$. Accordingly, we obtain the following admissible set of the relaxed $ECKP$:

$$X^{ad}_{cont} = \{x \in X_{cont} : \mathbb{E}\left[\mathbb{H}_{\mathbb{R}^+}(c - g(x,\chi))\right] \geq p\}$$

Checking that all steps of the algorithm stay in this subset is a central point of our work.

4 Numerical results for the relaxed and combinatorial ECKP

We tested our algorithms on an instance from the literature as well as on a great number of randomly generated instances.
Numerical tests of the SAH-algorithm involving the abovementioned adaptations have shown that the algorithm converges on all tested instances. We also compared our approach with a method that has previously been used to solve the relaxed $ECKP$. The idea of this method is to reformulate the problem as a deterministic equivalent second order cone problem ($SOCP$) and to solve it using an interior point algorithm. It turned out that in terms of running time, our SAH-algorithm outperforms the $SOCP$ approach for small and medium size instances (up to 1000 items). Concerning the resolution of the combinatorial problem using a branch-and-bound framework, we are able to solve problems with up to 250 items in an average computing time of $1h$. In comparison, when using the $SOCP$ procedure one can only solve problems up to 75 items in comparable time.

References

[1] L. Andrieu. Optimization sous contrainte en probabilité. Ecole Nationale des Ponts et Chaussés, 2004.

[2] Laetitia Andrieu, Guy Cohen, and Felisa Vzquez-Abad. Stochastic programming with probability constraints. http://fr.arxiv.org/abs/0708.0281 (Accessed 24 October 2008), 2007.

[3] J. C. Culioli and G. Cohen. Optimisation stochastique sous contraintes en espérance. *Comptes rendus de l'Académie des sciences, Paris, Série I*, 320(6):753 758, 2008.

[4] Stefanie Kosuch and Abdel Lisser. Upper bounds for the 0-1 stochastic knapsack problem and a b&b algorithm. *Annals of Operations Research (Online First)*, 2009. http://dx.doi.org/10.1007/s10479-009-0577-5.

Determining Optimal Stationary Strategies for Discounted Stochastic Optimal Control Problem on Networks

Dmitrii Lozovanu [a,*], Stefan Pikl [b]

[a] *Institute of Mathematics and Computer Science, Academy of Sciences, Academy str., 5, Chisinau, MD–2028, Moldova*

[b] *Institut für Theoretische Informatik, Mathematik und Operations Research, Fakultät fur Informatik, Universität der Bundeswehr, München*

Abstract

The stochastic version of discrete optimal control problem with infinite time horizon and discounted integral-time cost criterion is considered. This problem is formulated and studied on certain networks. A polynomial time algorithm for determining the optimal stationary strategies for the considered problems is proposed and some applications of the algorithm for related Markov decision problems are described.

Key words: Discounted Stochastic Control Problem, Optimal Stationary Strategies, Polynomial Time Algorithm, Discounted Markov Processes

1 Introduction, Problem Formulation and the Main Concept

In this paper we consider the stochastic version of the following discrete optimal control problem with infinite time horizon and a discounted integral-time cost criterion by trajectory. Let a time-discrete system L with a finite set of states X be given. Assume that the dynamics of the system is described by a directed graph of states transitions $G = (X, E)$ where the set of vertices X corresponds to the set of states of the dynamical system; an arbitrary directed edge $e = (x, y)$ expresses the possibility of the system to pass from the state $x = x(t)$ to the state $y = x(t)$ at every discrete moment of time

* Dmitrii Lozovanu
Email address: lozovanu@usm.md, stefan.pickl@unibw.de (Stefan Pikl).

$t = 0, 1, 2, \ldots$. Hereby, a directed edge $e = (x, y) \in E$ corresponds to a feasible stationary control of system L in the state x and the subset of edges $E^+(x) = \{e = (x, y) \in E | y \in X\}$ corresponds to the set of feasible stationary controls of the system in the state $x \in X$. We assume that on the edge set E a cost function $c : E \to R$ is defined which assigns a cost c_e to each directed edge $e = (x, y) \in E$ when the system makes a transition from the state $x = x(t)$ to the state $y = x(t + 1)$ for every $t = 0, 1, 2, \ldots$,i.e. the costs $c_{x(t),x(t+1)}$ does not depend on t. We define a stationary control of system L in G as a map

$$s : x \to y \in X^+(x) \quad for \quad x \in X,$$

where $X^+(x) = \{y \in X | (x, y) \in E\}$. Let s be an arbitrary stationary control. Then the set of edges of the form $(x, s(x))$ in G generates a subgraph $G_s = (X, E_s)$ where each vertex $x \in X$ contains one leaving directed edge. So, if the starting state $x_0 = x(0)$ is fixed then the system makes transitions from one state to another through the corresponding directed edges $e_0^s, e_1^s, e_2^s, \ldots, e_t^s, \ldots$, where $e_t^s = (x(t), x(t+1)), \quad t = 0, 1, 2, \ldots$. This sequence of directed edges generates a trajectory $x_0 = x(0), x(1), x(2), \ldots$ which leads to a unique directed cycle. For an arbitrary stationary strategy s and a fixed starting state x_0 the discounted integral-time cost $\sigma_{x_0}^{\lambda}(s)$ is defined as follows $\sigma_{x_0}^{\lambda}(s) = \sum_{t=0}^{\infty} \lambda^t c_{e_t^s}$, where $\lambda, \quad 0 \leq \lambda < 1$, is a given (so called) discounted factor. Based on the results from [1,3] it is easy to show that for an arbitrary stationary strategy s there exists $\sigma_{x_0}^{\lambda}(s)$. If we denote by $\sigma^{\lambda}(s)$ the vector column with components $\sigma_x^{\lambda}(s)$ for $x \in X$ then $\sigma_{x_0}^{\lambda}(s)$ can be found by solving the system of linear equations $(I - \lambda P^s)\sigma^{\lambda}(s) = \overline{c}^s$, where $\overline{c}^s$ is the vector with corresponding components $c_{(x,s(x))}$ for $x \in X, \quad I$ is the identity matrix and P^s the matrix with elements $p_{x,y}^s$ for $x, y \in X$ defined as follows

$$p_{x,y}^s = \begin{cases} 1, \text{ if } y = s(x); \\ 0, \text{ if } y \neq s(x). \end{cases}$$

We are seeking for a stationary control s^* such that $\sigma_{x_0}^{\lambda}(s^*) = \min_s \sigma_{x_0}^{\lambda}(s)$. In this paper we consider the stochastic version of the problem formulated above. We assume that the dynamical system may admit states in which the vector of control parameters is changed in a random way. So, the set of states X is divided into two subsets $X = X_1 \cup X_2, \quad X_1 \cap X_2 = \emptyset$, where X_1 represents the set of states in which the decision maker is able to control the dynamical system and X_2 represents the set of states in which the dynamical system makes transition to the next state in a random way. So, for every $x \in X$ on the set of feasible transitions $E^+(x)$ the distribution function $p : E^+(x) \to R$ is defined such that $\sum_{e \in E^+(x)} p_e = 1, \quad p_e \geq 0, \forall e \in E^+(x)$ and the transitions from the states $x \in X_2$ to the the next states are made according to these distribution functions. Here in a similar way as in the previous case of the problem we assume that to each directed edge $e = (x, y) \in E$ a cost c_e is associated when the system makes a transition from the state $x = x(t)$ to

the state $y = x(t+1)$ for every $t = 0, 1, 2, \ldots$. In addition we assume that the discounted factor $\lambda, \quad 0 \leq \lambda < 1$, and the starting state x_0 are given. We define a stationary control for the considered problem as a map

$$s : x \to y \in X^+(x) \quad for \quad x \in X_1.$$

For an arbitrary stationary strategy s we define the graph $G_s = (X, E_s \cup E_{X_2})$, where $E_s = \{e = (x, y) \in E | x \in X_1, y = s(x)\}, \quad E_{X_2} = \{e = (x, y) | x \in X_2, y \in X\}$. This graph corresponds to a Markov process with the probability matrix $P^s = (p^s_{x,y})$, where

$$p^s_{x,y} = \begin{cases} p_{x,y}, \text{ if } x \in X_2 \text{ and } y = X; \\ 1, \quad \text{if } x \in X_1 \text{ and } y = s(x); \\ 0, \quad \text{if } x \in X_1 \text{ and } y \neq s(x). \end{cases}$$

For this Markov process with associated costs $c_e, \quad e \in E$ we can define the expected discounted integral-time cost $\sigma^\lambda_{x_0}(s)$ in the same way as for discounted Markov processes with rewards (if we treat the rewards as the costs). In this paper we consider the problem of determining the strategy s^* for which $\sigma^\lambda_{x_0}(s^*) = \min_s \sigma^\lambda_{x_0}(s)$.

2 The Main Results

The stationary case of the considered discounted stochastic control problem can be studied and solved using the general concept of Markov decision processes and the linear programming approach to corresponding problems (see [1–3]). Here we develop a new technique and we will formulate a new linear programming problem which is more suitable to the specific context. To obtain our linear model we shall use the following condition:

$$\begin{cases} \sigma_x - \lambda \sum_{y \in X^+(x)} p^s_{x,y} \sigma_y = \sum_{y \in X^+(x)} c_{(x,y)} p^s_{x,y}, \quad \forall x \in X_1; \\ \sigma_x - \lambda \sum_{y \in X^+(x)} p_{x,y} \sigma_y = \sum_{y \in X+(x)} c_{(x,y)} p_{x,y}, \quad \forall x \in X_2, \end{cases} \tag{1}$$

for an arbitrary stationary strategy s. For fixed s the probabilities $p^s_{x,y}, \quad x \in X, y \in X^+(x)$, satisfy the conditions: $\sum_{y \in X^+(x)} p^s_{x,y} = 1, \forall x \in X_1; \quad p_{x,y} \in \{0, 1\}, \forall x \in X_1, y \in X^+(x)$. The system (1) has a unique solution with respect to σ_x for $x \in X$ and therefore we uniquely determine $\sigma^s_{x_0}$. Thus we can consider the linear programming problem: Maximize

$$\psi_{p^s}(\sigma) = \sigma_{x_0} \tag{2}$$

subject to (1). This problem has a unique feasible solution which is the optimal one. The dual program for this problem is: Minimize

$$\varphi_{p^s}(\alpha) = \sum_{x\in X_1}\sum_{y\in X(x)} c_{(x,y)}p^s_{x,y}\alpha_x + \sum_{x\in X_2}\sum_{y\in X(x)} c_{(x,y)}p_{x,y}\alpha_x \tag{3}$$

subject to

$$\begin{cases} \alpha_y - \lambda\sum_{x\in X_1^-(y)} p^s_{x,y}\alpha_x - \lambda\sum_{x\in X_2^-(y)} p_{x,y}\alpha_x \geq 1, & y = x_0; \\ \alpha_y - \lambda\sum_{x\in X_1^-(y)} p^s_{x,y}\alpha_x - \lambda\sum_{x\in X_2^-(y)} p_{x,y}\alpha_x \geq 0, & \forall\ y \in X\setminus\{x_0\}. \end{cases} \tag{4}$$

If we take here the minimum with respect to s then we obtain a bilinear programming problem with respect to α_x and $p^s_{x,y}$, where $p^s_{x,y}$ satisfy the conditions: $\sum_{y\in X^+(x)} p^s_{x,y} = 1; p^s_{x,y} \in \{0,1\}, \forall x \in X_1, y \in X^+(x)$. We have proved that the optimal solution is preserved if these conditions are changed by conditions: $\sum_{y\in X^+(x)} p^s_{x,y}\alpha_x = \alpha_x, \forall x \in X_1, \forall y \in X^+(x); \alpha_x \geq 0, \beta_{x,y} \geq 0, \forall x \in X, y \in X^+(y)$. If we substitute after that operation $\beta_{x,y} = p^s_{x,y}\alpha_x$ then our bilinear programming problem obtained on the bases of (3),(4) with mentioned above conditions is reduced to the linear programming problem: Minimize

$$\varphi(\alpha,\beta) = \sum_{x\in X_1}\sum_{y\in X(x)} c_{(x,y)}\beta_{x,y} + \sum_{x\in X_2}\sum_{y\in X(x)} c_{(x,y)}p_{x,y}\alpha_x \tag{5}$$

subject to

$$\begin{cases} \alpha_y - \lambda\sum_{x\in X_1^-(y)} \beta_{x,y} - \lambda\sum_{x\in X_2^-(y)} p_{x,y}\alpha_x \geq 1, & y = x_o; \\ \alpha_y - \lambda\sum_{x\in X_1^-(y)} \beta_{x,y} - \lambda\sum_{x\in X_2^-(y)} p_{x,y}\alpha_x \geq 0, & \forall\ y \in X\setminus\{x_0\}; \\ \sum_{y\in X^-(x)} \beta_{x,y} = \alpha_x, & \forall x \in X_1;\ \beta_{x,y} \geq 0, \alpha_x \geq 0, \forall x \in X, y \in X^+(x), \end{cases} \tag{6}$$

where $X_1^-(y) = \{x \in X_1 | (x,y) \in E\}$, $X_2^-(y) = \{x \in X_2 | (x,y) \in E\}$. The following result holds: If $\alpha^*_x, \beta^*_{x,y}$ is a basic optimal solution of the problem (5),(6) and $\alpha^*_x \neq 0$ for $x \in X_1$ then $p^{s^*}_{x,y} = \beta^*_{x,y}/\alpha^*_x \in \{0,1\}, x \in X_1, y \in X^+(y)$; the optimal stationary strategy $s^* : x \to y$ for $y \in X^+(x)$ corresponds to $p^{s^*}_{x,y} = 1$ for $x \in X_1, y \in X^+(x)$.

References

[1] Howard R.A., Dynamic Programming and Markov Processes. Wiley (1960)

[2] Lozovanu D., Pickl. S., Optimization and Multiobjective Control of Time-Discrete Systems. Springer Verlag (2009)

[3] Puterman M., Markov Decision Processes. Wiley (1993)

Approximating Independent Set in Semi-Random Graphs

Bodo Manthey [a] Kai Plociennik [b]

[a] *University of Twente, Department of Applied Mathematics*
P. O. Box 217, 7500 AE Enschede, The Netherlands

[b] *TU Chemnitz, Fakultät für Informatik*
Straße der Nationen 62, 09107 Chemnitz, Germany

Abstract

We present an algorithm for the independent set problem on semi-random graphs, which are generated as follows: An adversary chooses an n-vertex graph, and then each edge is flipped independently with a probability of $\varepsilon > 0$. Our algorithm runs in expected polynomial time and guarantees an approximation ratio of roughly $O(\sqrt{n\varepsilon})$, which beats the inapproximability bounds.

1 Introduction and Our Results

Given an undirected graph $G = (V, E)$, the goal of the maximum independent set problem (IS) is to find an *independent set* $I \subseteq V$ (i.e., no edge of E connects two vertices of I) of maximum cardinality. The size of the largest such set is G's *independence number* $\alpha(G)$. IS is NP-hard and even hard to approximate with a ratio of $O(n^{1-\delta})$ for every $\delta > 0$, where n is the number of vertices [5]. The best worst-case polynomial-time approximation algorithm for IS achieves an approximation ratio of $O(n\frac{(\log\log n)^2}{(\log n)^3})$ [2]. Often, however, approximation algorithms show a better performance than their worst-case guarantees promise. To explain the gap between worst-case and observed approximability, the average-case approximability of IS with respect to random graphs in the $G(n, p)$ model has been analyzed by Krivelevich and Vu [4]. They achieve an approximation ratio of $O(\sqrt{np}/\log n)$ in expected polynomial time. A drawback of such an average-case analysis is that it often says little about

Email addresses: b.manthey@utwente.nl (Bodo Manthey), kai.plociennik@informatik.tu-chemnitz.de (Kai Plociennik).

`GreedyIS`(G)
1: Set $C_1 := \{1\}$ and $\chi := 1$.
2: For $v = 2, \ldots, n$: If there is an i such that $C_i \cup \{v\}$ is an independent set of G, then $C_i := C_i \cup \{v\}$ for the smallest such i. Otherwise, create a new class by setting $\chi := \chi + 1$ and $C_\chi := \{v\}$.
3: Choose i that maximizes $|C_i|$. Output $I = C_i$.

Algorithm 1: A simple greedy algorithm for independent set.

typical performance: random instances have very special properties with high probability and often do not reflect real instances. To circumvent this, graph problems have been analyzed with semi-random inputs [1,3].

In this paper, we analyze the approximability of IS for a semi-random input model: An adversary specifies a graph $G = (V, E)$. Then, a random graph $\mathcal{G} = (V, \mathcal{E})$ is produced by flipping each potential edge in G independently with a probability of $\varepsilon > 0$. More precisely, for every $e \in E$, we have $e \in \mathcal{E}$ with a probability of $p_e = 1 - \varepsilon$, while for every $e \notin E$, we have $e \in \mathcal{E}$ with a probability of $p_e = \varepsilon$. We call this distribution $\mathcal{G}(G, \varepsilon)$. In the extreme case $\varepsilon = 0$, the adversary has full power and we have $\mathcal{G} = G$. For larger values of ε, the adversary loses power. We adapt the algorithm of Krivelevich and Vu [4] to our semi-random model and show that it guarantees an approximation ratio of roughly $O(\sqrt{n\varepsilon})$ in expected polynomial time.

2 Approximating Independent Set

For simplicity, we always assume $V = \{1, \ldots, n\}$ from now on. `GreedyIS` (Algorithm 1) is a simple greedy algorithm, which outputs an independent set. `GreedyIS` is later used as a subroutine of `ApproxIS`.

Lemma 1 *Let $G = (V, E)$ be a graph, and let ε be arbitrary with $n^{-1/2} \leq \varepsilon \leq 1/2$. Let $\mathrm{gis}(G, \varepsilon) = \frac{1}{32} \cdot \min\{\frac{\ln n}{\varepsilon}, \frac{n^2 \ln n}{|E| \ln(1/\varepsilon)}\}$. Let I be the independent set of $\mathcal{G}$ computed by `GreedyIS`, where $\mathcal{G}$ is drawn from $\mathcal{G}(G, \varepsilon)$. Then $\Pr[|I| < \mathrm{gis}(G, \varepsilon)] \leq e^{-n \ln n}$.*

We also need an upper bound on the independence number of a graph drawn from $\mathcal{G}(G, \varepsilon)$. The proof is an adaption of Krivelevich and Vu's technique [4] to our semi-random model. For a graph G and a random graph $\mathcal{G} = (V, \mathcal{E})$ drawn from $\mathcal{G}(G, \varepsilon)$, let $A = A(\mathcal{G}, G, \varepsilon) = (a_{ij})_{1 \leq i,j \leq n}$ be the $n \times n$-matrix given by $a_{ij} = 1$ if $e = \{i, j\} \notin \mathcal{E}$ and $a_{ij} = -(1 - p_e)/p_e$ if $e = \{i, j\} \in \mathcal{E}$, where $p_e = \varepsilon$ if $e \notin E$ and $p_e = 1 - \varepsilon$ if $e \in E$. Note that A depends on G since G defines the values p_e. ¿From the results of Krivelevich and Vu [4, Lemma 2.4], we easily get $\alpha(\mathcal{G}) \leq \lambda_1(A(\mathcal{G}, G, \varepsilon))$ for any G, ε, and $\mathcal{G}$, where λ_1 denotes the largest eigenvalue of a matrix. We have to show that $\lambda_1(A(\mathcal{G}, G, \varepsilon)) =$

`ApproxIS`$(\mathcal{G} = (V, \mathcal{E}), G, \varepsilon)$

1: $I :=$ `GreedyIS`$(\mathcal{G})$. If $|I| < \mathrm{gis}(G, \varepsilon)$, go to Step 5.
2: Compute $\lambda_1(A(\mathcal{G}, G, \varepsilon))$. If $\lambda_1 < 2^8 \cdot (\log n) \cdot \sqrt{n/\varepsilon}$, then output I.
3: Compute $|\overline{N}(S')|$ for all sets $S' \subseteq V$ with $|S'| = (8 \log n)/\varepsilon$. If $|\overline{N}(S')| \leq (2 \log n) \cdot \sqrt{n/\varepsilon}$ for all such subsets S', output I.
4: Check all subsets $S'' \subseteq V$ with $|S''| = (8 \log n) \cdot \sqrt{n/\varepsilon}$. If none of them is independent, output I.
5: Try all subsets of V and output a largest independent set found.

Algorithm 2: Approximation algorithm for independent set with guaranteed approximation ratio and expected polynomial running-time.

$O((\log n) \cdot \sqrt{n/\varepsilon})$ with high probability. Krivelevich and Vu's proof [4] of the corresponding result for $G(n, p)$ graphs is based on a concentration result for the largest eigenvalue of a matrix. To apply their techniques, we first have to determine the expected value $\mathrm{E}[\lambda_1(A)]$ of the largest eigenvalue (Lemma 2). Using this, we can derive a tail bound for $\lambda_1(A)$ (Lemma 3). Together with $\alpha(\mathcal{G}) \leq \lambda_1(A)$, we then get a concentration result for the size of the largest independent set in $\mathcal{G}(G, \varepsilon)$ graphs (Theorem 4).

Lemma 2 *Fix a graph $G = (V, E)$, and let $\varepsilon = \Omega((\log n)^2/n)$, $\varepsilon \leq 1/2$. Let $A = A(\mathcal{G}, G, \varepsilon)$ for $\mathcal{G}$ drawn from $\mathcal{G}(G, \varepsilon)$. Then $\mathrm{E}[\lambda_1(A)] \leq 2^7 (\log n) \sqrt{n/\varepsilon}$.*

Lemma 3 *Under the same assumptions as in Lemma 2, we have $\Pr[\lambda_1(A) \geq 2^8 (\log n) \sqrt{n/\varepsilon}] \leq 4 \exp(-2^9 n\varepsilon (\log n)^2)$.*

Theorem 4 *Fix a graph $G = (V, E)$, and let $\varepsilon = \Omega((\log n)^2/n)$, $\varepsilon \leq 1/2$. Let $\mathcal{G}$ be a random graph drawn from $\mathcal{G}(G, \varepsilon)$. Then $\mathrm{E}[\alpha(\mathcal{G})] \leq 2^7 \cdot (\log n) \cdot \sqrt{n/\varepsilon}$ and $\Pr[\alpha(\mathcal{G}) \geq 2^8 \cdot (\log n) \cdot \sqrt{n/\varepsilon}] \leq 4 \exp(-2^9 \cdot n\varepsilon \cdot (\log n)^2)$.*

Our algorithm `ApproxIS` (Algorithm 2) gets the adversarial graph G, the flip probability ε, and the random graph $\mathcal{G}$ from $\mathcal{G}(G, \varepsilon)$ as input. It runs `GreedyIS` and tries to certify that this yields a good approximation. If this fails, `ApproxIS` turns to brute-force search. Let us briefly discuss the influence of G. With decreasing ε and increasing $|E|$, the graph G gains influence. This is reflected in the approximation ratio below. In Step 3, the following definition is used: For a graph $G = (V, E)$ and $S \subseteq V$, the *non-neighborhood* $\overline{N}(S)$ of S is the set of vertices not adjacent to S.

Theorem 5 *Fix a graph $G = (V, E)$ and a flip probability $n^{-1/2} \leq \varepsilon \leq 1/2$. Let $\mathcal{G}$ be drawn from $\mathcal{G}(G, \varepsilon)$. Then `ApproxIS`$(\mathcal{G}, G, \varepsilon)$ has polynomial expected running time. If ε is sufficiently large, i.e., $\frac{\ln(1/\varepsilon)}{\varepsilon} \leq \frac{n^2}{|E|}$, it guarantees an approximation ratio of $O(\sqrt{n\varepsilon})$. Otherwise, it guarantees an approximation ratio of $O\left(\frac{|E| \log(1/\varepsilon)}{n^{3/2}\sqrt{\varepsilon}}\right)$.*

`GreedyIS` alone is an algorithm with worst-case polynomial running-time, but the approximation ratio then holds only in expectation and with high probability. This complements the performance of `ApproxIS`. The analysis follows from Lemma 1 and Theorem 4.

Corollary 6 *Let $G = (V, E)$ be a graph, and let $n^{-1/2} \leq \varepsilon \leq 1/2$. Then* `GreedyIS` *achieves an expected approximation ratio of $O\Big(\frac{\sqrt{n/\varepsilon}}{\min\{1/\varepsilon, n^2/(|E|\ln(1/\varepsilon))\}}\Big)$ on graphs drawn from $\mathcal{G}(G, \varepsilon)$. If $\frac{\ln(1/\varepsilon)}{\varepsilon} \leq \frac{n^2}{|E|}$, i.e., ε is large enough, this simplifies to $O(\sqrt{n\varepsilon})$. The approximation ratio is not only achieved in expectation, but also with a probability of at least $1 - \exp(-\Omega(n\varepsilon(\log n)^2))$.*

3 Discussion

We have analyzed the approximability of IS in a semi-random input model. We have presented an approximation algorithm that guarantees an approximation ratio of roughly $O(\sqrt{n\varepsilon})$ in expected polynomial time. A simple greedy algorithm (Algorithm 1) alone achieves an expected approximation ratio of $O(\sqrt{n\varepsilon})$ in worst-case polynomial time. A subtlety of our Algorithm 2 is that it needs the original graph as an additional input. We believe that avoiding this requires new techniques if it can be avoided at all. The reason is that our goal was a *guaranteed approximation ratio*. To achieve this, the algorithm has to know when to switch to brute-force search in order to maintain also an expected polynomial running-time. `GreedyIS` alone, which needs neither the original graph nor ε, has complementary properties: worst-case polynomial running-time but the approximation ratio is only achieved in expectation.

References

[1] Amin Coja-Oghlan. Finding large independent sets in polynomial expected time. *Combin. Probab. Comput.*, 15(5):731–751, 2006.

[2] Uriel Feige. Approximating maximum clique by removing subgraphs. *SIAM J. Discrete Math.*, 18(2):219–225, 2004.

[3] Uriel Feige and Joe Kilian. Heuristics for semirandom graph problems. *J. Comput. System Sci.*, 63(4):639–671, 2001.

[4] Michael Krivelevich and Van H. Vu. Approximating the independence number and the chromatic number in expected polynomial time. *J. Comb. Optim.*, 6(2):143–155, 2002.

[5] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(6):103–128, 2007.

Lattices and maximum flow algorithms in planar graphs

Jannik Matuschke and Britta Peis

Technische Universität Berlin, Institut für Mathematik
Straße des 17. Juni 136, 10623 Berlin, Germany

Key words: Lattice polyhedra, Flows in planar graphs, Left/right relation

1 Introduction

The special case of flows in planar graphs has always played a significant role in network flow theory. The predecessor of Ford and Fulkerson's well-known path augmenting algorithm – and actually the first combinatorial flow algorithm at all – was a special version for s-t-planar networks, i.e., those networks where s and t can be embedded adjacent to the infinite face [3]. The basic idea of this *uppermost path algorithm* is to iteratively augment flow along the "uppermost" non-saturated s-t-path in the planar embedding of the network. In 2006, Borradaile and Klein [1] established an intuitive generalization of this algorithm to arbitrary planar graphs, which relies on a partial order on the set of s-t-paths in the graph, called the *left/right relation*.

We connect these results from planar network flow theory with another field of combinatorial optimization, the optimization on lattice structures. In 1978, Hoffman and Schwartz introduced the notion of *lattice polyhedra* [5], a generalization of Edmond's polymatroids that is based on lattices, and proved total dual integrality of the corresponding inequaltity systems if certain additional properties hold, which are defined below.

Definition 1 *Let E be a finite set, $\mathcal{L} \subseteq 2^E$ and $\preceq$ be a partial order on $\mathcal{L}$. Then $(\mathcal{L}, \preceq)$ is a* lattice *if for any pair of elements $S, T \in \mathcal{L}$, there is a unique largest common lower bound $S \wedge T$ called* meet *and a unique least common upper bound $S \vee T$ called* join*. A lattice is* submodular *if $(S \wedge T) \cap (S \vee T) \subseteq S \cap T$ and $(S \wedge T) \cup (S \vee T) \subseteq S \cup T$ for all $S, T \in \mathcal{L}$. It is* consecutive *if $S \cap U \subseteq T$ for all $S, T, U \in \mathcal{L}$ with $S \preceq T \preceq U$.*

Based on the total dual integrality result by Hoffman and Schwartz, several different versions of two-phase-greedy algorithms were developed by Frank [4]

and Faigle and Peis [2] in order to solve linear programming problems on lattice polyhedra like the packing problem

$$\max \left\{ r^T y \ : \ y \in \mathbb{R}_+^{\mathcal{L}}, \sum_{S \in \mathcal{L}: e \in S} y(S) \leq c(e) \ \forall e \in E \right\}$$

and its dual if $\mathcal{L}$ is a submodular and consecutive lattice and the objective function r is supermodular and monotone [1]. Clearly, the path formulation of the maximum flow problem is a special case of the general packing problem.

Results: We show that the left/right relation induces a submodular lattice on the set of simple s-t-paths in a planar graph. If the network is s-t-planar, this lattice is also consecutive, thus meeting all requirements of Hoffman and Schwartz' framework. This implies that Ford and Fulkerson's uppermost path algorithm is a special case of the two-phase greedy algorithm on lattice polyhedra (with $r \equiv 1$). Even more, this algorithm can also solve a weighted flow problem, if the weights on the paths are supermodular and monotone. An additional result will show that whenever the graph is just planar but not s-t-planar, there is no partial order on the paths that induces a consecutive and submodular lattice.

2 The left/right relation and the path lattice

We are given a directed graph $G = (V, E)$ with a fixed planar embedding, a fixed infinite face f_∞ and two designated vertices $s, t \in V$. In our setting, paths are allowed to use edges in either direction [2], i.e., every path P is represented by a subset of $\overleftrightarrow{E} := \{\overrightarrow{e}, \overleftarrow{e} : e \in E\}$ such that $\overrightarrow{e} \in P$ if P uses the edge e in its forward direction and $\overleftarrow{e} \in P$ if P uses e in backward direction. We denote the set of all simple s-t-paths in G by $\mathcal{P} \subseteq 2^{\overleftrightarrow{E}}$. We will analyze a partial order on $\mathcal{P}$ called left/right relation and show that it induces a submodular lattice (cf. Theorem 2), which is furthermore consecutive if the embedding is s-t-planar (cf. Theorem 3). Finally, we will show that there is no partial order on $\mathcal{P}$ that induces a consecutive and submodular lattice, if there is no s-t-planar embedding of the graph (cf. Theorem 4).

2.1 The left/right relation

The left/right relation as presented in this subsection is a partial order on $\mathcal{P}$ due to Klein [6]. We consider the *cycle space* of G, i.e., the subspace of all those vectors in $\mathbb{R}^E$ that fulfill flow conservation at all vertices. The elements of the cycle space are called *circulations*. The vectors corresponding to the clockwise

[1] These requirements on the weight function r are explained in [2].
[2] The resulting lattice can later be restricted to directed paths, maintaining all its properties.

boundary of the non-infinite faces in the embedded graph form a basis of the cycle space. Thus, for every circulation, there is a unique face potential, i.e., an assignment of numbers to the faces corresponding to the circulation. We say a circulation is *clockwise* if the corresponding face potential is non-negative.

A path $P \in \mathcal{P}$ induces a vector $\delta_P \in \mathbb{R}^E$ by $\delta_P(e) = 1$ if $\overrightarrow{e} \in P$, $\delta_P(e) = -1$ if $\overleftarrow{e} \in P$ and $\delta_P(e) = 0$ otherwise. It is easy to see that for two paths $P, Q \in \mathcal{P}$, the vector $\delta_P - \delta_Q$ is a circulation. We say that P *is left of* Q and write $P \succeq Q$ if this circulation is clockwise. It can be easily verified that the left/right relation arising from this definition is a partial order on $\mathcal{P}$. More details on the definition can be found in [1] and [7].

2.2 The path lattice in planar graphs in general

We give a short description of how to obtain a largest common lower bound of two paths $P, Q \in \mathcal{P}$ with respect to the left/right relation. Let ϕ be the face potential corresponding to the circulation $\delta_P - \delta_Q$. For $S^+ := \{f \in V^* : \phi(f) > 0\}$, we define $\delta^{P \wedge Q} := \delta_P - \sum_{f \in S^+} \phi(f)\delta_f$. The vector $\delta^{P \wedge Q}$ induces a set of darts

$$D^{P \wedge Q} := \{\overrightarrow{e} : \delta^{P \wedge Q}(e) = 1\} \cup \{\overleftarrow{e} : \delta^{P \wedge Q}(e) = -1\}.$$

Intuitively speaking, the set $D^{P \wedge Q}$ is obtained by subtracting the "clockwise part" of the circulation $P - Q$ from P. Unfortunately, this set is not necessarily a simple path. However, by flow decomposition, it can be decomposed into a path and several cycles, which can be shown to be clockwise. From this, it can be derived that the path actually is the meet of P and Q. Analogously, one can construct a set $D^{P \vee Q}$ containing the join of P and Q. A detailed proof of this result can be found in [7].

Theorem 2 *$(\mathcal{P}, \preceq)$ is a submodular lattice with $P \wedge Q$ being the unique simple s-t-path contained in $D^{P \wedge Q}$ and $P \vee Q$ being the unique simple s-t-path contained in $D^{P \vee Q}$.*

Note that the path lattice is not consecutive in general.

2.3 The path lattice in s-t-planar graphs

In case the embedding of G is s-t-planar, meet and join of the path lattice can be characterized in a more convenient way than in the general case, and even more, the lattice turns out to be consecutive. As already mentioned, Ford and Fulkerson used the existence of a unique uppermost path from s to t in every s-t-planar graph for their uppermost path algorithm. Formally, this uppermost path (and likewise a lowermost path) can be defined as the unique path with the infinite face to the left (right) of all of its elements.

For some paths $P, Q \in \mathcal{P}$ let the subgraph of G containing only the edges of P and Q be denoted by $G[E(P \cup Q)]$. It can be shown that P is left of

Q if and only if P is the uppermost path in $G[E(P \cup Q)]$ (or, equivalently, Q is the lowermost path in $G[E(P \cup Q)]$). Given this characterization of the left/right relation in s-t-planar graphs, it is easy to verify that if P and Q are incomparable, join and meet are also the uppermost and lowermost paths of $G[E(P \cup Q)]$. In order to show consecutivity, one verifies that $P \succeq Q \succeq R$ implies that P and R are uppermost and lowermost path of $G[E(P \cup Q \cup R)]$ as well, and thus any dart in $d \in P \cap R$ is a loop in the dual and, by cycle/cut duality, a one-element s-t-cut in the primal graph. This implies $d \in Q$.

Theorem 3 *If the embedding of G is s-t-planar, $(\mathcal{P}, \preceq)$ is a consecutive and submodular lattice with $P \wedge Q$ being the lowermost path in $G[E(P \cup Q)]$ and $P \vee Q$ being the uppermost path in $G[E(P \cup Q)]$.*

As a direct corollary, Ford and Fulkerson's uppermost path algorithm [3] turns out to be a special case of Phase 1 of the two-phase greedy algorithm for submodular lattice polyhedra [2].

2.4 The negative result and a characterization of s-t-planar graphs

As pointed out above, the path lattice is not consecutive on planar graphs in general. It can actually be shown that no graph that is planar but not s-t-planar can be equipped with a partial order of the paths that achieves consecutivity and submodularity at the same time. Together with the above positive result for s-t-planar graphs, we achieve the following characterization of s-t-planar graphs.

Theorem 4 *A graph is s-t-planar if and only if it is planar and there is a partial order on the set of its s-t-paths that induces a consecutive and submodular lattice.*

Sketch of proof. Consider the two graphs K_5^- and $K_{3,3}^-$ that are obtained from the Kuratowski graphs K_5 and $K_{3,3}$ by deleting the edge connecting s and t. It can be shown by very elementary arguments that for both K_5^- and $K_{3,3}^-$ there is no partial order of the s-t-paths that induces a submodular and consecutive lattice. The result then follows by applying Kuratowski's Theorem.

3 Conclusion

We provided an extensive analysis of the left/right relation on the set of s-t-paths in a planar graph. The relation induces a submodular lattice, which is even consecutive if the graph is s-t-planar. The latter result implies that the uppermost path algorithm by Ford and Fulkerson is a special case of the two-phase greedy algorithm for packing problems on submodular lattice polyhedra. We furthermore showed that submodularity and consecutivity cannot be achieved simultaneously by any partial order if the graph is not s-t-planar, thus providing a characterization of this special class of planar graphs.

References

[1] G. Borradaile, P. Klein, An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph, in: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, 2006, pp. 524–533.

[2] U. Faigle, B. Peis, Two-phase greedy algorithms for some classes of combinatorial linear programs, in: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, 2008, pp. 161–166.

[3] L. R. Ford, Jr., D. R. Fulkerson, Maximal flow through a network, Canadian Journal of Mathematics. Journal Canadien de Mathématiques 8 (1956)

[4] A. Frank, Increasing the rooted-connectivity of a digraph by one, Mathematical Programming 84 (3, Ser. B) (1999) 565–576. 399–404.

[5] A. J. Hoffman, D. E. Schwartz, On lattice polyhedra, in: A. Hajnal, V. T. Sós (Eds.), Proceedings of the Fifth Hungerian Colloquium on Combinatorics, Vol. I, Vol. 18 of Colloquia mathematica Societatis János Bolyai, North-Holland, Amsterdam, 1978, pp. 593–598.

[6] P. N. Klein, Multiple-source shortest paths in planar graphs, in: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, 2005, pp. 146–155.

[7] J. Matuschke, Lattices and maximum flow algorithms in planar graphs, Diploma thesis, TU Berlin (2009).

Maximum Δ-edge-colorable subgraphs of class II graphs

Vahan V. Mkrtchyan, [a,1] Eckhard Steffen [a]

[a] *Paderborn Institute for Advanced Studies in Computer Science and Engineering, Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany*

Key words: maximum Δ-edge-colorable subgraph, matching, 2-factor, edge-chromatic number, chromatic index, class II graph

1 Introduction

We consider finite, undirected graphs $G = (V, E)$ with vertex set V and edge set E. The graphs might have multiple edges but no loops. Let $\delta(G)$ and $\Delta(G)$ denote the minimum and maximum degree of a graph G, respectively. A partial proper t-edge-coloring of a graph G is an assignment of colors $\{1, ..., t\}$ to some edges of G such that adjacent edges receive different colors. If θ is a partial proper t-edge-coloring of a graph G and P is a path, then P is called to be $\alpha - \beta$ alternating, if the edges of P are colored by the colors α or β. A partial proper t-edge-coloring of a graph G is called a proper t-edge-coloring (or just t-edge-coloring) of G if all edges are assigned some color. The least number t for which G has a t-edge-coloring is called the chromatic index of G and is denoted by $\chi'(G)$. The classical theorems of Shannon and Vizing state:

Theorem 1 *(Shannon) For any graph G: $\Delta(G) \leq \chi'(G) \leq \lfloor \frac{3\Delta(G)}{2} \rfloor$.*

Theorem 2 *(Vizing) For any graph G: $\Delta(G) \leq \chi'(G) \leq \Delta(G) + \mu(G)$, where $\mu(G)$ is the maximum multiplicity of an edge in G.*

A graph G with $\chi'(G) = \Delta(G) = \Delta$ is *class I*, otherwise it is *class II*. There are long standing open conjectures on the class II graphs, cf. [9]. It is a notorious difficult open problem to characterize class II graphs or even to obtain some insight into their structural properties. This paper focuses on the Δ-colorable

Email addresses: vahanmkrtchyan2002@{ysu.am, ipia.sci.am,yahoo.com} (Vahan V. Mkrtchyan,), es@upb.de (Eckhard Steffen).

[1] The author is supported by a fellowship from the Heinrich Hertz-Stiftung

part of graphs. A subgraph H of G is called *maximum* Δ-edge-colorable, if it is Δ-edge-colorable and contains as many edges as possible. The fraction $|E(H)|/|E(G)|$ is subject of many papers, and e.g. lower bounds are proved for cubic, subcubic or 4-regular graphs, [1,3,5]. The aim of this paper is to prove a general best possible lower bound for all graphs.

Let H be a maximum Δ-edge-colorable subgraph of G, which is properly colored with $\{1, ..., \Delta\}$. Usually, we will refer to edges of $E(G)\backslash E(H)$ as uncolored edges. For a vertex v of G let $C(v)$ be the set of colors that appear at v, and $\bar{C}(v) = \{1, ..., \Delta\}\backslash C(v)$ be the set of colors which are missing at v. Let $e = (v, u) \in E(G)\backslash E(H)$ be an uncolored edge, and $\alpha \in \bar{C}(u)$, $\beta \in \bar{C}(v)$. Since H is a maximum Δ-edge-colorable subgraph of G, we have that $\alpha \in C(v)$ and $\beta \in C(u)$. Consider the $\alpha - \beta$ alternating path P starting from the vertex v. Again, since H is a maximum Δ-edge-colorable subgraph of G, the path P ends in u. Thus P is an even path, which together with the edge e forms an odd cycle. We will denote this cycle by $C^e_{\alpha,\beta,H}$. If the subgraph H is fixed, then we will shorten the notation to $C^e_{\alpha,\beta}$.

The cycles corresponding to uncolored edges, that are the cycles $C^e_{\alpha,\beta}$, play a central role in [6,8] in the study of cubic graphs. One aim of the present paper is to generalize some of these results to arbitrary graphs, and to investigate the maximum Δ-edge-colorable subgraphs. We show that any set of vertex disjoint cycles of a graph G with $\Delta(G) \geq 3$ can be extended to a maximum Δ-edge-colorable subgraph of G. In particular, any 2-factor of a graph with maximum degree at least three can be extended to such a subgraph.

For a graph G let $r_e(G)$ denote the minimum number of edges that should be removed from G in order to obtain a graph H with $\chi'(H) = \Delta(G)$. Let G be a graph and ϕ a $\chi'(G)$-coloring of G with $\chi'(G) = \Delta(G) + k$ $(k \geq 1)$. Let $r'_\phi(G) = \min \sum_{j=1}^{k} |\phi^{-1}(i_j)|$, and define $r'_e(G) = \min_\phi r'_\phi(G)$ as the minimum size of the union of k color-classes in a $\chi'(G)$-edge-coloring of G.

Clearly, $r_e(G) = |E(G)| - |E(H)|$, where H is a maximum Δ-edge-colorable subgraph of G. In [4] it is shown that the complement of any maximum 3-edge-colorable subgraph of a cubic graph is a matching, and hence $r_e(G) = r'_e(G)$ for cubic graphs. This paper generalizes this result to simple graphs. We further prove some bounds for the vertex degrees of a maximum $\Delta(G)$-colorable subgraph H.

2 The main results

The key property of cycles corresponding to uncolored edges that is used in [6,8] is their vertex-disjointness. There are many examples showing that they

can have even common edges in the general case. Despite this, it turns out that, as Theorem 3 demonstrates below, the edge-disjointness of the cycles can be preserved.

Theorem 3 *Let H be any maximum $\Delta(G)$-edge-colorable subgraph of a graph G, and let $E(G) - E(H) = \{e_i = (u_i, v_i) | 1 \leq i \leq n\}$ be the set of uncolored edges. Assume that H is properly edge-colored with colors $1, \ldots, \Delta(G)$. Then there is an assignment of colors $\alpha_1 \in \overline{C}(u_1), \beta_1 \in \overline{C}(v_1), \ldots, \alpha_n \in \overline{C}(u_n), \beta_n \in \overline{C}(v_n)$ to the uncolored edges such that $E(C^{e_i}_{\alpha_i,\beta_i}) \cap E(C^{e_j}_{\alpha_j,\beta_j}) = \emptyset$, for all $1 \leq i < j \leq n$.*

The next Theorem generalizes the result of [4] that any 2-factor of a cubic graph can be extended to a maximum 3-edge-colorable subgraph to arbitrary graphs.

Theorem 4 *Let $\bar{F}$ be any set of vertex-disjoint cycles of a graph G with $\Delta = \Delta(G) \geq 3$. Then there is a maximum Δ-edge-colorable subgraph H of G, such that $E(\bar{F}) \subseteq E(H)$.*

Corollary 1 *Let $\bar{F}$ be any $2-$factor of a graph G with $\Delta(G) \geq 3$. Then there is a maximum $\Delta(G)$-edge-colorable subgraph H of G, such that $E(\bar{F}) \subseteq E(H)$.*

Let G be a graph. The length of the shortest (odd) cycle of the underlying simple graph of G is called *(odd) girth* of G. If $X \subseteq V(G)$, then $\partial_G(X)$ denotes the set of edges with precisely one end in X.

Theorem 5 *Let H be any maximum $\Delta(G)$-edge-colorable subgraph of a graph G. Then*

(1) $|\partial_H(X)| \geq \lceil \frac{|\partial_G(X)|}{2} \rceil$ *for each* $X \subseteq V(G)$.
(2) $d_H(x) \geq \lceil \frac{d_G(x)}{2} \rceil$ *for each vertex* x *of* G.
(3) $\delta(H) \geq \lceil \frac{\delta(G)}{2} \rceil$.

The bounds are best possible.

Theorem 6 *Let G be a graph with girth $g \in \{2k, 2k+1\}$ $(k \geq 1)$, and H a maximum $\Delta(G)$-edge-colorable subgraph of G, then $|E(H)| \geq \frac{2k}{2k+1}|E(G)|$, and the bound is best possible.*

Theorem 7 *Every simple graph G contains a maximum Δ-edge-colorable subgraph such that the uncolored edges form a matching.*

Theorem 7 is equivalent to:

Theorem 8 *For any simple graph G: $r_e(G) = r'_e(G)$.*

It can be shown that a maximum Δ-edge-colorable subgraph of a multigraph can be class II as well. This cannot be the case for simple graphs as the following theorem shows.

Theorem 9 *Let H be a maximum $\Delta(G)$-edge-colorable subgraph of a simple graph G, then $\Delta(H) = \Delta(G)$, i.e. H is class I.*

Theorem 7 says, that every simple class II graph G has a maximum Δ-colorable subgraph H, such that $\chi'(G \backslash E(H)) = 1$. We believe that this can be generalized to multigraphs:

Conjecture 1 *Let G be a graph with $\chi'(G) = \Delta(G) + k$ ($k \geq 0$). Then there is a maximum $\Delta(G)$-colorable subgraph H of G such that $\chi'(G \backslash E(H)) = k$.*

This conjecture is equivalent to the following statement.

Conjecture 2 *For any graph G: $r_e(G) = r'_e(G)$.*

References

[1] M. O. Albertson, R. Haas, Parsimonious edge coloring, Disc. Math. 148 (1996) 1-7

[2] H. A. Kierstead, On the chromatic index of multigraphs without large triangles, J. Comb. Theory, Ser. B 36 (1984) 156-160

[3] V. V. Mkrtchyan, S. Petrosyan, G. Vardanyan, On disjoint matchings in cubic graphs, Disc. Math. to appear, (available at: http://arxiv.org/abs/0803.0134)

[4] V. V. Mkrtchyan, S. Petrosyan, G. Vardanyan, On disjoint matchings in cubic graphs: maximum 3-edge-colorable subgraphs, under review, (available at: http://arxiv.org/abs/0909.2767)

[5] R. Rizzi, Approximating the maximum 3-edge-colorable subgraph problem, Disc. Math. 309 (2009) 4166-4170.

[6] E. Steffen, Classifications and characterizations of snarks, Disc. Math. 188 (1998) 183-203.

[7] E. Steffen, A refinement of Vizing's theorem. Disc. Math. 218 (2000) 289-291.

[8] E. Steffen, Measurements of edge-uncolorability, Disc. Math. 280 (2004) 191-214.

[9] T. R. Jensen, B. Toft, Graph coloring problems, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Inc., New York, 1995.

Interval total colorings of bipartite graphs

P.A. Petrosyan [a,b], A.S. Shashikyan [a] A.Yu. Torosyan [a]

[a] *Department of Informatics and Applied Mathematics, YSU*

[b] *Institute for Informatics and Automation Problems of NAS of RA*

Key words: total coloring, interval coloring, bipartite graph, tree, complete bipartite graph

1 Introduction

A total coloring of a graph G is a coloring of its vertices and edges such that no adjacent vertices, edges, and no incident vertices and edges obtain the same color. The concept of total coloring was introduced by V. Vizing [15] and independently by M. Behzad [3]. The total chromatic number $\chi''(G)$ is the smallest number of colors needed for total coloring of G. In 1965 V. Vizing and M. Behzad conjectured that $\chi''(G) \leq \Delta(G) + 2$ for every graph G [3,15], where $\Delta(G)$ is the maximum degree of a vertex in G. This conjecture became known as Total Coloring Conjecture [5]. It is known that Total Coloring Conjecture holds for cycles, for complete graphs, for bipartite graphs, for complete multipartite graphs [17], for graphs with a small maximum degree [6,11,14], for graphs with minimum degree at least $\frac{3}{4}|V(G)|$ [5] and for planar graphs G with $\Delta(G) \neq 6$ [5,7,13,16]. M. Rosenfeld [11] and N. Vijayaditya [14] independently proved that the total chromatic number of graphs G with $\Delta(G) = 3$ is at most 5. A. Kostochka in [6] proved that the total chromatic number of graphs with $\Delta(G) \leq 5$ is at most 7. The general upper bound for the total chromatic number was obtained by M. Molloy and B. Reed [8], who proved that $\chi''(G) \leq \Delta(G) + 10^{26}$ for every graph G. The exact value of the total chromatic number is known only for paths, cycles, complete and complete bipartite graphs, n-dimensional cubes, complete multipartite graphs of odd order [5], outerplanar graphs [18] and planar graphs G with $\Delta(G) \geq 9$ [4,5,7,16].

Email addresses: pet_petros@ipia.sci.am (P.A. Petrosyan), anishashikyan@gmail.com (A.S. Shashikyan), arman.yu.torosyan@gmail.com (A.Yu. Torosyan).

The key concept discussed in this work is the following. Given a graph G, we say that G is interval total colorable if there is $t \geq 1$ for which G has a total coloring with colors $1, 2, \ldots, t$ such that at least one vertex or edge of G is colored by i, $i = 1, 2, \ldots, t$, and the edges incident with each vertex v together with v are colored by $d_G(v) + 1$ consecutive colors, where $d_G(v)$ is the degree of the vertex v in G.

The concept of interval total coloring [9,10] is a new one in graph coloring, synthesizing interval colorings [1,2] and total colorings. The introduced concept is valuable as it connects to the problems of constructing a timetable without a "gap"and it extends to total colorings of graphs one of the most important notions of classical mathematics - the one of continuity.

In this work interval total colorings of bipartite graphs are investigated.

2 Main results

All graphs considered in this work are finite, undirected, and have no loops or multiple edges. Let $V(G)$ and $E(G)$ denote the sets of vertices and edges of G, respectively. An (a, b)-biregular bipartite graph G is a bipartite graph G with the vertices in one part have degree a and the vertices in the other part have degree b. An interval total t-coloring of a graph G is a total coloring of G with colors $1, 2, \ldots, t$ such that at least one vertex or edge of G is colored by i, $i = 1, 2, \ldots, t$, and the edges incident to each vertex v together with v are colored by $d_G(v) + 1$ consecutive colors. The set of all interval total colorable graphs is denoted by $\mathfrak{T}$. For a graph $G \in \mathfrak{T}$, the least (the minimum span) and the greatest (the maximum span) values of t for which G has an interval total t-coloring is denoted by $w_\tau(G)$ and $W_\tau(G)$, respectively. Clearly,

$$\chi''(G) \leq w_\tau(G) \leq W_\tau(G) \leq |V(G)| + |E(G)| \text{ for every graph } G \in \mathfrak{T}.$$

First, we give a general upper bound for the maximum span in interval total colorings of bipartite graphs.

Theorem 1. If G is a connected bipartite graph and $G \in \mathfrak{T}$, then

$$W_\tau(G) \leq 2|V(G)| - 1.$$

Note that the bound of Theorem 1 is sharp for the simple path P_n, since $W_\tau(P_n) = 2n - 1$ for any $n \in \mathbf{N}$.

Next, we show that many bipartite graphs such as subcubic bipartite graphs, regular bipartite graphs, trees, complete bipartite graphs, n-dimensional cubes, $(2, b)$-biregular bipartite graphs, doubly convex bipartite graphs, grids and

some classes of bipartite graphs with maximum degree 4 have interval total colorings. Moreover, we also obtain some bounds for the minimum span and the maximum span in interval total colorings of these graphs. In particular, we prove the following theorems.

Theorem 2. If G is a bipartite graph with $\Delta(G) \leq 3$, then $G \in \mathfrak{T}$ and $w_\tau(G) \leq 5$.

Theorem 3. If G is an r-regular bipartite graph with $r \geq 2$, then $G \in \mathfrak{T}$ and $r+1 \leq w_\tau(G) \leq r+2$.

Note that for any $r \geq 3$, there is an r-regular bipartite graph such that $G \in \mathfrak{T}$ and $w_\tau(G) = r+2$. In [12], it was proved that the problem of determining whether $\chi''(G) = 4$ is NP-complete even for a cubic bipartite graph G. Therefore we can conclude that verification whether $w_\tau(G) = r+1$ for an r-regular bipartite graph G with $r \geq 3$ is also NP-complete.

Theorem 4. If T is a tree, then $T \in \mathfrak{T}$ and $w_\tau(T) \leq \Delta(T) + 2$.

Theorem 5. If $m+n+2-\gcd(m,n) \leq t \leq m+n+1$, where $\gcd(m,n)$ is the greatest common divisor of m and n, then the complete bipartite graph $K_{m,n}$ has an interval total t-coloring for any $m, n \in \mathbf{N}$.

Theorem 6. For any $m, n \in \mathbf{N}$

$$W_\tau(K_{m,n}) = \begin{cases} m+n+1, & \text{if } m = n = 1, \\ m+n+2, & \text{otherwise.} \end{cases}$$

Theorem 7. For the n-dimensional cube Q_n, we have $Q_n \in \mathfrak{T}$ with

$$w_\tau(Q_n) = \chi''(Q_n) \text{ and } W_\tau(Q_n) \geq \tfrac{(n+1)(n+2)}{2} \text{ for any } n \in \mathbf{N}.$$

Moreover, for the n-dimensional cube Q_n, we show that if $w_\tau(Q_n) \leq t \leq \frac{(n+1)(n+2)}{2}$, then Q_n admits an interval total t-coloring.

Theorem 8. If G is a $(2,b)$-biregular bipartite graph with $b \geq 3$, then $G \in \mathfrak{T}$ and $b+1 \leq w_\tau(G) \leq b+2$.

Finally, we show that there are bipartite graphs which have no interval total coloring. The smallest known bipartite graph with 26 vertices and maximum degree 18 that is not interval total colorable was obtained by A. Shashikyan.

References

[1] A.S. Asratian, R.R. Kamalian, Interval colorings of edges of a multigraph, Appl. Math. 5 (1987) 25-34 (in Russian).

[2] A.S. Asratian, R.R. Kamalian, Investigation on interval edge-colorings of graphs, J. Combin. Theory Ser. B 62 (1994) 34-43.

[3] M. Behzad, Graphs and their chromatic numbers, Ph.D. thesis, Michigan State University, 1965.

[4] O.V. Borodin, A.V. Kostochka, D.R. Woodall, Total colorings of planar graphs with large maximum degree, J. Graph Theory 26 (1997) 53-59.

[5] T.R. Jensen, B. Toft, Graph coloring problems, Wiley Interscience Series in Discrete Mathematics and Optimization, 1995.

[6] A.V. Kostochka, The total coloring of a multigraphs with maximal degree 5 is at most seven, Discrete Mathematics 162 (1996) 199-214.

[7] L. Kowalik, J.-S. Sereni, R. Skrekovski, Total-colouring of plane graphs with maximum degree nine, SIAM J. Discrete Math. 22 (2008) 1462-1479.

[8] M. Molloy, B. Reed, A bound on the total chromatic number, Combinatorica 18 (1998) 241-280.

[9] P.A. Petrosyan, Interval total colorings of complete bipartite graphs, Proceedings of the CSIT Conference (2007) 84-85.

[10] P.A. Petrosyan, Interval total colorings of certain graphs, Mathematical Problems of Computer Science 31 (2008) 122-129.

[11] M. Rosenfeld, On the total coloring of certain graphs, Israel J. Math. 9 (1971) 396-402.

[12] A. Sanchez-Arroyo, Determining the total colouring number is NP-hard, Discrete Mathematics 78 (1989) 315-319.

[13] D.P. Sanders, Y. Zhao, On total 9-coloring planar graphs of maximum degree seven, J. Graph Theory 31 (1999) 67-73.

[14] N. Vijayaditya, On the total chromatic number of a graph, J. London Math. Soc. (2) 3 (1971) 405-408.

[15] V.G. Vizing, Chromatic index of multigraphs, Doctoral Thesis, Novosibirsk, 1965 (in Russian).

[16] W. Wang, Total chromatic number of planar graphs with maximum degree ten, J. Graph Theory 54 (2006) 91-102.

[17] H.P. Yap, Total colorings of graphs, Lecture Notes in Mathematics 1623, Springer-Verlag, Berlin, 1996.

[18] Z. Zhang, J. Zhand, J. Wang, The total chromatic numbers of some graphs, Scientia Sinica A 31 (1988) 1434-1441.

Pixel Guards in Polyominoes

Val Pinciu

Department of Mathematics, Southern Connecticut State University, New Haven, CT 06515

Key words: art gallery problem, pixel, polyomino

1 Introduction

The original art gallery problem, posed by Klee in 1973, asks to find the minimum number of guards sufficient to cover any polygon with n vertices. The first solution to this problem was given by Chvátal [1], who proved that $\lfloor n/3 \rfloor$ guards are sometimes necessary, and always sufficient to cover a polygon with n vertices. Later Fisk [2] provided a shorter proof of Chvátal's theorem using an elegant graph coloring argument. Klee's art gallery problem has since grown into a significant area of study. Numerous *art gallery problems* have been proposed and studied with different restrictions placed on the shape of the galleries or the powers of the guards. (See the monograph by O'Rourke [4], and the surveys by Shermer [5] and Urrutia [6].)

In this paper we consider a variation of the art gallery problem where the gallery is an *m-polyomino*, consisting of a connected union of m 1×1 unit squares called *pixels*. Throughout this paper P_m denotes an m-polyomino. We say that a point $a \in P_m$ *covers* a point $b \in P_m$ provided $a = b$, or the line segment ab does not intersect the exterior of P_m. We say that a pixel A *covers* a point b, provided some point $a \in A$ covers b. A set of points G is called a *point guard set* for P_m if for every point $b \in P_m$ there is point $a \in G$ such that a covers b. A set of pixels $\mathcal{G}$ is called a *pixel guard set* for P_m if for every point $b \in P_m$ there is a pixel $A \in \mathcal{G}$ such that A covers b.

In [3], Irfan et al. show that $\lceil \frac{m-1}{3} \rceil$ point guards are sufficient and sometimes necessary to cover any m-polyomino P_m, with $m \geq 2$. They also note that $\lceil \frac{m-1}{3} \rceil$ is an upper bound for the minimum number of pixel guards sufficient to cover any m-polyomino. In this paper we improve this bound, showing that an m-polyomino always has a pixel guard set of cardinality $\lfloor \frac{m+1}{11} \rfloor + \lfloor \frac{m+5}{11} \rfloor + \lfloor \frac{m+9}{11} \rfloor$. We also show that this bound is sharp, by constructing m-polyominoes that require exactly $\lfloor \frac{m+1}{11} \rfloor + \lfloor \frac{m+5}{11} \rfloor + \lfloor \frac{m+9}{11} \rfloor$ pixel guards.

2 Main Results

Here is our main result:

Theorem 1 *For any m-polyomino P_m with $m \geq 2$, $\lfloor \frac{m+1}{11} \rfloor + \lfloor \frac{m+5}{11} \rfloor + \lfloor \frac{m+9}{11} \rfloor$ pixel guards are always sufficient, and sometimes necessary to cover P_m.*

Proof. We will use a construction to prove the necessity part of our result. The polyomino P_{11k+2} from Figure 1 has $3+7k+4(k-1)+3 = 11k+2$ pixels. The dual graph of this polyomino is a tree with $1+2k+(k-1)+1 = 3k+1$ leaves. Since two pixels that correspond to a leaf cannot be guarded by the same pixel guard, then the number of pixels required to guard P_{11k+2} is at least $3k+1$. Simple alterations of this construction can provide examples of m-polyominos that require at least $\lfloor \frac{m+1}{11} \rfloor + \lfloor \frac{m+5}{11} \rfloor + \lfloor \frac{m+9}{11} \rfloor$ pixel guards, for any integer $m \geq 2$. Next we will prove several technical lemmas, and the sufficiency will follow from Proposition 2. □

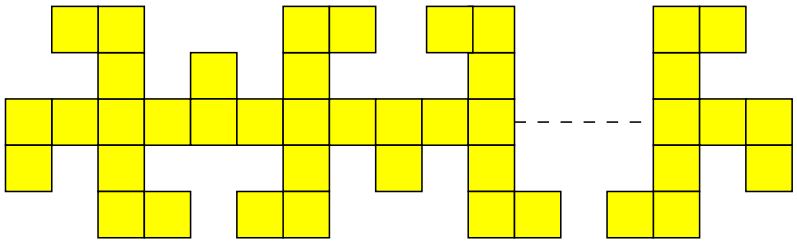


Fig. 1. An $(11k+2)$-polyomino that requires $3k+1$ pixel guards.

Lemma 1 *For each positive integer m we define*

$$f(m) = \left\lfloor \frac{m+1}{11} \right\rfloor + \left\lfloor \frac{m+5}{11} \right\rfloor + \left\lfloor \frac{m+9}{11} \right\rfloor$$

Then the following are true:

(1) $f(m+3) \leq f(m)+1 \leq f(m+4)$ *for all positive integers m.*
(2) $f(m+7) \leq f(m)+2 \leq f(m+8)$ *for all positive integers m.*
(3) $f(m+11) = f(m)+3$ *for all positive integers m.*
(4) $f(m+n-2) \geq f(m)+f(n)-1$ *for all positive integers m and n.*

Lemma 2 *For any m-polyomino P_m with $m \geq 13$, there exists a k, $4 \leq k \leq 10$ such that P_m is the union of a k-polyomino P_k and an $(m-k)$-polyomino P_{m-k}. Moreover, if the smallest k that satisfies this property is $k = 10$, then we can assume that exactly one pixel of P_{m-10} is adjacent to P_{10}.*

Proof. Given an m-polyomino P_m, let G_m^* be the dual graph of P_m, and let T_m be a spanning tree of G_m^*. Since every vertex of G_m^* has maximum degree

4, we can look at T_m as a rooted trinary tree. For simplicity, we will label the vertices of the rooted tree as the corresponding pixels in the dual polyomino. We will also transfer the common terminology from rooted trees (child, parent, sibling, etc) to the corresponding pixels. For any vertex A of T_m that is not the root, we can obtain a spanning forest of G_m^* with two components, by deleting the edge that connects A with its parent. These two componenets will generate a decomposition of P_m into two polyominoes: a k-polyomino P_k that contains A, called the *polyomino generated by A and T_m*, and another polyomino P_{m-k} that does not contain A. If B is a pixel of P_{m-k} and C is a pixel of P_k such that B and C are adjacent, we can create another spanning tree T'_m of G_m^* by replacing the edge that connects C with its parent in T_m with the edge BC. We will call this an *adoption* and say that B adopted C. An adoption will transfer a pixel of the polyomino generated by A, and all its descendants to the complementary polyomino. Now if h is the height of T_m, we consider the pixels of level $h-1$, $h-2$, $h-3$, or $h-4$ that have at least three descendants. Obviously this set is not empty. Let A be such a pixel with a minimum number of descendants. Then one can show that the polyomino generated by A satisfies the conditions of the proposition, or we can do an adoption to decrease the number of descendants of A. □

Lemma 3 *(1) One pixel guard is always sufficient to cover any 5-polyomino.*
(2) Two pixel guards are always sufficient to cover any 9-polyomino.
(3) Three pixel guards are always sufficient to cover any 12-polyomino.

Proposition 2 *For any m-polyomino P_m, if $m \geq 2$, then $\lfloor \frac{m+1}{11} \rfloor + \lfloor \frac{m+5}{11} \rfloor + \lfloor \frac{m+9}{11} \rfloor$ pixel guards are sufficient to cover P_m.*

Proof. The proof of this proposition is by induction on m. If $2 \leq m \leq 12$, the statement follows from Lemma 3. If $m \geq 13$, then by Lemma 2, P_m is the union of a k-polyomino P_k and an $(m-k)$-polyomino P_{m-k}, where $4 \leq k \leq 10$. Assume k is the smallest with this property. Let $f(m)$ be the function from Lemma 1. Then by induction hypothesis the minimum number of pixel guards required to watch P_{m-k} is $g(P_{m-k}) \leq f(m-k)$.
If $k = 4$ or $k = 5$, we obtain:
$g(P_m) \leq g(P_k) + g(P_{m-k}) \leq 1 + g(P_{m-k}) \leq 1 + f(m-4) \leq f(m)$.
If $k = 8$ or $k = 9$, we obtain:
$g(P_m) \leq g(P_k) + g(P_{m-k}) \leq 2 + g(P_{m-k}) \leq 2 + f(m-8) \leq f(m)$.

If $k = 6$, we should note that 33 out of the 35 possible hexaminoes can be covered by only one pixel guard, and we can use an argument similar with the case $k = 4$ or $k = 5$. Otherwise, if P_k requires two pixel guards, let A be the pixel that generated P_k, and let B be the parent of A. If B is the only pixel in P_{m-k} adjacent to P_k, then one can show that P_{m-k} has a pixel guard set of cardinality $f(m-4)$ that contains B. Then B can also be used to guard part of P_k, and we need only one additional guard. Otherwise, let C be a pixel

in P_{m-k} adjacent to P_k. Then C can adopt a descendant of A, reducing the problem to the case $k = 4$ or $k = 5$, or C is a leaf, in which case we can remove B and C from P_{m-k}, add them to P_k, and reduce the problem to the case $k = 8$. (note that the minimality of k was not used in the case $k = 8$.)
If $k = 7$, let A be the pixel that generated P_k, and let B be its parent. Then this case can also be reduced to the one of the cases $k = 4$, $k = 5$, or $k = 8$, or we can show that B has exactly two children. In this last case, let C be the other child of B, and let D be the parent of B. If in T_m we remove the edge BC, and the edge that connects D with its parent, we can obtain a decomposition of P_m into three polyominos. One of them is 9-polyomino. Using the induction hypothesis and Lemma 1 we obtain:
$g(P_m) \leq g(P_9) + g(P_l) + g(P_{m-l-9}) \leq 2 + f(l) + f(m - l - 9)$
$\leq 2 + f(l + m - l - 9 - 2) + 1 = 3 + f(m - 11) = f(m)$.
Finally, if $k = 10$, since k is the smallest that satisfies the property from Lemma 2, we can assume that exactly one pixel of P_{m-10} is adjacent to P_{10}. Then by removing this pixel from P_{m-10}, and adding it to P_{10}, we can assume that P_m is the union of an 11-polyomino P_{11} and an $(m-11)$-polyomino P_{m-11}. Then $g(P_m) \leq g(P_{11}) + g(P_{m-11}) \leq 3 + g(P_{m-11}) \leq 3 + f(m-11) = f(m)$. □

References

[1] V. Chvátal, A combinatorial theorem in plane geometry: *J. Combin. Theory Ser. B* **18** (1975) 39–41.

[2] S. Fisk, A short proof of Chvátal's watchman theorem: *J. Combin. Theory Ser. B* **24** (1978) 374.

[3] M. Irfan, J. Iwerks, J. Kim, J. Mitchell, Guarding Polyominoes: *19th Annual Workshop on Computational Geometry* (2009).

[4] J. O'Rourke: *Art Gallery Theorems.* Oxford University Press, 1987.

[5] T.C. Shermer: Recent results in art gallery theorems, *Proc. IEEE* **80** (1992) 1384–1399.

[6] J. Urrutia: Art gallery and illumination problems, in: *Handbook of Computational Geometry,* J.-R Sack and J. Urrutia (Eds.), Elsevier Science B. V., 1999, pp. 973–1027.

Mixed connectivity of Cartesian graph products and bundles

Rija Erveš [a,b], Janez Žerovnik [a,c]

[a] *Institute of Mathematics, Physics and Mechanics, Jadranska 19, Ljubljana 1000, Slovenia.*

[b] *FCE, University of Maribor, Smetanova 17, Maribor 2000, Slovenia.*

[c] *FME, University of Ljubljana, Aškerčeva 6, Ljubljana 1000, Slovenia.*

Key words: vertex connectivity, edge connectivity, mixed connectivity, Cartesian graph bundle, Cartesian graph product, interconnection network, fault tolerance.

1 Introduction

An interconnection network should be fault tolerant, because practical communication networks are exposed to failures of network components. Both failures of nodes and failures of connections between them happen and it is desirable that a network is robust in the sense that a limited number of failures does not break down the whole system. A lot of work has been done on various aspects of network fault tolerance, see for example the survey [6] and more recent papers [9,12,14]. In particular the fault diameter with faulty vertices which was first studied in [10] and the edge fault diameter has been determined for many important networks recently [1–4,7,8,11,13]. Usually either only edge faults or only vertex faults are considered, while the case when both edges and vertices may be faulty is studied rarely. In recent work on fault diameter of Cartesian graph products and bundles [1–4], analogous results were found for both fault diameter and edge fault diameter. However, the proofs for vertex and edge faults are independent, and our effort to see

Email addresses: rija.erves@uni-mb.si (Rija Erveš), janez.zerovnik@imfm.uni-lj.si (Janez Žerovnik).

how results in one case may imply the others was not successful. A natural question is whether it is possible to design a uniform theory that would enable unified proofs or provide tools to translate results for one type of faults to the other. It is therefore of interest to study general relationships between invariants under simultaneous vertex and edge faults. Some basic results on edge, vertex and mixed fault diameters for general graphs appear in [5]. In order to study the fault diameters of graph products and bundles under mixed faults, it is important to understand the generalized connectivities. We define mixed connectivity which generalizes both vertex and edge connectivity, and observe some basic facts for any connected graph. Furthermore, we generalize results of vertex connectivity and edge connectivity of Cartesian graph bundles [1,4]. As a corollary, mixed connectivity of the Cartesian product of finite number of factors is given. In particular Theorem 3.2 improves the result on edge connectivity of Cartesian graph products and bundles.

2 Mixed connectivity

Definition 2.1 *Let G be any connected graph. A graph G is* (p, q)+connected*, if G remains connected after removal of any p vertices and any q edges.*

Any connected graph G is $(0, 0)$+connected, $(p, 0)$+connected for any $p < \kappa(G)$ and $(0, q)$+connected for any $q < \lambda(G)$, where $\kappa(G)$ and $\lambda(G)$ are the usual vertex- and edge-connectivities. In our notation $(i, 0)$+connected is the same as $(i + 1)$-connected, i.e. the graph remains connected after removal of any i vertices. Similarly, $(0, j)$+connected is the same as $(j + 1)$-edge connected, i.e. the graph remains connected after removal of any j edges. Clearly, if G is (p, q)+connected graph, then G is (p', q')+connected for any $p' \leq p$ and any $q' \leq q$. Furthermore, for any connected graph G with $k < \kappa(G)$ faulty vertices, at least k edges are not in functional. Roughly speaking, graph G remains connected if any faulty vertex in G is replaced with any edge. It is easy to prove that if a graph G is (p, q)+connected and $p > 0$, then G is $(p - 1, q + 1)$+connected. Hence for $p > 0$ we have a chain of implications: (p, q)+connected $\Longrightarrow$ $(p - 1, q + 1)$+connected $\Longrightarrow \ldots \Longrightarrow (1, p - 1 + q)$+connected $\Longrightarrow (0, p + q)$+connected, that generalizes the well-known proposition that any k-connected graph is also k-edge connected. Therefore, a graph G is (p, q)+connected if and only if $p < \kappa(G)$ and $p + q < \lambda(G)$.

If for a graph G $\kappa(G) = \lambda(G) = k$, then G is (i, j)+connected exactly when $i + j < k$. However, if $2 \leq \kappa(G) < \lambda(G)$, the question whether G is (i, j)+connected for $1 \leq i < \kappa(G) \leq i + j < \lambda(G)$ is not trivial. The example below shows that in general knowing $\kappa(G)$ and $\lambda(G)$ is not enough to decide whether G is (i, j)+connected.

Example 2.2 *For graphs on Fig. 1 we have* $\kappa(G_1) = \kappa(G_2) = 2$ *and* $\lambda(G_1) = \lambda(G_2) = 3$. *Both graphs are* $(1,0)$+*connected* $\Longrightarrow$ $(0,1)$+*connected, and* $(0,2)$+*connected. Graph* G_1 *is not* $(1,1)$+*connected, while graph* G_2 *is.*

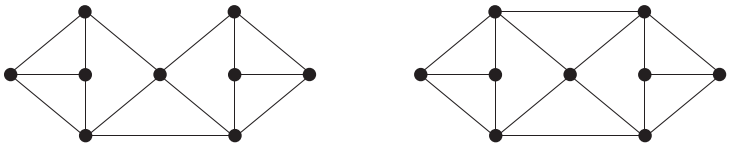


Fig. 1. Graphs G_1 and G_2 from Example 2.2.

Both edge connectivity and vertex connectivity of a graph can be computed in polyonimal time. Therefore it is interesting to ask

Problem. Let G be a graph and $1 \leq i < \kappa(G) \leq i + j < \lambda(G)$. Is there a polynomial algorithm to decide whether G is (i,j)+connected?

3 Mixed connectivity of Cartesian graph products and bundles

Graph products and bundles are among frequently studied interconnection network topologies. For example the meshes, tori, hypercubes and some of their generalizations are Cartesian products. It is less known that some well-known topologies are Cartesian graph bundles, i.e. some twisted hypercubes and multiplicative circulant graphs. Graph bundles also appear as computer topologies. A well known example is the twisted torus, a Cartesian graph bundle with fibre C_4 over base C_4 is the ILLIAC IV architecture, a famous supercomputer that inspired some modern multicomputer architectures. It may be interesting to note that the original design was a graph bundle with fibre C_8 over base C_8, but due to high cost a smaller version was build. A Cartesian graph bundle is a generalization of graph cover and the Cartesian graph product.

Definition 3.1 *Let* B *and* F *be graphs. A graph* G *is a* Cartesian graph bundle with fibre F over the base graph B *if there is a* graph map $p : G \to B$ *such that for each vertex* $v \in V(B)$, $p^{-1}(\{v\})$ *is isomorphic to* F, *and for each edge* $e = uv \in E(B)$, $p^{-1}(\{e\})$ *is isomorphic to* $F \Box K_2$.

We have generalized the result [1] on (vertex) connectivity and improved the result [4] on edge connectivity:

Theorem 3.2 *Let* G *be a Cartesian graph bundle with fibre* F *over the base graph* B, *graph* F *be* (p_F, q_F)+*connected and graph* B *be* (p_B, q_B)+*connected. Then Cartesian graph bundle* G *is* $(p_F + p_B + 1, q_F + q_B)$+*connected.*

As the Cartesian product is a Cartesian graph bundle where all the isomor-

phisms between the fibres are identities, the statement about mixed connectivity of Cartesian graph products of a finite number of factors follows easily from Theorem 3.2.

Corollary 3.3 *Let graphs $G_i, i = 1, \ldots, k$, be (p_i, q_i)+connected. Then the Cartesian graph product $G = G_1 \Box G_2 \Box \ldots \Box G_k$ is $(\sum p_i + k - 1, \sum q_i)$+connected.*

References

[1] I. Banič, J. Žerovnik, Fault-diameter of Cartesian graph bundles, Inform. Process. Lett. 100 (2006) 47–51.

[2] I. Banič, J. Žerovnik, Edge fault-diameter of Cartesian product of graphs, Lecture Notes in Comput. Sci. 4474 (2007) 234-245.

[3] I. Banič, J. Žerovnik, Fault-diameter of Cartesian product of graphs, Adv. in Appl. Math. 40 (2008) 98–106.

[4] I. Banič, R. Erveš, J. Žerovnik, The edge fault-diameter of Cartesian graph bundles, European J. Combin. 30 (2009) 1054–1061.

[5] I. Banič, R. Erveš, J. Žerovnik, Edge, vertex and mixed fault diameters, Adv. in Appl. Math. 43 (2009) 231–238.

[6] J.-C. Bermond, N. Honobono, C. Peyrat, Large Fault-tolerant Interconnection Networks, Graphs Combin. 5 (1989) 107-123.

[7] K. Day, A. Al-Ayyoub, Minimal fault diameter for highly resilient product networks, IEEE Trans. Parallel. Distrib. Syst. 11 (2000) 926–930.

[8] D. Z. Du, D. F. Hsu, Y. D. Lyuu, On the diameter vulnerability of kautz digraphs, Discrete Math. 151 (2000) 81–85.

[9] C. H. Hung, L. H. Hsu, T. Y. Sung, On the Construction of Combined k-Fault-Tolerant Hamiltonian Graphs, Networks 37 (2001) 165-170.

[10] M. Krishnamoorthy, B. Krishnamurty, Fault diameter of interconnection networks, Comput. Math. Appl. 13 (1987) 577–582.

[11] S. C. Liaw, G. J. Chang, F. Cao, D. F. Hsu, Fault-tolerant routing in circulant networks and cycle prefix networks, Ann Comb. 2 (1998) 165–172.

[12] C. M. Sun, C. N. Hung, H. M. Huang, L. H. Hsu, Y. D. Jou, Hamiltonian Laceability of Faulty Hypercubes, Journal of Interconnection Networks 8 (2007) 133-145.

[13] M. Xu, J.-M. Xu, X.-M. Hou, Fault diameter of Cartesian product graphs, Inform. Process. Lett. 93 (2005) 245–248.

[14] J. H. Yin, J. S. Li, G. L. Chen, C. Zhong, On the Fault-Tolerant Diameter and Wide diameter of ω-Connected Graphs, Networks 45 (2005) 88–94.

Wide - sense nonblocking $\log_d(N, 0, p)$ networks

Maja Rotovnik [a] Janez Žerovnik [a,b]

[a] *IMFM, Jadranska 19, SI-1000 Ljubljana, Slovenia*

[b] *FME, University of Ljubljana, Aškerčeva 6, SI-1000 Ljubljana, Slovenia*

Key words: switching network, WSNB, strong product, chromatic number

1 Introduction

One of the common architectures that are used in high-speed photonic and electronic switching networks is the $\log_d(N, m, p)$ switching network [1,2]. In this paper we consider a special case of such network, $\log_d(N, 0, p)$ network. Usually, the architecture is accompanied with the control algorithm that works online, i.e. after arrival of a connection request, a path through the network is assigned that connects the input and output of the new connection [3,4]. A switching network is wide sense nonblocking (WSNB) if a new call is always routable as long as all previous requests were routed according to a given routing algorithm [5]. WSNB switching networks were first introduced by Beneš [6] for symmetric 3−stage Clos network. He proved that $C(n, m, 2)$ is WSNB if and only if $m \geq \left\lfloor \frac{3n}{2} \right\rfloor$.

Here we introduce an auxiliary *path graph* of switching network such that the number of required planes is the chromatic number of the path graph. From the structure of the path graphs it follows that the minimal p is $d^{\lfloor \frac{n}{2} \rfloor} - 1$ for every n. Furthermore, our analysis implies that at least p planes are needed regardless of the control algorithm used.

A special case of this result was proved for a particular algorithm and even n (for $d = 2$ in [7] and for general d in [8]). For other cases only bounds in terms of so-called non-blocking conditions were known [5,9,10].

Email addresses: maja.rotovnik@imfm.si (Maja Rotovnik), janez.zerovnik@imfm.uni-lj.si (Janez Žerovnik).

2 Basic definitions and notations

The basic conponents of switching network are crossbar switches and links which connect switches. The $\log_d(N, 0, p)$ switching network consist of p copies of $\log_d(N, 0, 1)$, called the planes. Each plane contains $n \cdot d^{n-1}$ switches divided into n stages. In each stage there are d^{n-1} switches and each switch has d inputs and d outputs. Inputs and outputs are numbered $0, 1, ..., N-1$, $N = d^n$, from top to bottom, and stages are numbered $0, 1, ..., n$ from left to right. Examples of $\log_2(8, 0, 1)$, $\log_2(16, 0, 1)$, and $\log_3(27, 0, 1)$ switching networks are given on figures 1, 2, and 3 respectively.

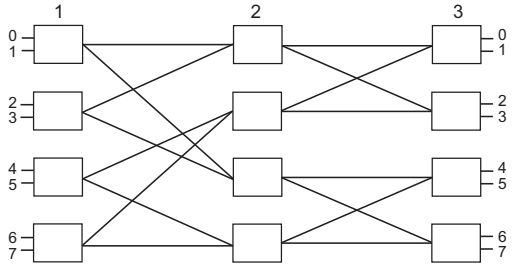


Fig. 1. Example of switching network for $n = 3$ and $d = 2$.

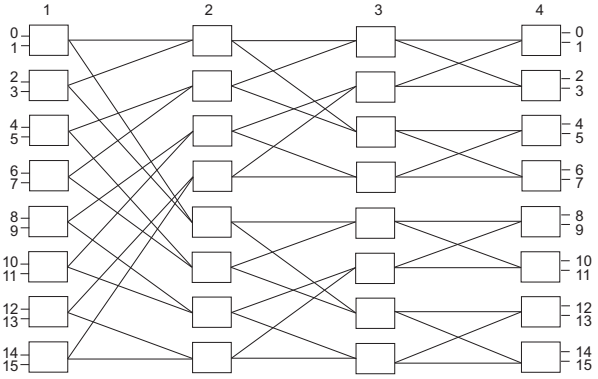


Fig. 2. Example of switching network for $n = 4$ and $d = 2$.

Definition: The **path graph** $G_P(n, d)$ of a switching network $\log_d(N = d^n, 0, 1)$ is the graph with a vertex (i, j) for every connection $\langle i, j \rangle$ in the switching network. Two vertices (i_1, j_1) and (i_2, j_2) are adjacent in G_P if connections $\langle i_1, j_1 \rangle$ and $\langle i_2, j_2 \rangle$ share an interstage link in switching network.

For example in figure 2 the connections $< 0, 0 >, < 1, 5 >$ share the link between stages 1 and 2, so they are adjacent. The connections $< 0, 0 >, < 2, 9 >$ are independent because they do not share a link.

Proposition. The **path graph** $G_P(n, d)$ is in the worst case isomorphic to

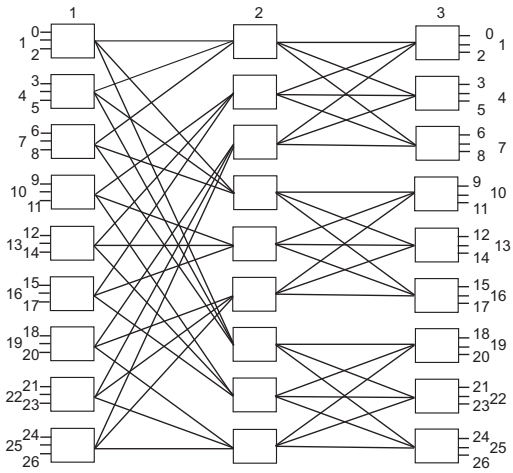


Fig. 3. Example of switching network for $n = 3$ and $d = 3$.

the strong product $(K_d \square K_d) \boxtimes K_{d^{k-2}}$ when $n = 2k - 1$, $k \in \mathbb{N}$ and isomorphic to the d^k copies of K_{d^k}, when $n = 2k,\ k \in \mathbb{N}$.

Proof ommited in the abstract.

Proposition. The number of planes p needed to make $\log_d(N, 0, p)$ switching network a WSBN is equal to the chromatic number of the path graph.

$$p = \chi\left(G_P(n, d)\right)$$

(Proof ommited in the abstract.) Idea of proof: by construction of the path graph, the vertices corresponding to independent connections are independent and two vertices are connected when the connections share a link. Hence any subset of connections can be can be divided into $p = \chi\left(G_P(n, d)\right)$ independent sets and each of them can be realized in a different plane. On the othere hand, it is easy find a set of connections that forms a clique, so it can not be realized in less than p planes without blocking.

Remark. From details of our construction one can find a formula which assigns the color (the plane) on which it should be realized, so there is no complicated algorithm for assigning the planes needed. Compare to [7].

Recall that the chromatic number of the strong product of graphs is less than or equal to product of chromatic numbers of factors. Equality holds when chromatic number of both factors is equal to their clique number [11].

Hence we have

Theorem: The $\log_d(N, 0, p)$ switching network, $n = \log_d N$, is WSNB if and only if

$$p \geq \begin{cases} d^k, & if\ n = 2k,\ k \in \mathbb{N} \\ d^{k-1}, & if\ n = 2k-1,\ k \in \mathbb{N}. \end{cases}$$

References

[1] C. T. Lea. Multi-$log_2 N$ networks and their applications in high-speed electronic and photonic switching systems. *IEEE Trans. Commun.*, **38** (1990) 1740–1749.

[2] C. T. Lea and D. J. Shyy. Tradeoff of horizontal decomposition versus vertical stacking in rearrangeable nonblocking networks. *IEEE Trans. Commun.*, **39**, (1991) 899904.

[3] W. Kabacinski, M. Michalski. Lower Bounds for WSNB Multi-$log_2 N$ Switching Networks. *Conference on Telecommunications A-ICT 2005*, (Lisbon,Portugal), (2005) 202–206.

[4] G. Danilewicz. Wide-Sense Nonblocking $Log_d(N, 0, p)$ Multicast Switching Networks. *IEEE Trans. Commun.*, **55** (2007) 2193-2200.

[5] F. K. Hwang. *The Mathematical Theory of Nonblocking Switching Networks.* World Scientific, Singapore, first ed., 1998; second ed., 2004.

[6] V. E. Beneš. *Mathematical Theory of Connecting Networks and Telephone Traffic.* Academic Press, New York, 1965.

[7] W. Kabacinski, M. Michalski. Wide-Sense Nonblocking $log_2(N, 0, p)$ Switching Networks with Even Number of Stages. *IEEE ICC 2005*, (Seul, South Korea), (2005).

[8] G. Danilewicz, W. Kabacinski, M. Michalski and M. Zal. Wide-Sense Nonblocking Multiplane Baseline Switching Networks Composed of $d \times d$ Switches. *IEEE ICC 2007*, (Glasgow, Scotland), (2007).

[9] A. Pattavina. *Switching Theory - Architectures and Performance in Broadband ATM Networks.* John Wiley & Sons, England, 1998.

[10] W. Kabacinski. *Nonblocking Electronic and Photonic Switching Fabrics.* Springer, 2005.

[11] W. Imrich, S. Klavžar. *Product graphs: structure and recognition.* JohnWiley & Sons, New York, USA, 2000.

Efficient total domination

Oliver Schaudt

Institut für Informatik, Arbeitsgruppe Faigle/Schrader, Universität zu Köln
Weyertal 80, 50931 Cologne, Germany

Key words:
total domination, induced matchings, neighborhood hypergraphs

1 Introduction

All relevant graph classes and graph class inclusions not defined here are displayed in [1]. For each graph G, $V(G)$ denotes its set of vertices.

Total domination has been introduced 1980 by Cockayne, Dawes and Hedetniemi in [2] and is intensively studied now. A good introduction to the theory of (total) domination, giving a broad overview of the important results and applications, is given in [5]. In the problem of total domination, one is interested in determining the value $\gamma_t(G)$ of a given graph G, defined as the smallest size of a subset $X \subseteq V(G)$ such that each vertex of G has at least one neighbor in X.

Let G be a simple undirected graph. A set $X \subseteq V(G)$ is said to be an *efficiently total dominating set* of G, or an *etd set*, if each $v \in V(G)$ is adjacent to exactly one vertex in X. G is then said to be an *efficiently total dominatable graph*, or G is *etd*. The corresponding decision problem is denoted by *ETD*. Let $\mathbf{1}$ denote the vector with all components equal to 1 of suitable dimension. ETD can alternatively be defined as the class of graphs whose neighborhood hypergraph has a perfect matching, as the class of graphs whose adjacency matrix A accepts the equation $Ax = \mathbf{1}$ for some 01-vector x, and as the class of graphs that have an induced matching, such that each vertex is adjacent to exactly one matched vertex. There is some literature on efficient domination, but in the case of efficient *total* domination, only a few papers have been published so far (according to our knowledge).

Email address: schaudt@zpr.uni-koeln.de (Oliver Schaudt).

A simple but important result mentioned in [5] is the following

Theorem 1 (See [5]) *Let G be an etd graph. Each etd set X of G has cardinality $\gamma_t(G)$.*

We can therefore understand efficient total domination as an extremal case of total domination. Furthermore, understanding the structure of efficiently total dominatable graphs and the algorithmical complexity of the corresponding decision problem may put some light on total domination, too.

2 Main results

Graph classes on which ETD is NP-complete can be obtained by reducing the well known Exact Cover decision problem (EC) to ETD. Given an arbitrary 01-matrix A, EC asks for the 01-solvability of $Ax = 1$. It is possible to reduce EC to ETD in the following way: Let I denote the identity matrix of suitable dimension. Given a 01-matrix A, we define a function

$$A(X) = \begin{pmatrix} X & 0 & 0 & A \\ 0 & 0 & I & I \\ 0 & I & 0 & 0 \\ A^t & I & 0 & 0 \end{pmatrix} \tag{1}$$

and observe for each X, that A is in EC iff $A(X)$ is in EC.

As $A(0)$ is the adjacency matrix of a bipartite graph, $A(0)$ is in EC iff this very graph is in ETD. Let J denote the square matrix with all components equal to 1 of suitable dimension. $A(J - I)$ is the adjacency matrix of a $(1, 2)$-colorable chordal graph, i.e. a chordal graph which can be partitioned into a clique and two independent sets, and $A(J - I)$ is in EC iff this very graph is in ETD. As EC is well known to be NP-complete, we conclude NP-completeness of ETD restricted to bipartite graphs and to $(1, 2)$-colorable chordal graphs. As the class of $(1, 2)$-colorable chordal graphs is only slightly bigger than the class of split graphs (which are exactly the $(1, 1)$-colorable graphs) and ETD restricted to split graphs is trivial, we see that the gap of complexity between the two classes is big compared to their structural differences.

A further result is inspired by an idea stated by Lozin [7] in the context of induced matchings. Let $\mathcal{F}$ be a (not necessarily finite) set of graphs. We set $K(\mathcal{F})$ as the supremum over the lengths of all paths in graphs of $\mathcal{F}$, whose inner vertices have degree 2. For given non negative integers i, j, k, a $star_{i,j,k}$

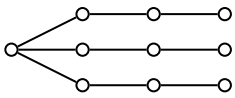


Fig. 1. T_3.

graph is constructed in the following way. Start with three paths consisting of i, j and k vertices. Choose an endvertex of each path and connect these to a single new vertex r. For example, a $star_{k,0,0}$ is a path of length k and a $star_{1,1,1}$ is a claw.

Theorem 2 *Let $\mathcal{F}$ be a set of graphs with finite $K(\mathcal{F})$ such that there is no graph of $\mathcal{F}$ whose every connected component is a $star_{i,j,k}$. ETD restricted to the class of bipartite $\mathcal{F}$-free graphs is NP-complete.*

Choosing $\mathcal{F} = \{K_{1,4}\}$, we see that ETD is NP-complete on the class of bipartite graphs with maximum degree 3. Summarizing our results, we obtain the following

Theorem 3 *ETD is NP-complete when restricted to the following classes:*

- *planar bipartite graphs with maximum degree 3, bipartite graphs, comparability graphs*
- *$(1, 2)$-colorable chordal graphs, chordal graphs, perfect graphs*

In our research we observed that ETD is polynomial time solvable on various classes. A first class can be obtained by using the property of each balanced matrix A [3], that the corresponding set partitioning polytope $\{x : Ax = 1, 0 \leq x \leq 1\}$ only has integral extreme points. Therefore ETD is polynomial solvable on the class of graphs with balanced adjacency matrices (*balanced* graphs, [3]), i.e. graphs which only induce cycles of length four.

Our main results are the polynomial solvability of ETD restricted to claw-free graphs [8] and to T_3-free chordal graphs [9] (T_3 is displayed in Fig. 1).

ETD on claw-free graphs can be reduced to ETD on line graphs in two steps in polynomial time. ETD on line graphs is reducible to the perfect matching problem in certain auxiliary graphs in linear time. Therefore, ETD on claw-free graphs is polynomial time solvable. Furthermore, etd claw-free graphs are necessarily perfect. Thus, our result can be seen as an example for the claim stated in [4], that claw-free perfect graphs often accept polynomial time algorithms for problems which are NP-complete in general.

In the case of T_3-free chordal graphs we use a polynomial time procedure to label the vertices of the graph with 0 and 1, using the well known *perfect elimination ordering* of chordal graphs. Each labeled vertex v is either in every

etd set of the graph, if v is labeled with 1, or in no etd set, if v is labeled with 0. The etd condition restricted to the unlabeled vertices forms, by T_3-freeness, an instance of 2-SAT. It therefore is a polynomial solvable problem.

Summarizing our results, we obtain the following

Theorem 4 *ETD is polynomial solvable when restricted to the following classes:*

- *balanced graphs, chordal bipartite graphs, bipartite permutation graphs*
- *claw-free graphs, line graphs, line graphs of bipartite graphs*
- *T_3-free chordal graphs, interval graphs, circular arc graphs*
- *strongly chordal graphs, doubly chordal graphs*
- *$\overline{C_4}$-free graphs, co-chordal graphs, split graphs = $(1,1)$-colorable graphs*
- *P_4-free graphs = cographs*

If we compare this list with the list of time complexities for total domination given in [6], we see some interesting differences: Total domination on the classes of line graphs of bipartite graphs and split graphs is NP-complete, while polynomially solvable in the case of ETD. On the other hand, ETD is NP-complete restricted to the class of graphs with adjacency matrix $A(0)$ while total domination is trivially decidable on this class.

References

[1] A. Brandstädt, V.B. Le, J. Spinrad, *Graph classes: a survey*, SIAM Monographs on Discrete Math. Appl., Vol. 3, SIAM, Philadelphia, 1999.

[2] E.J. Cockayne, R.M. Dawes and S.T. Hedetniemi, *Total domination in graphs*, Networks 10 (1980), pp. 211–219.

[3] M. Conforti, G. Cornuéjols and K. Vušković, *Balanced matrices*, Discrete Mathematics 306 (2006), pp. 2411–2437.

[4] R. Faudree, E. Flandrin and Z. Ryjàcek, *Claw-free graphs – A survey*, Discrete Mathematics 164 (1997), pp. 87–147.

[5] T.W. Haynes, S.T. Hedetniemi, P.J. Slater (Eds.), *Fundamentals of Domination in Graphs*, Marcel Dekker, Inc., New York, 1998.

[6] M.A. Henning, *A survey of selected recent results on total domination in graphs*, Discrete Mathematics 309, (2009), pp. 32–63.

[7] V.V. Lozin, *On maximum induced matchings in bipartite graphs*, Information Processing Letters 81 (2002), pp. 7–11.

[8] O. Schaudt, *On weighted efficient total domination*, submitted preprint.

[9] O. Schaudt, *On efficient total domination*, submitted preprint.

Progress on rainbow connection

Ingo Schiermeyer

Institut für Diskrete Mathematik und Algebra, Technische Universität Bergakademie Freiberg, 09596 Freiberg, Germany,
Ingo.Schiermeyer@tu-freiberg.de

Key words: Rainbow colouring, rainbow connectivity, extremal problem

1 Introduction

We use [1] for terminology and notation not defined here and consider finite and simple graphs only.

An edge-coloured graph G is called *rainbow-connected* if any two vertices are connected by a path whose edges have different colours. This concept of rainbow connection in graphs was recently introduced by Chartrand et al. in [4]. The rainbow connection number of a connected graph G, denoted $rc(G)$, is the smallest number of colours that are needed in order to make G rainbow connected. An easy observation is that if G has n vertices then $rc(G) \leq n-1$, since one may colour the edges of a given spanning tree of G with different colours, and colour the remaining edges with one of the already used colours. Chartrand et al. computed the precise rainbow connection number of several graph classes including complete multipartite graphs [4]. The rainbow connection number has been studied for further graph classes in [3] and for graphs with fixed minimum degree in ([3], [7], [9]).

Rainbow connection has an interesting application for the secure transfer of classified information between agencies (cf. [5]). While the information needs to be protected since it relates to national security, there must also be procedures that permit access between appropriate parties. This two-fold issue can be addressed by assigning information transfer paths between agencies which may have other agencies as intermediaries while requiring a large enough number of passwords and firewalls that is prohibitive to intruders, yet small enough to manage (that is, enough so that one or more paths between every pair of agencies have no password repeated). An immediate question arises: What is the minimum number of passwords or firewalls needed that allows one or

more secure paths between every two agencies so that the passwords along each path are distinct?

The computational complexity of rainbow connectivity has been studied in ([2], [8]). It is proved that the computation of $rc(G)$ is NP-hard ([2],[8]). In fact it is already NP-complete to decide if $rc(G) = 2$, and in fact it is already NP-complete to decide whether a given edge-coloured (with an unbounded number of colours) graph is rainbow connected [2]. More generally it has been shown in [8], that for any fixed $k \geq 2$, deciding if $rc(G) = k$ is NP-complete.

For the rainbow connection numbers of graphs the following results are known (and obvious).

Proposition 1
Let G be a connected graph of order n. Then
1. $1 \leq rc(G) \leq n - 1$,
2. $rc(G) \geq diam(G)$,
3. $rc(G) = 1 \Leftrightarrow G$ *is complete*,
4. $rc(G) = n - 1 \Leftrightarrow G$ *is a tree.*

2 Rainbow connection and minimum degree

Motivated by the fact that there are graphs with minimum degree 2 and with $rc(G) = n - 3$ (just take two vertex-disjoint triangles and connect them by a path of length $n - 5$), and by the fact that cliques have $rc(G) = 1$, it is interesting to study the behaviour of $rc(G)$ with respect to the minimum degree $\delta(G)$. In [3] Caro et al. have shown the following theorem.

Theorem 1 *If G is a connected graph with n vertices and $\delta(G) \geq 3$ then* $rc(G) < \frac{5n}{6}$.

They also made the following conjecture.

Conjecture 1 *If G is a connected graph with n vertices and $\delta(G) \geq 3$ then* $rc(G) < \frac{3n}{4}$.

For 2-connected graphs Conjecture 1 is true. This follows from the following proposition in [3].

Proposition 2 *If G is a 2-connected graph with n vertices then* $rc(G) \leq \frac{2n}{3}$.

Corollary 2 *If G is a 2-connected graph with n vertices then* $rc(G) \leq \frac{3n-1}{4}$

for $n \geq 3$.

Conjecture 1 has recently been proven in [9] by the following theorem.

Theorem 3 *If G is a connected graph with n vertices and $\delta(G) \geq 3$ then $rc(G) \leq \frac{3n-1}{4}$.*

The presented results motivate the following challenging problem.

Problem 1 *For every $k \geq 2$ find a minimal constant c_k with $0 < c_k \leq 1$ such that $rc(G) \leq c_k \cdot n$ for all graphs G with minimum degree $\delta(G) \geq k$. Is it true that $c_k = \frac{3}{k+1}$ for all $k \geq 2$?*

This is true for $k = 2, 3$ as shown before ($c_2 = 1$ and $c_3 = \frac{3}{4}$).

3 Rainbow connection and size of graphs

Another approach for achieving upper bounds is based on the size (number of edges) of the graph. Those type of sufficient conditions are known as Erdős-Gallai type results. Research on the following Erdős-Gallai type problem has been started in [6].

Problem 2 *For every $k, 1 \leq k \leq n-1$, compute and minimize the function $f(n,k)$ with the following property: If $|E(G)| \geq f(n,k)$, then $rc(G) \leq k$.*

First we can show a lower bound for $f(n,k)$.

Proposition 3
$f(n,k) \geq \binom{n-k+1}{2} + (k-1)$.

Proof: We construct a graph G_k as follows: Take a $K_{n-k+1} - e$ and denote the two vertices of degree $n-k-1$ with u_1 and u_2. Now take a path P_k with vertices labeled $w_1, w_2, \ldots, w_k$ and identify the vertices u_2 and w_1. The resulting graph G_k has order n and size $|E(G)| = \binom{n-k+1}{2} + (k-2)$. For its diameter we obtain $d(u_1, w_k) = diam(G) = k+1$. Hence $f(n,k) \geq \binom{n-k+1}{2} + (k-1)$. □

Using Propositions 2 and 3 we can compute $f(n,k)$ for $k \in \{1, n-2, n-1\}$.

Proposition 4
$f(n,1) = \binom{n}{2}$,
$f(n,n-1) = n-1$,
$f(n,n-2) = n$.

For $k = 2$ we obtain $f(n, 2) = \binom{n-1}{2} + 1$ by the following stronger result shown in [6].

Theorem 4 *Let G be a connected graph of order n and size m. If $\binom{n-1}{2} + 1 \leq m \leq \binom{n}{2} - 1$, then $rc(G) = 2$.*

References

[1] J. A. Bondy and U.S.R. Murty, *Graph Theory*, Springer, 2008.

[2] S. Chakraborty, E. Fischer, A. Matsliah, and R. Yuster, *Hardness and algorithms for rainbow connectivity*, Proceedings STACS 2009, to appear in Journal of Combinatorial Optimization.

[3] Y. Caro, A. Lev, Y. Roditty, Z. Tuza, and R. Yuster *On rainbow connection*, The Electronic Journal of Combinatorics 15 (2008), #57.

[4] G. Chartrand, G. L. Johns, K. A. McKeon, and P. Zhang, *Rainbow connection in graphs*, Math. Bohemica 133 (2008) 85-98.

[5] A. B. Ericksen, *A matter of security*, Graduating Engineer & Computer Careers (2007) 24-28.

[6] A. Kemnitz and I. Schiermeyer, *Graphs with rainbow connection number two*, preprint 2009, submitted to Discussiones Mathematicae Graph Theory.

[7] M. Krivelevich and R. Yuster, *The rainbow connection of a graph is (at most) reciprocal to its minimum degree*, Preprint 2009.

[8] V. B. Le and Z. Tuza, *Finding optimal rainbow connection is hard*, Preprint 2009.

[9] I. Schiermeyer, *Rainbow connection in graphs with minimum degree three*, IWOCA 2009, LNCS 5874 (2009) 432-437.

An Optimal Algorithm for the Indirect Covering Subtree Problem

Joachim Spoerhase

Lehrstuhl für Informatik I, Universität Würzburg, Am Hubland, 97074 Würzburg, Germany

Key words: Graph algorithm, coverage, medianoid, tree, efficient algorithm

---

Introduction Suppose that a company wants to open a number r of facilities in a given network, modeled by an edge-weighted graph. A potential customer u, located at some node of the graph, is willing to use the service provided by that company if the cost thereby incurred is limited by some bound $\varrho(u)$. The costs are modeled by shortest-path distances in the underlying graph, that is, the cost for customer u equals the distance to the closest server of the company. For example, the distances may represent transportation costs or travel times in a logistical network, but also response times in a communication network. If the company serves customer u—that is, his distance to the closest server does not exceed $\varrho(u)$—it earns a profit of $w(u)$, which generally corresponds to the demand of the customer. The goal of the company is to identify r locations (nodes of the graph) for its facilities such that the total profit is maximized.

The resulting optimization problem, called *maximum coverage location* [MZH83], is NP-hard for arbitrary graphs. It can, however, be solved in time $O(rn^2)$ on *trees* [Tam96]. As trees form the sparsest (and thus cheapest) networks connecting a given set of nodes they play an important role in many areas of application such as logistical and communication networks. For example, we may think of backbones of a computer network, which are often tree-shaped. This is the case that we investigate in this paper.

In the past years there has been an increasing interest in location problems combined with *connectivity* requirements (for example, connected dominating set, or connected facility location). We follow this line of research and consider a variant of maximum coverage location on a tree where the facilities are

Email address: joachim.spoerhase@uni-wuerzburg.de (Joachim Spoerhase).

required to form a *connected* subgraph, that is, a subtree of the input tree. Albeit consisting of a plurality of nodes, such a subtree is considered as a *single*, *tree-shaped* facility. The cost of this facility is given by its total edge weight. (The number of nodes within the facility is not bounded.) The goal is to identify a tree-shaped facility that maximizes the net profit, that is, the income produced by the customers minus the setup cost of the facility. As possible application we may think of a company that wants to establish a *high-bandwidth core* (tree-shaped facility) in a given communication network in order to provide some service to potential customers. A customer is only willing to pay for the service if the response time is sufficiently low.

Kim et al. [KLTW96] introduce the *indirect covering subtree problem*, which can be used to model the above scenario. But instead of assigning to each customer a *profit* $w(u)$ they assume that a *penalty* $\pi(u)$ is imposed on the company if customer u is *not* served. The company aims at minimizing the sum of setup cost and the total penalty cost. It should be clear that both formulations are equivalent.

Problem Definition The input of the indirect covering subtree problem is an undirected tree $T = (V, E)$ with non-negative edge weights $c\colon E \to \mathbb{R}_{\geq 0}$. The edge weights induce a distance function $d\colon V \times V \to \mathbb{R}_{\geq 0}$ on the node set. Each node is associated with a radius $\varrho(u)$ and a non-negative penalty $\pi(u)$. Consider a subtree Y of T. A node u is said to be *covered* (indirectly) by Y if $d(u, Y) \leq \varrho(u)$, that is, if u lies within distance $\varrho(u)$ from Y. If u is *not* covered by Y, then u imposes a penalty $\pi(u)$ on Y. If $U \subseteq V$ is a set of nodes then $p(U, Y) := \sum\{\, \pi(u) \mid u \in U \text{ and } d(u, Y) > \varrho(u) \,\}$ denotes the penalty imposed on Y by U. The total penalty imposed on Y is given by $p(Y) := p(V, Y)$. The indirect covering subtree problem asks for a subtree Y of T such that the total cost $c(Y) + p(Y)$, given by the sum of setup and penalty cost, is minimum among all subtrees of T.

If we require that Y be a node rather than a subtree we obtain the *single maximum coverage location problem* [MZH83,SW09a]. It is not hard to see that single maximum coverage location is a special case of indirect covering subtree. (Scale all edge lengths and radii with a sufficiently large factor while leaving the penalties unchanged.)

Previous Results The maximum coverage location problem (allowing the placement of an arbitrary set of r nodes) is NP-hard on general graphs [MZH83] while it can be solved in time $O(rn^2)$ on trees [Tam96]. This leads to an $O(n^2)$ algorithm for the *single* maximum coverage location problem on trees by setting $r = 1$. Kim et al. [KLTW96] provide a faster algorithm running

in $O(n \log^2 n)$. Their algorithm works even for the more general indirect covering subtree problem. Recently a slightly faster $O(n \log^2 n / \log\log n)$-time algorithm for single maximum coverage location has been reported [SW09a].

Our Contribution We propose an $O(n \log n)$ algorithm for indirect covering subtree, which is faster than the previously best algorithms for that problem and single maximum coverage location on trees. We complement this result with a matching lower bound on the running showing that our algorithm is optimal.

Our result also implies faster algorithms for competitive location problems [Hak83]. Specifically, we obtain an $O(n \log n)$ algorithm for $(1, X)$-medianoid and $O(n^2 \log n \log w(T))$ and $O(n^2 \log n \log w(T) \log D)$ algorithms for the discrete and absolute $(1, p)$-centroid problems on trees, respectively. Here, $w(T)$ denotes the total node weight of the tree and D is the maximum edge weight. The previously best algorithms are slower by factor of $O(\log n / \log\log n)$ [SW09c].

Sketch of the Algorithm We employ the algorithmic framework used by Kim et al. [KLTW96]. We improve, however, one of their core routines by using a more sophisticated technique to subdivide trees. This technique, which we call *two-terminal subtree subdivision*, is a simplification of the recursive coarsening strategy [SW09a] used for solving single maximum coverage location on a tree. The source of our speedup is that we manage to avoid explicitly sorting the nodes according to their distances and radii during the recursion, which has been necessary in the coarsening approach and also in the original algorithm of Kim et al. A further advantage of our algorithm is that it is a lot simpler than the recursive coarsening algorithm.

The two-terminal subtree technique has proved successful also for other location problems [SW10,SW09b]. We believe that there are further problem classes where it can be applied.

Our algorithm is based on the so-called *bitree model* [KLTW96]. A bitree is a directed graph T' that can be derived from an undirected tree T by replacing any edge (u, v) of T with a pair of anti-parallel arcs (u, v) and (v, u). With each arc (u, v) of such a bitree T' we associate a cost $c_{T'}(u, v)$. But in contrast to the edge costs in the indirect covering subtree problem we allow these arc costs to be negative and asymmetric. This induces a distance function $d_{T'} \colon V \times V \to \mathbb{R}$ where $d_{T'}(u, v)$ is the length of the unique u-v-path in T'. If v is a node and U a set of nodes of T' we define $p'(U, v) := \sum\{\, \pi(u) \mid u \in U \text{ and } d_{T'}(u, v) > \varrho(u) \,\}$ and $p'(v) := p'(V, v)$ similar to the undirected case.

Kim et al. reduce the indirect covering subtree problem to the computation of $p'(v)$ for all nodes of a given bitree. Specifically, they show that if $p'(v)$ can be computed in time $O(h(n))$ for all nodes v of an arbitrary bitree then the indirect covering subtree problem can be solved in the same asymptotic running time on undirected trees. They develop a routine to compute the $p'(\cdot)$-values of an bitree in total time $O(n \log^2 n)$. Thus they can also solve indirect covering subtree in $O(n \log^2 n)$.

We suggest an algorithm that computes all $p'(\cdot)$-values of an arbitrary bitree in $O(n \log n)$, which yields our main result.

We assume that any node of the given bitree T' has out-degree at most three. It is not hard to see, that this is no proper restriction. If s and t are distinct nodes then T'_{st} denotes the maximal sub-bitree of T' having s and t as leaves. We call s and t *terminals of* T'_{st} and T'_{st} *two-terminal sub-bitree* (TTSB) of T'.

Our algorithm divides the input bitree T' recursively into TTSBs. Since we are dealing with a degree-bounded bitree we can subdivide any TTSB T'_{st} into five (or fewer) TTSBs, called *child TTSBs of* T'_{st}, which have bounded size.

Lemma 1 *Let T'_{st} be a TTSB with maximum out-degree three. Then T'_{st} can be partitioned into five (or fewer) arc-disjoint TTSBs each of which having at most $\frac{1}{2}|T'_{st}| + 1$ nodes. This subdivision can be computed in $O(|T'_{st}|)$ time.* □

Consider a TTSB T'_{st}. We introduce the lists $L_{d,s}(T'_{st})$ and $L_{\varrho,s}(T'_{st})$. Both lists contain all nodes v of T'_{st} sorted in increasing order with respect to the values $d_{T'}(s, v)$ and $\varrho(v) - d_{T'}(v, s)$, respectively. The lists $L_{d,t}(T'_{st})$ and $L_{\varrho,t}(T'_{st})$ are defined symmetrically.

Our algorithm computes $p'(v, T'_{st})$ for all $v \in T'_{st}$ as well as the four lists $L_{d,s}(T'_{st})$, $L_{d,t}(T'_{st})$, $L_{\varrho,s}(T'_{st})$ and $L_{\varrho,t}(T'_{st})$ for any TTSB T'_{st} occurring during the recursion. We show that this information can be propagated inductively from child towards parent TTSBs such that we will have computed $p'(\cdot, T') = p'(\cdot)$ at the top of the recursion. One such propagation step takes time linear in the size $|T'_{st}|$ of the current TTSB T'_{st}. Together with Lemma 1 we infer.

Theorem 2 *The indirect covering subtree problem and hence also single maximum coverage location on a tree can be solved in time $O(n \log n)$.* □

We complement our algorithm with a lower bound $\Omega(n \log n)$ on the running time for solving single maximum coverage location on a tree. To this end we introduce a variant of the set disjointness problem, which needs $\Omega(n \log n)$ time to be recognized on certain real-number RAMs [BAG01]. Finally, we provide a linear time reduction from this problem to single maximum coverage location on a tree.

References

[BAG01] A. M. Ben-Amram and Z. Galil. Topological lower bounds on algebraic random access machines. *SIAM Journal on Computing*, 31(3):722–761, 2001.

[Hak83] S. L. Hakimi. On locating new facilities in a competitive environment. *European Journal of Operational Research*, 12:29–35, 1983.

[KLTW96] T. U. Kim, T. J. Lowe, A. Tamir, and J. E. Ward. On the location of a tree-shaped facility. *Networks*, 28(3):167–175, 1996.

[MZH83] N. Megiddo, E. Zemel, and S. Hakimi. The maximum coverage location problem. *SIAM Journal on Algebraic and Discrete Methods*, 4(2):253–261, 1983.

[SW09a] J. Spoerhase and H.-C. Wirth. An $O(n\,(\log n)^2/\log\log n)$ algorithm for the single maximum coverage location or the $(1, X_p)$-medianoid problem on trees. *Information Processing Letters*, 109(8):391–394, 2009.

[SW09b] J. Spoerhase and H.-C. Wirth. Optimally computing all solutions of Stackelberg with parametric prices and of general monotonous gain functions on a tree. *Journal of Discrete Algorithms*, 7(2):256–266, 2009.

[SW09c] J. Spoerhase and H.-C. Wirth. (r, p)-centroid problems on paths and trees. *Theoretical Computer Science*, 410(47–49):5128–5137, 2009.

[SW10] J. Spoerhase and H.-C. Wirth. Relaxed voting and competitive location under monotonous gain functions on trees. *Discrete Applied Mathematics*, 158:361–373, 2010.

[Tam96] A. Tamir. An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19:59–64, 1996.

Radio Labeling Cartesian Graph Products

Cynthia Wyels [a]

[a] *California State University Channel Islands*

Maggy Tomova [b]

[b] *University of Iowa*

Key words: radio number, radio labeling, Cartesian product

1 Introduction

Radio labeling is derived from the assignment of radio frequencies (channels) to a set of transmitters. The frequencies assigned depend on the geographical distance between the transmitters: the closer two transmitters are, the greater the potential for interference between their signals. Thus when the distance between two transmitters is small, the difference in the frequencies assigned must be relatively large, whereas two transmitters at a large distance may be assigned frequencies with a small difference.

The use of graphs to model the "channel assignment" problem was first proposed by Hale in 1980 [5]. Several schemes for distance labeling were subsequently introduced and have been extensively studied; Chartrand et al introduced the variation known as radio labeling in 2001 [2].

In the graph model of the channel assignment problem, the vertices correspond to the transmitters, and graph distance plays the role of geographical distance. We assume all graphs are connected and simple. The *distance* between two vertices u and v of a graph G, $d(u, v)$, is the length of a shortest path between u and v. The *diameter* of G, $diam(G)$, is the maximum distance, taken over all pairs of vertices of G. A *radio labeling* of a graph G is then defined to be a function $c : V(G) \to \mathbb{Z}_+$ satisfying

$$d(u, v) + |c(u) - c(v)| \geq 1 + diam(G) \tag{1}$$

for all distinct pairs of vertices $u, v \in V(G)$. The *span* of a radio labeling c is the maximum integer assigned by c. The *radio number* of a graph G, $rn(G)$,

is the minumum span, taken over all radio labelings of G [1].

As Liu and Zhu write, "It is surprising that determining the radio number seems a difficult problem even for some basic families of graphs." [9] The radio number is known exactly for only a few graph families, including paths and cycles [9] and the squares of paths [8] and cycles [7]; wheels and gears [3], some generalized prisms [11], and Cartesian products of a cycle with itself [10]. Meanwhile, bounds for the radio numbers of trees [6], ladders [4], and square grids [1] have been identified, while the radio number of cubes of the cycles C_n^3 for $n \leq 20$ and $n \equiv 0$, 2, or 4 $\pmod 6$ is known [12].

In this investigation we focus as follows:

Question: *What may be said about the radio number of the Cartesian product of two graphs?*

The Cartesian product of two graphs G and H has vertex set $V(G\Box H) = V(G) \times V(H) = \{(g,h) \,|\, g \in V(G) \text{ and } h \in V(H)\}$. The edges of $G\Box H$ consist of those pairs of vertices $\{(g,h), (g',h')\}$ satisfying $g = g'$ and h is adjacent to h' in H or $h = h'$ and g is adjacent to g' in G. We note the following facts about Cartesian products:

- The order of a product is the product of the orders of the factor graphs, i.e., $G\Box H$ has $|V(G)| \cdot |V(H)|$ vertices.
- Distances in products are sums of distances between corresponding vertices in factor graphs, i.e., $d_{G\Box H}((g_1,h_1),(g_2,h_2)) = d_G(g_1,g_2) + d_H(h_1,h_2)$.
- In particular, the diameter of a product is the sum of the diameters of the factors, i.e., $diam(G\Box H) = diam(G) + diam(H)$.

In Section 2 we provide three lower bounds for the radio number of a Cartesian product, each of which outperforms the others in specific cases. Two upper bounds are provided in Section 3, along with some comments as to their efficacy.

2 Lower Bounds

Our first bound follows directly from the fact that a radio labeling is an injection. As such, the span of any radio labeling may never be less than the number of vertices of the associated graph.

[1] We use the convention, established in [2], that the co-domain of a radio labeling is $\mathbb{Z}_+ = \{1,2,\ldots\}$. Some authors use $\{0,1,2,\ldots\}$ as the co-domain; radio numbers specified using the non-negative integers as co-domain are one less than those determined using the positive integers.

Theorem 1 *(Vertex Lower Bound)*

$$rn(G\Box H) \geq |V(G)| \cdot |V(H)|.$$

This lower bound is tight for the product of the Petersen graph with itself.

The second lower bound is stated in terms of the radio numbers of the factor graphs.

Theorem 2 *(Radio Number Lower Bound)Let G and H be graphs.*

$$rn(G\Box H) \geq rn(G) + rn(H) - 1.$$

This bound outperforms the Vertex Lower Bound on prism graphs, which are products of 2-paths with n-cycles. These prism graphs have $2n$ vertices (so the Vertex Lower Bound states the radio number is not less than $2n$); the Radio Lower Bound gives a lower bound that is $O(n^2)$.

To specify the third lower bound, an additional term, the "gap," must be introduced. Essentially, the gap of G is the smallest possible difference between the ith and $(i+2)$nd largest labels in a radio labeling of G.

Definition 3 *Let c be a radio labeling of a graph G, and let $\{x_1, x_2, \ldots, x_n\}$ be the vertices of G, arranged so that $c(x_i) < c(x_j)$ whenever $i < j$. Define $\phi(G, c)$ to be the smallest integer satisfying $\phi(G, c) \geq c(x_{i+2}) - c(x_i)$ for all $i \in \{1, 2, \ldots, n-2\}$. Finally, define $\phi(G)$ to be the minimum of $\phi(G, c)$ taken over all radio labelings c of G.*

Our third lower bound is expressed in terms of this gap.

Theorem 4 *(Gap Lower Bound)*

$$rn(G\Box H) \geq \left(\left\lfloor \frac{1}{2}|V(G)| \cdot |V(H)| \right\rfloor - 1\right)(\phi(G) + \phi(H) - 2) + a,$$

where $a = 1$ when $|V(G)| \cdot |V(H)|$ is odd and $a = 2$ otherwise.

This lower bound is again sharp for the product of the Petersen graph with itself. Moreover, it is significantly more effective than the Vertex and Radio Lower Bounds on products of cycles with themselves and products of paths with themselves. The Vertex and Radio Lower Bounds give lower bounds that are $O(n^2)$ for each; the Gap Lower Bound gives $O(n^3)$ bounds. (We note that the radio number of both families of products has been shown to be $O(n^3)$ [1], [10].)

3 Upper Bounds

In this section we turn our attention to determining upper bounds for radio numbers of Cartesian products of graphs. The radio condition (1) depends on distance and diameter; without knowledge of specific distances between pairs of vertices in the product graph, it is unreasonable to expect upper bounds to be sharp. Nonetheless, we present two upper bounds with varied hypotheses.

Theorem 5 *Let G be a graph with diameter 2 and $rn(G) = |V(G)| = n$. Then*

$$rn(G\Box G) \leq n^2 + 2(2n-2) + \left(2\left\lfloor \frac{n}{2} \right\rfloor - 1\right)\left\lfloor \frac{n-1}{2} \right\rfloor.$$

The statement of the next theorem is similar in spirit.

Theorem 6 *Assume G and H are graphs satisfying $rn(G) = |V(G)| = n$, $rn(H) = |V(H)| = m$, and $diam(G) - diam(H) \geq 2$. Then*

$$rn(G\Box H) \leq diam(G)(n+m-2) + 2mn - 2m - 4n + a,$$

where $a = 7$ when m and n have opposite parity, $a = 8$ when both m and n are odd, and $a = 6$ when both m and n are even.

To establish both of these theorems, one must replace the radio condition (1) with conditions sufficient for labelings of the products to be radio labelings. As nothing is assumed regarding distances between vertices, the bounds necessarily involve products of the diameters of the factor graphs. It would be of interest to investigate additional upper bounds resulting from removing the hypotheses that the factor graphs have the smallest possible radio numbers, or by including additional hypotheses regarding structure of the factor graphs.

References

[1] Calles, L., Gomez, H., Tomova, M., and Wyels, C., Bounds for the radio number of square grids, in preparation.

[2] Chartrand, G., Erwin, D., Zhang, P. and Harary, F., Radio labelings of graphs, Bull. Inst. Combin. Appl., **33** (2001), 77–85.

[3] Fernandez, C., Flores, A., Tomova, M., and Wyels, C., The radio number of gear graphs, preprint available at front.math.ucdavis.edu/0809.2623.

[4] Flores, J., Lewis, K., Tomova, M., and Wyels, C., The radio number of ladder graphs, in preparation.

[5] Hale, W.K.,Frequency assignment: theory and application, Proc. IEEE, **68** (1980), 1497–1514.

[6] Liu, D.D.-F., Radio number for trees, Discrete Math. **308** (2008), no. 7, 1153–1164.

[7] Liu, D.D.-F., and Xie, M., Radio number for square cycles, Congr. Numer. **169** (2004), 105125.

[8] Liu, D.D.-F., and Xie, M., Radio number for square paths, preprint available at www.calstatela.edu/faculty/dliu/ArsComFinal.pdf.

[9] Liu, D.D.-F., Zhu, X., Multilevel distance labelings for paths and cycles, SIAM J. Discrete Math. **19** (2005), No. 3, 610–621.

[10] Morris-Rivera, M., Tomova, M., Wyels, C., and Yeager, A., The Radio Number of $C_n \Box C_n$, preprint available at faculty.csuci.edu/cynthia.wyels/REU/aids.htm.

[11] Ortiz, J.P., Martinez, P., Tomova, M., and Wyels, C., Radio numbers of some generalized prism graphs, preprint available at faculty.csuci.edu/cynthia.wyels/REU/aids.htm.

[12] Sooryanarayana, B., and Raghunath, P., Radio labeling of cube of a cycle, Far East J. Appl. Math. **29** (2007), no. 1, 113–147.

Cycle embedding in alternating group graphs with faulty vertices and faulty edges

Ping-Ying Tsai *

Institute of Mathematics, Academia Sinica, Taipei 10617, Taiwan, ROC

Key words: Alternating group graph, Pancyclicity, Fault-tolerant, Cayley graph, Cycle embedding, Interconnection network.

1 Introduction

Let G be a graph with the vertex set $V(G)$ and the edge set $E(G)$. Unless otherwise stated, we follow [2] for graph terminologies and notations. A path $P_{v_0,v_k} = \langle v_0, v_1, \ldots, v_k \rangle$ is a sequence of distinct vertices except possibly $v_0 = v_k$ such that every two consecutive vertices are adjacent. The *length* of a path is the number of edges on the path. The *distance* between u and v is denoted by $d(u, v)$, which is the length of a shortest path between u and v. A *cycle* is a special path with at least three vertices such that the first vertex is the same as the last one. A cycle of length l is referred to as an *l-cycle*.

An interconnection network is usually modeled as an undirected simple graph, where the vertices represent processors and the edges represent communication links between processors. Study of the topological properties of an interconnection network is an important part of the study of any parallel or distributed system. The alternating group graph [8], which is an instance of Cayley graphs, is suitable to serve as a network because of its scalability and other favorable properties, e.g., regularity, recursiveness, symmetry, sublogarithmic degree and diameter, and maximal fault tolerance.

Let $u = a_1 a_2 \cdots a_n$ be a permutation of 1, 2, ..., n, i.e., $a_i \in \{1, 2, \ldots, n\}$ and $a_i \neq a_j$ for $i \neq j$. A pair of symbols a_i and a_j of u are said to be an inversion if $a_i < a_j$ whenever $i > j$. An even permutation is a permutation that contains an even number of inversions. Let A_n denote the set of all even permutations

* corresponding author
Email address: bytsai@math.sinica.edu.tw (Ping-Ying Tsai).

over $\{1, 2, \ldots, n\}$. For $3 \leq i \leq n$, we define two operations, g_i^+ and g_i^-, on A_n by setting ug_i^+ (respectively, ug_i^-) to be the permutation obtained from u by rotating the symbols a_1, a_2, a_i from left to right (respectively, from right to left), while retaining the other $n-3$ symbols stationary. For example, we have $12345g_4^+ = 41325$ and $12345g_4^- = 24315$. The n-dimensional alternating group graph AG_n, has the vertex set $V(AG_n) = A_n$ and the edge set $E(AG_n) = \{(u, v)|u, v \in V(AG_n)$ and $v = ug_i^+$ or $v = ug_i^-$ for some $3 \leq i \leq n\}$. It is not difficult to see that AG_n is regular with vertex degree $2(n-2)$, $|V(AG_n)| = n!/2$, and $|E(AG_n)| = (n-2)n!/2$. In addition, AG_n is both vertex symmetric and edge symmetric [8].

For $n \geq 3$ and $1 \leq i \leq n$, let $A_n^{(i)}$ be the subset of A_n that consists of all even permutations with element i in the rightmost position, and let $AG_n^{(i)}$ be the subgraph of AG_n induced by $A_n^{(i)}$. Obviously, $AG_n^{(i)}$ is isomorphic to AG_{n-1} for every $i \in \{1, 2, \ldots, n\}$. Due to the hierarchical structure, AG_n can also be defined recursively as follows. AG_n is constructed from n disjoint copies of $(n-1)$-dimensional alternating group graphs $AG_n^{(i)}$ for $i \in \{1, 2, \ldots, n\}$ such that $AG_n^{(i)}$ and $AG_n^{(j)}$, $i \neq j$, are connected by $(n-2)!$ edges, called *external edges*, of the form $(kj \cdots i, ik \cdots j)$ or $(jk \cdots i, ki \cdots j)$ for $k \in \{1, 2, \ldots, n\} \setminus \{i, j\}$. By contrast, edges joining vertices in the same subgraph $AG_n^{(i)}$ are called *internal edges*. In particular, for each internal edge (u, v) with $u = kj \cdots i$ and $v = jk' \cdots i$ in $AG_n^{(i)}$, there exist two adjacent vertices $s = ik \cdots j$ and $t = k'i \cdots j$ in $AG_n^{(j)}$ such that $s = ug_n^+$, $t = vg_n^-$, and $\langle u, s, t, v, u\rangle$ forms a 4-cycle in AG_n. For convenience, such a property is called the 4-*cycle structure* of (u, v). Note that the pair of vertices s and t is uniquely determined by the 4-cycle structure of (u, v). As a result, every vertex $u \in V(AG_n^{(i)})$ is connected to exactly 2 external edges and $2n-6$ internal edges.

A path (or cycle) in G is called a *Hamiltonian path* (or *Hamiltonian cycle*) if it contains every vertex of G exactly once. A graph G is called *Hamiltonian* if it has a Hamiltonian cycle. G is called *Hamiltonian-connected* if every two vertices of G are connected by a Hamiltonian path. For an integer $r \geq 3$, G is called *r-pancyclic* if it contains an l-cycle for each l with $r \leq l \leq |V(G)|$. In particular, G is called *vertex r-pancyclic* (or *edge r-pancyclic*) if every vertex (or edge) of G belongs to an l-cycle for each l with $r \leq l \leq |V(G)|$. A 3-pancyclic graph, a vertex 3-pancyclic graph, and an edge 3-pancyclic graph are called *pancyclic*, *vertex-pancyclic*, and *edge-pancyclic*, respectively. Exploring the pancyclicity of the given graph has attracted a lot of mathematicians [1,9,10]. Recently, some researchers have focused on the problem on interconnection networks because networks with cycle topology are suitable for designing simple algorithms with low communication costs (for example, [5,6,13,16]).

A graph G is *panconnected* if, for any two distinct vertices $u, v \in V(G)$ and for each integer l with $d(u, v) \leq l \leq |V(G)| - 1$, there is a $P_{u,v}$ of length l in G. It was shown in [4] that the alternating group graph is panconnected.

Lemma 1 ([4]) *For $n \geq 3$, AG_n is panconnected.*

Since faults may occur in networks, the consideration of fault-tolerance ability is a major factor in evaluating the performance of networks. Cycle embedding is also concerned extensively in many interconnection networks with faulty elements [7,11,14]. Suppose $F_v \subset V(G)$, $F_e \subset E(G)$, and $F = F_v \cup F_e$. A graph G is *k-vertex-fault-tolerant pancyclic* if $G - F_v$ remains pancyclic whenever $|F_v| \leq k$. A graph G is called *k-edge-fault-tolerant pancyclic* if $G - F_e$ is pancyclic whenever $|F_e| \leq k$. A graph G is called *k-fault-tolerant pancyclic* if $G - F$ remains pancyclic when $|F| \leq k$. The notion for *k-fault-tolerant Hamiltonian*, *k-fault-tolerant Hamiltonian-connected*, *k-fault-tolerant vertex r-pancyclic*, and *k-fault-tolerant edge r-pancyclic* can also be defined similarly.

Lemma 2 ([3]) *AG_n is $(n-4)$-vertex-fault-tolerant edge 4-pancyclic and $(n-3)$-vertex-fault-tolerant vertex-pancyclic, where $n \geq 4$.*

Lemma 3 ([15]) *AG_n is $(2n-6)$-vertex-fault-tolerant pancyclic, where $n \geq 3$.*

Lemma 4 ([12]) *AG_n is $(2n-6)$-edge-fault-tolerant pancyclic, where $n \geq 3$.*

Lemma 5 ([12]) *AG_n is $(2n-7)$-fault-tolerant Hamiltonian-connected, where $n \geq 4$.*

2 Main results

We improve previous results in Lemma 2, Lemma 3, and Lemma 4, by showing that the n-dimensional alternating group graph AG_n is $(n-4)$-fault-tolerant edge 4-pancyclic, $(n-3)$-fault-tolerant vertex-pancyclic, and $(2n-6)$-fault-tolerant pancyclic, while considering both faulty vertices and faulty edges. All the results we achieved here are optimal with respect to the number of faulty elements tolerated.

Theorem 1 *AG_n is $(n-4)$-fault-tolerant edge 4-pancyclic, where $n \geq 4$.*

Theorem 2 *AG_n is $(n-3)$-fault-tolerant vertex-pancyclic, where $n \geq 3$.*

Theorem 3 *AG_n is $(2n-6)$-fault-tolerant pancyclic, where $n \geq 3$.*

Since a graph is Hamiltonian if it is pancyclic, we have the following Corollary.

Corollary 1 *AG_n is $(2n-6)$-fault-tolerant Hamiltonian, where $n \geq 3$.*

References

[1] J. A. Bondy. Pancyclic graphs. *Journal of Combinatorial Theory - Series B*, 11(1):80-84, 1971.

[2] J. A. Bondy and U. S. R. Murty. *Graph Theory,* volume 244 of *Graduate Texts in Mathematics*. Springer, Berlin, 2008.

[3] J. M. Chang and J. S. Yang. Fault-tolerant cycle-embedding in alternating group graphs. *Applied Mathematics and Computation*, 197(2):760-767, 2008.

[4] J. M. Chang, J. S. Yang, Y. L. Wang, and Y. Cheng. Panconnectivity, fault-tolerant Hamiltonicity and Hamiltonian-Connectivity in alternating group graphs. *Networks*, 44(4):302-310, 2004.

[5] A. Germa, M. C. Heydemann, and D. Sotteau. Cycles in the cube-connected cycles graph. *Discrete Applied Mathematics*, 83(1-3):135-155, 1998.

[6] S. Y. Hsieh and J. Y. Shiu. Cycle embedding of augmented cubes. *Applied Mathematics and Computation*, 191(2):314-319, 2007.

[7] S. Y. Hsieh and T. H. Shen. Edge-bipancyclicity of a hypercube with faulty vertices and edges. *Discrete Applied Mathematics*, 156(10):1802-1808, 2008.

[8] J. S. Jwo, S. Lakshmivarahan, and S. K. Dhall. A new class of interconnection networks based on the alternating group. *Networks*, 23:315-326, 1993.

[9] M. Kouider and A. Marczyk. On pancyclism in Hamiltonian graphs. *Discrete Mathematics*, 251(1-3):119-127, 2002.

[10] B. Randerath, I. Schiermeyer, M. Tewes, and L. Volkmann. Vertex pancyclic graphs. *Discrete Applied Mathematics*, 120(1-3):219-237, 2002.

[11] C. H. Tsai. Fault-tolerant cycles embedded in hypercubes with mixed link and node failures. *Applied Mathematics Letters*, 21(8):855-860, 2008.

[12] P. Y. Tsai, J. S. Fu, and G. H. Chen. Edge-fault-tolerant pancyclicity of alternating group graphs. *Networks*, 53(3):307-313, 2009.

[13] J. M. Xu and M. J. Ma. Cycles in folded hypercubes. *Applied Mathematics Letters*, 19(2):140-145, 2006.

[14] M. Xu, X. D. Hu, and Q. Zhu. Edge-bipancyclicity of star graphs under edge-fault tolerant. *Applied Mathematics and Computation*, 183(2):972-979, 2006.

[15] Z. J. Xue and S. Y. Liu. An optimal result on fault-tolerant cycle-embedding in alternating group graphs. *Information Processing Letters*, 109(21-22):1197-1201, 2009.

[16] X. F. Yang, G. M. Megson, and D. J. Evans. Locally twisted cubes are 4-pancyclic. *Applied Mathematics Letters*, 17(8):919-925, 2004.

On Reed's Conjecture in Triangle-Free Graphs

Vera Weil

Institut für Informatik, Arbeitsgruppe Faigle/Schrader (AFS), University of Cologne, Weyertal 80, 50931 Cologne, Germany

Key words: Coloring, Reed's Conjecture, Minimal Counterexample, Triangle-Free Graphs

1 Introduction

The problem of finding the chromatic number of a graph G, i.e. the minimum number of colors needed to assign distinct colors to adjacent nodes of G, is NP-complete. However, there are some known bounds including the trivial lower bound

$$\chi(G) \geq \omega(G)$$

and the upper bound provided by Brooks [2]

$$\chi(G) \leq \Delta(G) + 1,$$

where $\chi(G)$ denotes the chromatic number, $\Delta(G)$ the maximum degree and $\omega(G)$ the number of nodes of a largest clique in G.

In [6], Reed conjectured that

$$\chi(G) \leq \left\lceil \frac{\Delta(G) + 1 + \omega(G)}{2} \right\rceil.$$

Note that this upper bound rounds up the arithmetic average of the previous mentioned upper and lower bounds.

In the following all graphs are simple, undirected and connected. $V(G)$ and $E(G)$ denote the nodes and edges of G respectively. The degree of $v \in V(G)$ ($deg(v)$) gives the number of neighbors of v in G.

Email address: weil@zpr.uni-koeln.de (Vera Weil).

Reed [6] proved the existence of a constant Δ_0 so that the conjecture holds for graphs with $\Delta(G) \geq \Delta_0$ and $\omega(G) \geq \lfloor (1 - \frac{1}{70000000})\Delta(G) \rfloor$.

Other graph classes for which the validity of Reed's conjecture could be shown have been found, among them the classes listed in Proposition 1. This list consists of graph classes for which the conjecture follows straightforward from their definition or for which other authors (sources given in parantheses) have shown the conjecture holds.

Proposition 1 *For the following graphs the conjecture holds:*

- *complete graphs*
- *perfect graphs*
- *circles*
- *graphs with size of maximum independent set $\alpha(G) = 2$ ([5])*
- *line graphs ([4])*
- *triangle-free graphs with minimum degree $\delta(G) \geq \frac{n}{3}$ ([1])*
- *triangle-free graphs with chromatic number $\chi(G) \leq 4$ (consequence of Brooks' theorem)*

In this work we focus on triangle-free graphs ($\omega(G) = 2$), for which Reed's conjecture reads as follows:

$$\chi(G) - 1 \leq \left\lceil \frac{\Delta(G) + 1}{2} \right\rceil .$$

In order to investigate the conjecture we search for a triangle-free counterexample with a minimum number of nodes.

Gathering properties of a minimal order counterexample, we can continue a list already 'started' in [3]. Some of these properties turn out to be quite fruitful, in particular the fact that a minimal counterexample has to be vertex-color-critical, i.e. $\chi(G - v) < \chi(G)$ for any vertex v of G.

Definition 2 (heavy edge) *Let G be a graph. Then we call $(x, y) \in E(G)$ a heavy edge if $deg(x) = deg(y) = \Delta(G)$.*

Definition 3 (heavy circle) *Let G be a graph and $C \subseteq V(G)$ a circle of G. We call C a heavy circle if*

$$v \in V(C) \Rightarrow deg(v) \geq \Delta(G) - 2.$$

2 Results

Theorem 4 *Let G be a triangle-free minimal counterexample to Reed's conjecture. Then $\Delta(G) = 2\chi(G) - 5$.*

Theorem 5 *Let G be a triangle-free minimal counterexample to Reed's conjecture. Then there exists at least one heavy edge.*

Theorem 6 *Let G be a triangle-free minimal counterexample to Reed's conjecture. Then there exists at least one heavy circle C of odd length.*

Moreover, we describe a construction which embeds an arbitrary triangle-free graph G in a regular triangle-free graph preserving $\chi(G)$ and $\Delta(G)$. From this we get that it suffices to prove Reed's conjecture for regular triangle-free graphs.

Theorem 7 *If Reed's conjecture holds for all regular triangle-free graphs, it holds for all triangle-free graphs.*

References

[1] S. Brandt, S. Thomassé (2005): *Dense triangle-free graphs are four-colorable: A solution to the Erdös-Simonovits problem.* To appear in Journal of Combinatorial Theory B.
http://www.lirmm.fr/ thomasse/liste/vega11.pdf

[2] R.L. Brooks (1941): *On colouring the nodes of a network.* Proceedings of the Cambridge Philosophical Society 37, 194 - 197.

[3] D. Gernert, L. Rabern (2007): *A knowledge-based system for graph theory, demonstrated by partial proofs for graph coloring problems.* MATCH Commun. Math. Comput. Chem., Vol. 58, 445 - 460.

[4] A.D. King, B.A. Reed, A. Vetta (2007): *An upper bound for the chromatic number of line Graphs.* European Journal Of Combinatorics, Vol. 28(8), 2182-2187.

[5] M. Molloy (1991): *Chromatic neighbourhood sets.* Journal Of Graph Theory, Vol. 31, 303 - 311.

[6] B. Reed (1998): *ω,Δ and χ (OMEGA, DELTA AND CHI).* Journal Of Graph Theory, Vol. 27, 177 - 212.

Appendix

Matjaz Konvalinka, Igor Pak
Complexity of O'Hara's algorithm

Key words:

1 Introduction

Some combinatorial results have an easy proof via generating functions and a more elusive, but also more interesting and important, bijective proof. It would be difficult to think of a better example of this than the generalization of Euler's classical distinct/odd theorem due to George Andrews (Theorem 1). The proof via generating functions is a trivial one-line calculation. On the other hand, the simplest bijective proof of this result, O'Hara's algorithm, is distinctly non-trivial and has numerous fascinating properties.

Note that a quest to find bijective proofs of partition identities goes back all to way to the pioneer work of Sylvester and his school. Despite remarkable successes in the last century (see [P06]) and some recent work of both positive and negative nature (see e.g. [P04b,P]), the problem remains ambiguous and largely unresolved. Much of this stems from the lack of clarity as to what exactly constitutes a bijective proof. Depending on whether one accentuates simplicity, ability to generalize, the time complexity, geometric structure, or asymptotic stability, different answers tend to emerge.

In one direction, the subject of partition bijections was revolutionized by Garsia and Milne with their *involution principle* [GM81a,GM81b]. This is a combinatorial construction which allows to use a few basic bijections and involutions to build more involved combinatorial maps. As a consequence, one can start with a reasonable analytic proof of a partition identity and trace every step to obtain a (possibly extremely complicated) bijective construction. Garsia and Milne used this route to obtain a long sought bijection proving the Rogers-Ramanujan identities, resolving an old problem in this sense [GM81b]. Unfortunately, this bijection is too complex to be analyzed and has yet to lead to new Rogers-Ramanujan type partition identities.

After Garsia-Milne paper, there has been a flurry of activity to obtain synthetic bijections for large classes of partition identities. Most of these bijections did not seem to lead anywhere with one notable exception. Remmel and Gordon found (rather involved) bijective proofs of the above-mentioned partition identity due to Andrews [R82,G83]. O'Hara's streamlined proof is in fact a direct generalization of Glaisher's classical bijection proving Euler's theorem. Moreover, in her thesis [O84], O'Hara showed that her bijection is computationally efficient in certain special cases. Until now, the reason why O'Hara's bijection has a number of nice properties distinguishing it from the other "involution principle bijections" remained mysterious.

In this extended abstract, we present results of both positive and negative type. First, we analyze the complexity of O'Hara's bijection, which we view as a discrete algorithm. Theorem 3 gives an *exact* formula for the number of steps of the algorithm in certain cases. From here it follows that O'Hara's bijection is computationally efficient in many special cases. On the other hand, perhaps surprisingly, the number of steps can be (mildly) exponential in the worst case (Theorem 7 part (3)). This is the first negative result of this kind, proving the analogue of a conjecture that remains open for the Garsia-Milne's "Rogers-Ramanujan bijection" (see Subsection 4.1).

Second, we show that O'Hara's bijection has a rich underlying geometry. In a manner similar to that in [P04a,PV05], we view this bijection as a map between integer points in polytopes which preserves certain linear functionals. We present an advanced generalization of Andrews's result and of O'Hara's bijection in this geometric setting. In a special case, the working of the map corresponds to the Euclid algorithm and, more generally, to terms in the continuing fractions. Thus one can also think of our generalization as a version of multidimensional continuing fractions.

Finally, by combining the geometric and complexity ideas we see that in the finite dimensional case the map defined by O'Hara's bijection is a solution of an integer linear programming problem. This implies that the map defined by the bijection can be computed in polynomial time, i.e. much more efficiently than by O'Hara's bijection.

The extended abstract is structured as follows. We start with definitions and notations in Section 2. In Section 3, we describe the main results on both geometry and complexity. We conclude with final remarks in Section 4.

Due to space constraints, we present almost no proofs. An interested reader is invited to find the proofs and some other results in the paper [KP], on which this abstract is based.

2 Definitions and background

2.1 Andrews's theorem

A *partition* λ is an integer sequence $(\lambda_1, \lambda_2, \ldots, \lambda_\ell)$ such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_\ell > 0$, where the integers λ_i are called the *parts* of the partition. The sum $n = \sum_{i=1}^{\ell} \lambda_i$ is called the *size* of λ, denoted $|\lambda|$; in this case we say that λ is a partition of n, and write $\lambda \vdash n$. We can also write $\lambda = 1^{m_1} 2^{m_2} \cdots$, where $m_i = m_i(\lambda)$ is the number of parts of λ equal to i. The *support* of $\lambda = 1^{m_1} 2^{m_2} \cdots$ is the set $\{i \colon m_i > 0\}$. The set of all positive integers will be denoted by $\mathbb{P}$.

Denote the set of all partitions by $\mathcal{P}$ and the set of all partitions of n by $\mathcal{P}_n$. The number of partitions of n is given by Euler's formula

$$\sum_{\lambda \in \mathcal{P}} t^{|\lambda|} = \sum_{n=0}^{\infty} |\mathcal{P}_n| t^n = \prod_{i=1}^{\infty} \frac{1}{1-t^i}.$$

For a sequence $\overline{a} = (a_1, a_2, \ldots)$ with $a_i \in \mathbb{P} \cup \{\infty\}$, define $\mathcal{A}$ to be the set of partitions λ with $m_i(\lambda) < a_i$ for all i; write $\mathcal{A}_n = \mathcal{A} \cap \mathcal{P}_n$. Denote by $\mathrm{supp}(\overline{a}) = \{i \colon a_i < \infty\}$ the *support* of the sequence $\overline{a}$.

Let $\overline{a} = (a_1, a_2, \ldots)$ and $\overline{b} = (b_1, b_2, \ldots)$. We say that $\overline{a}$ and $\overline{b}$ are *φ-equivalent*, $\overline{a} \sim_\varphi \overline{b}$, if φ is a bijection $\mathrm{supp}(\overline{a}) \to \mathrm{supp}(\overline{b})$ such that $i a_i = \varphi(i) b_{\varphi(i)}$ for all i. If $\overline{a} \sim_\varphi \overline{b}$ for some φ, we say that $\overline{a}$ and $\overline{b}$ are equivalent, and write $\overline{a} \sim \overline{b}$.

Theorem 1 (Andrews) *If $\overline{a} \sim \overline{b}$, then $|\mathcal{A}_n| = |\mathcal{B}_n|$ for all n.*

Proof: We use the notation $t^\infty = 0$. Clearly,

$$\sum_{n=0}^{\infty} |\mathcal{A}_n| t^n = \prod_{i=1}^{\infty} \frac{1-t^{i a_i}}{1-t^i} = \prod_{j=1}^{\infty} \frac{1-t^{j b_j}}{1-t^j} = \sum_{n=0}^{\infty} |\mathcal{B}_n| t^n,$$

which means that $|\mathcal{A}_n| = |\mathcal{B}_n|$. $\square$

Consider the classical Euler's theorem on partitions into distinct and odd parts. For $\overline{a} = (2, 2, \ldots)$ and $\overline{b} = (\infty, 1, \infty, 1, \ldots)$, $\mathcal{A}_n$ is the set of all partitions of n into distinct parts, and $\mathcal{B}_n$ is the set of partitions of n into odd parts. The bijection $i \mapsto 2i$ between $\mathrm{supp}(\overline{a}) = \mathbb{P}$ and $\mathrm{supp}(\overline{b}) = 2\mathbb{P}$ satisfies $i a_i = \varphi(i) b_{\varphi(i)}$, so $\overline{a} \sim_\varphi \overline{b}$ and $|\mathcal{A}_n| = |\mathcal{B}_n|$. We refer to this example as the *distinct/odd case*.

The analytic proof of Andrews's theorem shown above does not give an explicit bijection $\mathcal{A}_n \to \mathcal{B}_n$. Such a bijection is, by Theorem 2, given by the following algorithm.

Algorithm 1 (O'Hara's algorithm on partitions)

`Fix:` sequences $\overline{a} \sim_\varphi \overline{b}$
`Input:` $\lambda \in \mathcal{A}$
`Set:` $\mu \leftarrow \lambda$
`While:` μ contains more than b_j copies of j for some j
  `Do:` remove b_j copies of j from μ, add a_i copies of i to μ, where $\varphi(i) = j$
`Output:` $\psi(\lambda) \leftarrow \mu$

Theorem 2 (O'Hara) *Algorithm 1 stops after a finite number of steps. The resulting partition $\psi(\lambda) \in \mathcal{B}$ is independent of the order of the parts removed and defines a size-preserving bijection $\mathcal{A} \to \mathcal{B}$.*

Denote by $L_\varphi(\lambda)$ the number of steps O'Hara's algorithm takes to compute $\psi(\lambda)$, and by $\mathcal{L}_\varphi(n)$ the maximum value of $L_\varphi(\lambda)$ over all $\lambda \vdash n$.

In the distinct/odd case, O'Hara's algorithm gives the inverse of Glaisher's bijection, which maps $\lambda = 1^{m_1}3^{m_3}\cdots \in \mathcal{B}$ to the partition $\mu \in \mathcal{A}$ which contains $i2^j$ if and only if m_i has a 1 in the j-th position when written in binary. $\square$

Let $\overline{a} = (1,1,4,5,3,1,1,\ldots)$, $\overline{b} = (1,1,5,3,4,1,1,\ldots)$ and $\varphi(3) = 4$, $\varphi(4) = 5$, $\varphi(5) = 3$, $\varphi(i) = i$ for $i \neq 3,4,5$; observe that $\overline{a} \sim_\varphi \overline{b}$. Then O'Hara's algorithm on $\lambda = 3^3 4^4 5^2$ runs as follows:

$$\mathbf{3^3 4^4 5^2} \to 3^7 4^1 5^2 \to 3^2 4^1 5^5 \to 3^2 4^6 5^1 \to 3^6 4^3 5^1$$
$$\to 3^{10} 4^0 5^1 \to 3^5 4^0 5^4 \to 3^0 4^0 5^7 \to 3^0 4^5 5^3 \to \mathbf{3^4 4^2 5^3}$$

We have $L_\varphi(\lambda) = \mathcal{L}_\varphi(35) = 9$. $\square$

Take $\overline{a} = (2,2,1,2,2,1,\ldots)$ and $\overline{b} = (3,1,3,1,\ldots)$. Here $\mathcal{A}$ is the set of partitions into distinct parts $\equiv \pm 1 \bmod 3$, and $\mathcal{B}$ is the set of partitions into odd parts, none appearing more than twice. Define $\varphi\colon \mathbb{P} \to \mathbb{P}$ as follows:

$$\varphi(i) = \begin{cases} i & \text{if } i \text{ is divisible by 6} \\ i/3 & \text{if } i \text{ is divisible by 3, but not by 2} \\ 2i & \text{if } i \text{ is not divisible by 3} \end{cases}. \tag{1}$$

Clearly, $\overline{a} \sim_\varphi \overline{b}$. O'Hara's algorithm on $1^1 2^1 8^1 10^1 14^1 20^1$ runs as follows:

$$\begin{aligned}
&\mathbf{1^1 2^1 8^1 10^1 14^1 20^1} \to 1^1 2^1 8^1 10^3 14^1 \to 1^1 2^1 7^2 8^1 10^3 \to 1^1 2^1 5^2 7^2 8^1 10^2 \\
\to\ & 1^1 2^1 5^4 7^2 8^1 10^1 \to 1^1 2^1 5^6 7^2 8^1 \to 1^1 2^1 4^2 5^6 7^2 \to 1^1 2^3 4^1 5^6 7^2 \\
\to\ & 1^1 2^5 5^6 7^2 \to 1^3 2^4 5^6 7^2 \to 1^5 2^3 5^6 7^2 \to 1^7 2^2 5^6 7^2 \\
\to\ & 1^9 2^1 5^6 7^2 \to 1^{11} 5^6 7^2 \to 1^{11} 5^3 7^2 15^1 \to 1^{11} 7^2 15^2 \\
\to\ & 1^8 3^1 7^2 15^2 \to 1^5 3^2 7^2 15^2 \to 1^2 3^3 7^2 15^2 \to \mathbf{1^2 7^2 9^1 15^2}
\end{aligned}$$

The bijection ψ is similar in spirit to Glaisher's bijection: given $\lambda = 1^{m_1} 2^{m_2} 4^{m_4} 5^{m_5} \cdots \in \mathcal{A}$ and $j \in \mathbb{P}$, the number of copies of part $2j-1$ in $\psi(\lambda)$ is equal to the k-th digit in the ternary expansion of l, where k is the highest power of 3 dividing $2j-1$, $2j-1 = 3^k r$, and $l = \sum_i 2^i m_{r2^i}$. $\square$

2.3 Equivalent sequences and graphs

Choose equivalent sequences $\overline{a}$, $\overline{b}$. Define a directed graph G_φ on $\operatorname{supp}(\overline{a}) \cup \operatorname{supp}(\overline{b})$ by drawing an edge from i to j if $\varphi(j) = i$; an arrow from i to j therefore means that O'Hara's algorithm simultaneously removes copies of i and adds copies of j. Each vertex v has $\operatorname{indeg} v \le 1$, $\operatorname{outdeg} v \le 1$ and $\operatorname{indeg} v + \operatorname{outdeg} v \ge 1$. The graph splits into connected components of the following five types:

(i) cycles of length $m \ge 1$;
(ii) paths of length $m \ge 2$;
(iii) infinite paths with an ending point, but without a starting point;
(iv) infinite paths with a starting point, but without an ending point;
(v) infinite paths without a starting point or an ending point.

Figure 1 shows portions of graphs G_φ for certain φ:

(1) $\overline{a} = (1,1,4,5,3,1,1,\ldots)$, $\overline{b} = (1,1,5,3,4,1,1,\ldots)$, $\varphi(3) = 4$, $\varphi(4) = 5$, $\varphi(5) = 3$, $\varphi(i) = i$ for $i \ne 3,4,5$; components of G_φ are of type (i);
(2) $\overline{a} = (\infty,1,2,3,\infty,\infty,\infty,\ldots)$, $\overline{b} = (2,3,4,\infty,\infty,\infty,\infty,\ldots)$, $\varphi(2) = 1$, $\varphi(3) = 2$, $\varphi(4) = 3$; G_φ is of type (ii);
(3) the distinct/odd case: $\overline{a} = (2,2,\ldots)$, $\overline{b} = (\infty,1,\infty,1,\ldots)$, $\varphi(i) = 2i$; components of G_φ are of type (iii);
(4) the odd/distinct case: $\overline{a} = (\infty,1,\infty,1,\ldots)$, $\overline{b} = (2,2,\ldots)$, $\varphi(i) = i/2$; components of G_φ are of type (iv);
(5) $\overline{a} = (2,2,1,2,2,1,\ldots)$ and $\overline{b} = (3,1,3,1,\ldots)$, φ given by (1); components of G_φ are of types (i) and (v). $\square$

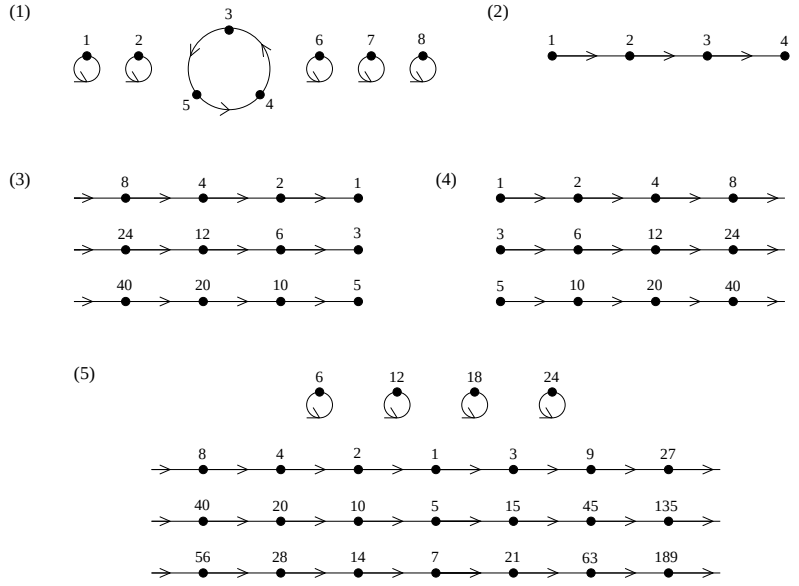


Fig. 1. Examples of graphs G_φ.

2.4 Scissor-congruence and Π*-congruence*

We say that convex polytopes A, B in $\mathbb{R}^m$ are *congruent*, write $A \simeq B$, if B can be obtained from A by rotation and translation. For convex polytopes $P, Q \subset \mathbb{R}^m$, we say that they are *scissor-congruent* if P can be cut into finitely many polytopes which can be rearranged and assembled into Q, i.e. if P and Q are the disjoint union of congruent polytopes: $P = \cup_{i=1}^n P_i$, $Q = \cup_{i=1}^n Q_i$, $P_i \simeq Q_i$.

Let π be a linear functional on $\mathbb{R}^m$. If Q_i can be obtained from P_i by a translation by a vector in the hyperplane $\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^m \colon \pi(\mathbf{x}) = 0\}$, we say that P and Q are π*-congruent*. If P and Q are π-congruent for some linear functional π, we say that they are Π*-congruent*.

If P can be cut into countably many polytopes which can be translated by a vector in the hyperplane $\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^m \colon \pi(\mathbf{x}) = 0\}$ and assembled into Q, we say that P and Q are *approximately* π*-congruent*. We say that they are *approximately* Π*-congruent* if they are approximately π-congruent for some linear functional π. If P and Q are approximately π-congruent, there exist, for every $\varepsilon > 0$, π-congruent polytopes $P_\varepsilon \subseteq P$ and $Q_\varepsilon \subseteq Q$, such that $\mathrm{vol}(P \setminus P_\varepsilon) < \varepsilon$ and $\mathrm{vol}(Q \setminus Q_\varepsilon) < \varepsilon$.

Finally, let $\mathbf{R}(a_1, \dots, a_m) = [0, a_1) \times \cdots \times [0, a_m)$ be a box in $\mathbb{R}^m$, and let $R(a_1, \dots, a_m) = \mathbf{R}(a_1, \dots, a_m) \cap \mathbb{Z}^m$ be the set of its integer points.

Let $d = 2$ and $\pi(x, y) = x + y$. Euclid's algorithm on (a, b) yields a π-congruence between $\mathbf{R}(a, b)$ and $\mathbf{R}(b, a)$: if $b = r_1 a + s_1$ with $0 \leq s_1 < a$, divide $[0, a) \times [0, r_1 a)$ into r_1 squares with side a, and translate the square $[0, a) \times [ia, (i + 1)a)$ by the vector $(ia, -ia)$ to $[ia, (i + 1)a) \times [0, a)$. Then write $a = r_2 s_1 + s_2$ with $0 \leq s_2 < s_1$, divide $[0, a) \times [r_1 a, b)$ into r_2 squares with side s_1, and translate the square $[is_1, (i + 1)s_1) \times [r_1 a, b)$ by the vector $(r_1 a - is_1, is_1 - r_1 a)$ to $[r_1 a, b) \times [is_1, (i+1)s_1)$. Continue until the remainder s_i is equal to 0. The first drawing of Figure 2 gives an example.

The second drawing shows that boxes $\mathbf{R}(12, 8)$ and $\mathbf{R}(32, 3)$ are π-congruent for $\pi(x, y) = x + 4y$. Finally, in Figure 3 we give a π-congruence between $\mathbf{R}(4, 5, 3)$ and $\mathbf{R}(5, 3, 4)$ for $\pi(x, y, z) = 3x + 4y + 5z$. $\square$

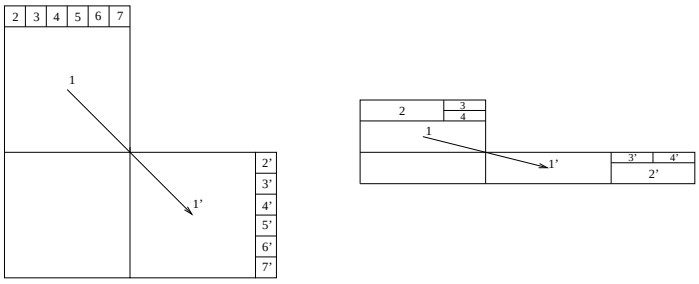


Fig. 2. Two Π-congruences.

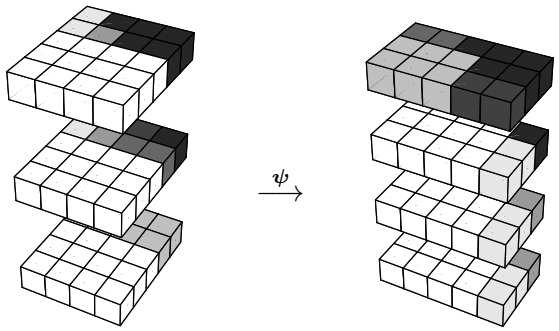


Fig. 3. π-congruence between $\mathbf{R}(4, 5, 3)$ and $\mathbf{R}(5, 3, 4)$.

3 Main results

3.1 Continuous O'Hara's algorithm and Π*-congruences*

Take the case when G_φ is a cycle $i_1 \to i_m \to i_{m-1} \to \ldots \to i_1$. In this case, $\varphi(i_1) = i_2$, $\varphi(i_2) = i_3$, etc. Throughout this section, identify a partition

$i_1^{t_1} \cdots i_m^{t_m}$ with the vector $\mathbf{t} = (t_1, \ldots, t_m)$. By Theorem 2, O'Hara's algorithm defines a bijection $\psi\colon R(a_1, \ldots, a_m) \to R(b_1, \ldots, b_m)$, where $i_j a_j = i_{j+1} b_{j+1}$ for all j, where the indices are taken cyclically. The following algorithm (see also Theorem 3) generalizes ψ to the continuous setting. It gives a bijection $\boldsymbol{\psi}\colon \mathbf{R}(a_1, \ldots, a_m) \to \mathbf{R}(b_1, \ldots, b_m)$, which is defined also for non-integer a_j, b_j. When a_j, b_j are integers, it is an extension of $\psi\colon R(a_1, \ldots, a_m) \to R(b_1, \ldots, b_m)$. As an immediate corollary, we prove that two boxes with rational coordinates and with equal volume are Π-congruent. We can use Theorem 3 to give an alternative proof of Theorem 2.

Algorithm 2 (continuous O'Hara's algorithm)

`Fix:` $\mathbf{i} = (i_1, \ldots, i_m) \in \mathbb{R}_+^m$
$\mathbf{a} = (a_1, \ldots, a_m) \in \mathbb{R}_+^m$, $\mathbf{b} = (b_1, \ldots, b_m) \in \mathbb{R}_+^m$ with $i_j a_j = i_{j+1} b_{j+1}$
`Input:` $\mathbf{t} \in \mathbf{R}(a_1, \ldots, a_m)$
`Set:` $\mathbf{s} \leftarrow \mathbf{t}$
`While:` $\mathbf{s}$ contains a coordinate $s_j \geq b_j$
`Do:` $s_j \leftarrow s_j - b_j$, $s_{j-1} \leftarrow s_{j-1} + a_{j-1}$
`Output:` $\boldsymbol{\psi}(\mathbf{t}) \leftarrow \mathbf{s}$

It is clear that the algorithm starts with an element of $P = \mathbf{R}(a_1, \ldots, a_m)$ and, if the while loop terminates, outputs an element of $Q = \mathbf{R}(b_1, \ldots, b_m)$. It is not obvious, however, that the loop terminates in every case, or that the output $\boldsymbol{\psi}(\mathbf{t})$ and the number of steps $\mathbf{L}_\varphi(\mathbf{t})$ depend only on $\mathbf{t}$, not on the choices made in the while loop.

Theorem 3 *Algorithm 2 has the following properties.*

(1) The algorithm stops after a finite number of steps, and the resulting vector $\boldsymbol{\psi}(\mathbf{t})$ and the number of steps $\mathbf{L}_\varphi(\mathbf{t})$ are independent of the choices made during the execution of the algorithm.

(2) The algorithm defines a bijection $\boldsymbol{\psi}\colon P \to Q$ which satisfies $\boldsymbol{\psi}(\mathbf{t}) - \mathbf{t} \in \mathcal{H}$, where $\mathcal{H}$ is the hyperplane defined by $i_1 x_1 + \ldots + i_m x_m = 0$.

(3) We have

$$\mathbf{L}_\varphi(\mathbf{t} + \mathbf{t}') \geq \mathbf{L}_\varphi(\mathbf{t}) + \mathbf{L}_\varphi(\mathbf{t}') \text{ for every } \mathbf{t}, \mathbf{t}', \mathbf{t} + \mathbf{t}' \in P.$$

In particular, $\mathbf{L}_\varphi(\mathbf{t}') \leq \mathbf{L}_\varphi(\mathbf{t})$ if $\mathbf{t}' \leq \mathbf{t}$.

(4) Let $\mathbf{t}, \mathbf{t}' \in P$, $\mathbf{s} = \boldsymbol{\psi}(\mathbf{t})$, with $t_j \leq t'_j < t_j + \varepsilon_j$, where $\varepsilon_j = b_j - s_j$. Then

$$\boldsymbol{\psi}(\mathbf{t}') - \mathbf{t}' = \boldsymbol{\psi}(\mathbf{t}) - \mathbf{t} \quad \text{and} \quad \mathbf{L}_\varphi(\mathbf{t}') = \mathbf{L}_\varphi(\mathbf{t}).$$

(5) For all $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_+^m$, we have

$$\max_{\mathbf{t} \in P} \mathbf{L}_\varphi(\mathbf{t}) \;=\; \operatorname{lcm}(c_1, \ldots, c_m) \cdot \left(\frac{1}{c_1} + \ldots + \frac{1}{c_m} \right) - m,$$

where $c_j = a_1 \cdots a_{j-1} b_j \cdots b_{m-1}$.

We call boxes $P = \mathbf{R}(a_1, \ldots, a_m)$, $Q = \mathbf{R}(b_1, \ldots, b_m)$ *relatively rational* if there exists λ, $\lambda \neq 0$, such that $\lambda a_j \in \mathbb{Z}, \lambda b_j \in \mathbb{Z}$. Clearly, two boxes P and Q with rational side-lengths are relatively rational.

Corollary 4 *Boxes* $P = \mathbf{R}(a_1, \ldots, a_m)$, $Q = \mathbf{R}(b_1, \ldots, b_m)$ *with equal volume are approximately* Π*-congruent. Moreover, when* P *and* Q *are relatively rational and have equal volume, they are* Π*-congruent.*

For $j = 1, \ldots, m$, take $i_j = a_1 \cdots a_{j-1} b_{j+1} \cdots b_m$. Clearly $i_j a_j = i_{j+1} b_{j+1}$ for $j = 1, \ldots, m-1$, and $a_1 \cdots a_m = b_1 \cdots b_m$ implies $i_m a_m = i_1 b_1$. Therefore, the numbers i_j, a_j, b_j satisfy the conditions of Algorithm 2. By Theorem 3 part (2), the algorithm defines a bijection $\boldsymbol{\psi} \colon P \to Q$. Parts (4) and (2) of Theorem 3 imply that we can cut P into (countably many) smaller boxes, each of which is translated by a vector in the plane $i_1 x_1 + \ldots + i_m x_m = 0$.

If P and Q are relatively rational, we can assume without loss of generality that all a_j, b_j are integers. For any integer vector $\mathbf{t}$, we have $\boldsymbol{\psi}(\mathbf{t}') - \mathbf{t}' = \boldsymbol{\psi}(\mathbf{t}) - \mathbf{t}$ and $\mathbf{L}_\varphi(\mathbf{t}') = \mathbf{L}_\varphi(\mathbf{t})$ whenever $t_j \leq t'_j < t_j + 1$, so P and Q are divided into a finite number (at most $a_1 \cdots a_m$) of boxes.

Even in the 3-dimensional case the Π-congruence defined by the algorithm can be quite complex, as the next figure suggests. Here the same shading is used for parallel translations by the same vector. $\square$

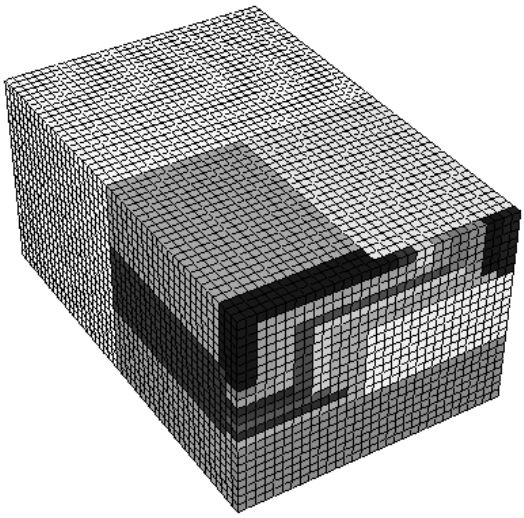


Fig. 4. The decomposition of the box $\mathbf{R}(31, 47, 23)$ given by O'Hara's algorithm (only the top, right, and back sides are shown) .

3.2 Complexity of O'Hara's algorithm

The complexity of O'Hara's algorithm has been an open problem, with the exception of the elementary distinct/odd case (see [O84]).

It turns out that the complexity depends heavily on the type of the graph G_φ defined in Subsection 2.3. Part (5) of Theorem 3 gives the maximum number of steps that O'Hara's algorithm takes when G_φ is a cycle. The following lemma gives an estimate for $\mathcal{L}_\varphi(n)$ when G_φ is a path.

Lemma 5 *Let G_φ be a finite or infinite path on $\mathcal{I} \subseteq \mathbb{P}$. Then $\mathcal{L}_\varphi(n) \leq n(\log n + 1)$. Moreover, if*

$$D = \sum_{i \in \mathcal{I}} \frac{1}{ia_i} = \sum_{j \in \mathcal{I}} \frac{1}{jb_j} < \infty,$$

then $\mathcal{L}_\varphi(n) \leq Dn$. Here, by $\log n$ we mean the natural logarithm of n.

Theorem 6 *Let $\overline{a}, \overline{b}$ be φ-equivalent sequences.*

(1) If G_φ has only a finite number of cycles of length > 2, then $\mathcal{L}_\varphi(n) = O(n \log n)$, and the constants implied by the O-notation are universal.
(2) If G_φ has only a finite number of cycles of length $> m$ for some $m > 2$, then $\mathcal{L}_\varphi(n) = O(n^{m-1})$, and the constants implied by the O-notation depend only on m.

The following theorem gives the corresponding lower bound on the worst case complexity. It shows that the estimates of Theorem 6 are close to being sharp.

Theorem 7 *There exist φ-equivalent sequences $\overline{a}$ and $\overline{b}$, such that:*

(1) G_φ is a path and $\mathcal{L}_\varphi(n) = \Omega(n \log \log n)$;
(2) G_φ contains only cycles of length $\leq m$ and $\mathcal{L}_\varphi(n) = \Omega(n^{m-1-\varepsilon})$ for every $\varepsilon > 0$;
(3) $\mathcal{L}_\varphi(n) = \exp \Omega(\sqrt[3]{n})$.

In other words, depending on the type of the graph, we have nearly matching upper and lower bounds on $\mathcal{L}_\varphi(n)$. For example, for an m-cycle, Theorem 6 shows that $\mathcal{L}_\varphi(n)$ is $O(n^{m-1})$, while Theorem 7 shows that it is $\Omega(n^{m-1-\varepsilon})$ for every $\varepsilon > 0$. Similarly, part (3) shows that O'Hara's algorithm can be very slow in general since the total number of partitions of n is asymptotically $\exp \Theta(\sqrt{n})$.

3.3 O'Hara's algorithm as an integer linear programming problem

Let us now give a new description of O'Hara's algorithm.

Proposition 8 *Let* $\mathbf{i}, \mathbf{a}, \mathbf{b} \in$ *be as above such that* $i_j a_j = i_{j+1} b_{j+1}$ *for* $j = 1, \ldots, m$. *Fix a vector* $\mathbf{t} \in \mathbf{R}(a_1, \ldots, a_m)$. *Then* $\mathbf{s} = \boldsymbol{\psi}(\mathbf{t})$ *satisfies the following:*

$$\mathbf{s} = \mathbf{t} + A\mathbf{k},$$

where

$$A = \begin{pmatrix} -b_1 & a_1 & 0 & \cdots & 0 \\ 0 & -b_2 & a_2 & \cdots & 0 \\ 0 & 0 & -b_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_m & 0 & 0 & \cdots & -b_m \end{pmatrix}$$

and $\mathbf{k} = (k_1, \ldots, k_m)$ *is the unique vector minimizing*

$$k_1 + \ldots + k_m$$

with constraints

$$\mathbf{k} \in \mathbb{Z}^m, \qquad \mathbf{k} \geq \mathbf{0}, \qquad A\mathbf{k} \geq -\mathbf{t}, \qquad A\mathbf{k} \leq \mathbf{b} - \mathbf{1} - \mathbf{t}.$$

Proposition 8 can be used to obtain a significant speed-up of (the usual) O'Hara's algorithm, in the case when G_φ contains only cycles of bounded length. Namely, we obtain the following result.

Theorem 9 *Let* $\overline{a} \sim_\varphi \overline{b}$. *If the lengths of cycles of* G_φ *are bounded, there exists a deterministic algorithm which computes* $\psi(\lambda)$ *in* $O(n \log n)$ *steps for* $\lambda \in \mathcal{A}_n$.

Without loss of generality, the support of $\lambda \in \mathcal{A}_n$ is contained in one of the connected components of G_φ. If this connected component is a path, O'Hara's algorithm takes $O(n \log n)$ steps by Lemma 5. If it is a cycle of length m, we can use the algorithm described in, say, [S86, Corollary 18.7b] to compute $\psi(\lambda)$ in $O(\log^c n)$ steps for some c. Obviously the $O(n \log n)$ term dominates.

4 Final remarks

4.1

The polynomial time algorithm in the proof of Theorem 9 is given implicitly, by using the general results in integer linear programming. It is saying that the function $\psi : \mathcal{A}_n \to \mathcal{B}_n$ can be computed much faster, by circumventing the elegant construction of O'Hara's algorithm. It would be interesting to give an explicit construction of such an algorithm.

In a different direction, it might prove useful to restate other involution principle bijections in the language of linear programming, such as the Rogers-Ramanujan bijection in [GM81b] or in [BP06]. If this works, this might lead to a new type of a bijection between these two classes of partitions. Alternatively, this might resolve the conjecture by the second author on the mildly exponential complexity of Garsia-Milne's Rogers-Ramanujan bijection, see [P06, Conjecture 8.5].

4.2

Note the gap between the number $\exp \Theta(\sqrt{n})$ of partitions of n and the lower bound $\mathcal{L}_\varphi(n) = \exp \Omega(\sqrt[3]{n})$ in Theorem 7. It would be interesting to decide which of the two worst complexity bounds on the number of steps of O'Hara's algorithm is closer to the truth.

Note that we applied our linear programming approach only in the bounded cycle case. We do not know if there is a way to apply the same technique to the general case. However, we believe that there are number theoretic obstacles preventing that and in fact, computing O'Hara's bijection as a function on partitions may be hard in the formal complexity sense.

4.3

Recently, variations on the O'Hara's bijection and applications of rewrite systems were found in [SSM04] and [K04,K07]. It would be interesting to see connections between our analysis and this work.

Recall also that the 2-dimensional case can be viewed as the Euclid algorithm which in turn corresponds to the usual continued fractions (see Example **??**). Thus the geometry of ψ can be viewed as a delicate multidimensional extension of continued fractions. Given the wide variety of (different) multidimensional continued fractions available in the literature, it would be interesting to see if there is a connection to at least one of these notions.

Acknowledgments. We are grateful to George Andrews and Dennis Stanton for their interest in the paper and to Kathy O'Hara for sending us a copy of her thesis [O84]. The second named author was supported by the NSF. He would also like to thank Vladimir Arnold, Elena Korkina and Mark Sapir for teaching him about multidimensional continued fractions.

References

[A98] G. E. Andrews, *The theory of partitions* (Second ed.), Cambridge U. Press, Cambridge, 1998.

[BP06] C. Boulet and I. Pak, A combinatorial proof of the Rogers-Ramanujan identities, *J. Combin. Theory Ser. A* **113** (2006), 1019–1030.

[GM81a] A. M. Garsia and S. C. Milne, Method for constructing bijections for classical partition identities, *Proc. Nat. Acad. Sci. U.S.A.* **78** (1981), no. 4, 2026–2028.

[GM81b] A. M. Garsia and S. C. Milne, A Rogers-Ramanujan bijection *J. Combin. Theory Ser. A* **31** (1981), 289–339.

[G83] B. Gordon, Sieve-equivalence and explicit bijections, *J. Combin. Theory Ser. A* **34** (1983), 90–93.

[K04] M. Kanovich, Finding direct partition bijections by two-directional rewriting techniques, *Discrete Math.* **285** (2004), 151–166.

[K07] M. Kanovich, The two-way rewriting in action: removing the mystery of Euler-Glaisher's map, *Discrete Math.* **307** (2007), 1909–1935.

[KP] M. Konvalinka and I. Pak, Geometry and complexity of O'Hara's algorithm, to appear in *Adv. Appl. Math.*

[O84] K. M. O'Hara, *Structure and Complexity of the Involution Principle for Partitions*, Ph.D. thesis, UC Berkeley, California, 1984, 135 pp.

[O88] K. M. O'Hara, Bijections for partition identities, *J. Combin. Theory Ser. A* **49** (1988), 13–25.

[P04a] I. Pak, Partition identities and geometric bijections, *Proc. A.M.S.* **132** (2004), 3457–3462.

[P04b] I. Pak, The nature of partition bijections I. Involutions, *Adv. Applied Math.* **33** (2004), 263–289.

[P06] I. Pak, Partition bijections, a survey, *Ramanujan J.* 12 (2006), 5–75.

[P] I. Pak, The nature of partition bijections II. Asymptotic stability, preprint, 32 pp., available at `http://www-math.mit.edu/~pak/`

[PV05] I. Pak and E. Vallejo, Combinatorics and geometry of Littlewood-Richardson cones, *Europ. J. Combin.* **26** (2005), 995–1008.

[R82] J. B. Remmel, Bijective proofs of some classical partition identities. *J. Combin. Theory Ser. A*, **33** (1982), 273–286.

[S86] A. Schrijver, *Theory of linear and integer programming*, John Wiley, Chichester, 1986.

[SSM04] J. A. Sellers, A. V. Sills and G. L. Mullen, Bijections and congruences for generalizations of partition identities of Euler and Guy, *Electronic J. Combin.* **11** (2004), no. 1, RP 43, 19 pp.

References

[1]

Alphabetical list of authors