Seventh Cologne-Twente Workshop on Graphs and Combinatorial Optimization

Scientific Program

Gargnano, Italy

May 13th to 15th, 2008

INDEX:

Graph theory (I)

Room A, Session 1

Tuesday 13, 10:30-12:00

Improving the gap of Erdős-Pósa property for minor-closed graph classes [1]

Fedor V. Fomin [a,2] Saket Saurabh [a,2] Dimitrios M. Thilikos [b,3,*]

[a] *Department of Informatics University of Bergen PO Box 7800, 5020 Bergen, Norway*

[b] *Department of Mathematics, University of Athens, Panepistimioupolis, GR-15784 Athens, Greece.*

Abstract

Let $\mathcal{H}$ and $\mathcal{G}$ be graph classes. We say that $\mathcal{H}$ has the Erdős-Pósa property for $\mathcal{G}$ if for any graph $G \in \mathcal{G}$, the minimum vertex covering of all $\mathcal{H}$-subgraphs of G is bounded by a function f of the maximum packing of $\mathcal{H}$-subgraphs in G (by $\mathcal{H}$-subgraph of G we mean any subgraph of G that belongs to $\mathcal{H}$). In his monograph "Graph Theory", R. Diestel proves that if $\mathcal{H}$ is the class of all graphs that can be contracted to a fixed planar graph H, then $\mathcal{H}$ has the Erdös-Pósa property for the class of all graphs (with an exponential bounding function). In this note, we give an alternative proof of this result with a better (still exponential) bounding function. Our proof, for the case when $\mathcal{G}$ is some non-trivial minor-closed graph class, yields a low degree polynomial bounding function f. In particular $f(k) = O(k^{1.5})$.

Key words: Erdős-Pósa property, treewidth, graph packing, graph covering

1. Introduction

Given a graph G we denote by $V(G)$ and $E(G)$ its vertex and edge set respectively. A graph class $\mathcal{G}$ is called *non-trivial* if it does not contain all graphs. We use the notation $H \subseteq G$ to denote that H is a subgraph of G. We say that a graph G is a *$\mathcal{G}$-subgraph of a graph G'* if $G \subseteq G'$ and $G \in \mathcal{G}$.

[1] This research has been done while the first two authors were visiting the Department of Mathematics of the National and Kapodistrian University of Athens during October 2007.

[2] Supported by the Research Council of Norway.

[3] Supported by the Project "Kapodistrias" (ΑΠ 736/24.3.2006) of the National and Kapodistrian University of Athens (project code: 70/4/8757).

Let $\mathcal{H}$ be a class of graphs. Given a graph G, we define the *covering number of G with respect to the class $\mathcal{H}$* as

$$\textbf{cover}_{\mathcal{H}}(G) = \min\{k \mid \exists\, S \subseteq V(G)\, \forall_{H\in\mathcal{H}} H \not\subseteq G - S\}.$$

In other words, $\textbf{cover}_{\mathcal{H}}(G) \leq k$ if there is a set of at most k vertices meeting any $\mathcal{H}$-subgraph of G. We also define the *packing number of G with respect to the class $\mathcal{H}$* as

$$\begin{aligned}\textbf{pack}_{\mathcal{H}}(G) = \max\{k \mid \exists \text{ a partition } V_1, \ldots, V_k \text{ of } V(G)\\ \text{such that } \forall_{i\in\{1,\ldots,k\}} \exists_{H\in\mathcal{H}}\ H \subseteq G[V_i]\}.\end{aligned}$$

Less formally, $\textbf{pack}_{\mathcal{H}}(G) \geq k$ if G contains k vertex-disjoint $\mathcal{H}$-subgraphs.

A graph class $\mathcal{H}$ satisfies the Erdös-Pósa property for some graph class $\mathcal{G}$ if there is a function f (depending only on $\mathcal{H}$ and $\mathcal{G}$) such that, for any graph $G \in \mathcal{G}$,

$$\textbf{pack}_{\mathcal{H}}(G)\ \leq \textbf{cover}_{\mathcal{H}}(G)\ \leq f(\textbf{pack}_{\mathcal{H}}(G)). \tag{1}$$

We say that a graph G *can be contracted to* H if H can be obtained from G after a series of edge contractions (the *contraction* of an edge $e = (u, v)$ in G results in a graph G', in which u and v are replaced by a new vertex v_e and in which for every neighbour w of u or v in G, there is an edge (w, v_e) in G'). We say that H *is a minor of* G if some subgraph of G can be contracted to H. We say that a graph class $\mathcal{G}$ is *minor-closed* if any minor of a graph in $\mathcal{G}$ is again a member of $\mathcal{G}$. We denote by $\mathcal{M}(H)$ the class of graphs that can be contracted to H.
Theorem 1 (Corollary 12.3.10 and Exercise 39 in [2]) *$\mathcal{M}(H)$ satisfies the Erdös-Pósa property for all graphs if and only if H is a connected planar graph.*

According to the proof of Theorem 1, the bounding function $f(k)$ of Relation (1) is exponential in k. Given a connected planar graph H and a graph class $\mathcal{G}$ we denote by $f_{H,\mathcal{G}} : \mathbb{N} \to \mathbb{N}$ the optimal upper bounding function with the property that, for any graph $G \in \mathcal{G}$, Relation (1) holds when $\mathcal{H} = \mathcal{M}(H)$ (we know that $f_{H,\mathcal{G}}$ exists because of Theorem 1).

For instance, according to the classic result of Erdös and Pósa [3] if $H = K_3$ and $\mathcal{G}$ contains all the graphs, then $f_{H,\mathcal{G}}(k) = O(k \cdot log k)$. However, if we restrict $\mathcal{G}$ to be a class of planar graphs, then $f_{H,\mathcal{G}}(k) = O(k)$ [4]. In this note, we conjecture that this last fact extends when H is any planar graph and when $\mathcal{G}$ is any non-trivial minor-closed graph class.
Conjecture 2 *If $\mathcal{G}$ is a non-trivial minor-closed graph class, then $f_{H,\mathcal{G}}$ is a linear function for any planar graph H.*

The purpose of this short note is to prove that the above conjecture holds if we ask for a polynomial bound for $f_{H,\mathcal{G}}$, thereby sharpening the result of Diestel in [2].

2. The main result and the proof

Our main result is the following:
Theorem 3 *Let H be for some planar graph and let $\mathcal{H}$ be the class of graphs that are contractible to H. Let also $\mathcal{G}$ be a non-trivial minor-closed graph class. Then there is a constant $c_{\mathcal{G},H}$ depending only on $\mathcal{G}$ and H such that for every graph $G \in \mathcal{G}$, it holds that*

$$\mathbf{p}ack_{\mathcal{H}}(G) \leq \mathbf{c}over_{\mathcal{H}}(G) \leq c_{\mathcal{G},H} \cdot (\mathbf{p}ack_{\mathcal{H}}(G))^{3/2}.$$

The proof of Theorem 3 will be the immediate consequence of Lemmata 1 and 2 below. Before we state and prove them, we need first to define the notions of tree decomposition and treewidth.

Let G be a graph. A *tree decomposition* of G is a pair $(T, \mathcal{X} = \{X_t\}_{t \in V(T)})$ where the following conditions hold.

- $\cup_{u \in V(T)} = V(G)$
- $\forall_{e \in E(G)}\ \exists_{t \in V(T)} : e \subseteq X_t$
- $\forall_{v \in V(G)}\ T[\{t \mid v \in X_t\}]$ is connected.

The *width* of a tree decomposition is $\max_{t \in V(T)} |X| - 1$ and the *treewidth* of G is the minimum width over all the tree decompositions of G.

Our first observation is the following.
Lemma 1 *Let H be a connected planar graph and let $\mathcal{H} = \mathcal{M}(H)$. Let also $\mathcal{G}$ be a non-trivial minor closed graph class. Then, there is a constant $c_{\mathcal{G},H}$, depending only on $\mathcal{G}$ and H such that for any graph G, **tw**$(G) \leq c_{\mathcal{G},H} \cdot (\mathbf{p}ack_{\mathcal{H}}(G))^{1/2}$.*

Proof. Let $k = \mathbf{pack}_{\mathcal{M}(H)}(G)$. During this proof, for any positive integer t we will denote by Γ_t the $(t \times t)$-grid. Let

$$c_H = \min\{r \mid H \text{ is a minor of the } (r \times r)\text{-grid}\}$$

and notice that if $m = \lceil k^{1/2} \rceil + 1$, then $\mathbf{pack}_{\mathcal{M}(H)}(\Gamma_{m \cdot c_H}) > k$. We conclude that G does not contain $\Gamma_{m \cdot c_H}$ as a minor. From the main result in [1], there is a constant $c_{\mathcal{G}}$ depending only on $\mathcal{G}$ such that $\mathbf{tw}(G) \leq c_{\mathcal{G}} \cdot m \cdot c_H$ and the lemma follows.

For the proof of the next Lemma, we will enhance the definition of a tree decomposition $(T, \mathcal{X})$ as follows: T is a tree rooted on some node r where $X_r = \emptyset$, each of its nodes have at most two children and can be one of the following

(1) *Introduce node*: a node t that has only one child t' where $X_t \supset X_{t'}$ and such that t' is not an introuce node.
(2) *Forget node*: a node t that has only one child t' where $X_t \subset X_{t'}$ and such that t' is not a forget node.
(3) *Join node*: a node t with two children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$.

(4) *Base node*: a node t that is a leaf of t, is different than the root and $X_t = \emptyset$.

Notice that, according the the above definitions, the root r of T is either a forget or a join node. It is easy to see that any tree decomposition can be transformed to one with the above requirements while maintaining the same width. From now one when we refer to a tree decomposition $(T, \mathcal{X})$ we will presume the above requirements.

Given a tree decomposition $(T, \mathcal{X})$ and some node t of T, we define as T_t the subtree of T rooted on t. Clearly, if r is the root of T, it holds that $T_r = T$. We also define $G_t = G[\cup_{s \in V(T_t)} X_s]$ and $G_t^- = G_t - X_t$.

Lemma 2 *Let H be a connected planar graph and let $\mathcal{H} = \mathcal{M}(H)$. Then for any graph G, it holds that* ***cover***$_{\mathcal{H}}(G) \leq (\textbf{\textit{tw}}(G) + 1) \cdot$ ***pack***$_{\mathcal{H}}(G)$.

Proof. Let $(T, \mathcal{X})$ be a rooted tree decomposition rooted on r with width at most k. We set up a labelling $p : V(T) \to \mathbb{N} \cup \{0\}$ such that

$$p(t) = \textbf{pack}_{\mathcal{H}}(G_t^-).$$

The following observations are direct consequences of the definitions.

Observation 1. If $t \in V(T)$ is an introduce node with t' as child, then $p(t') = p(t)$. This holds because then $G_{t'}^- = G_t^-$.

Observation 2. If $t \in V(T)$ is an forget node with t' as child, then $p(t') \leq p(t)$. This holds because then $G_{t'}^- \subseteq G_t^-$.

Observation 3. If $t \in V(T)$ is a join node with t_1 and t_2 as children, then $p(t_1) + p(t_2) = p(t)$. This holds because $G_{t_1}^-$ and $G_{t_2}^-$ are disjoint graphs.

Observation 4. If $t \in V(T)$ is a base node, then $p(t) = 0$. This holds because then G_t is the empty graph.

Observation 5. $p(r) =$ ***pack***$_{\mathcal{H}}(G)$. This holds because, $X_r = \emptyset$ and thus $G_r^- = G_r = G$.

Let t be a node of T and let $t_1, \ldots, t_\rho$ be its children (clearly, $0 \leq \rho \leq 2$). We say that t is a *critical* node if $p(t_1) + \cdots + p(t_\rho) < p(t)$. From the Observations 1–4, only forget nodes of T can be critical. Given a node t of T we define $R(t)$ as the set of all critical nodes in T_t. We omit the proof of the following claim.

Claim 1. For any $t \in V(T)$, $|R(t)| \leq p(t)$.

Given a forget node t, we denote its child by $c(t)$. For any node $t \in V(T)$, we define $S(t) = \bigcup_{z \in R(t)} X_{c(z)}$. The proof of the next claim is technical and is omitted from this extended abstract.

Claim 2. For any $t \in V(T)$, the set $S(t)$ intersects all the $\mathcal{H}$-subgraphs of G_t^-.

Applying Claim 2, for $T_r = T$ we have that $S(r)$ meets all $\mathcal{H}$-subgraphs of $G_r^- = G_r = G$. From Claim 1, $|S(r)| \leq (k+1) \cdot p(r) = (k+1) \cdot \mathbf{pack}_{\mathcal{H}}(G)$ (Observation 5) and the lemma follows.

References

[1] Erik Demaine and MohammadTaghi Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, to appear.

[2] Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005.

[3] Paul Erdős and Louis Pósa. On independent circuits contained in a graph. *Canad. J. Math.*, 17:347–352, 1965.

[4] Ton Kloks, C. M. Lee, and Jiping Liu. New algorithms for k-face cover, k-feedback vertex set, and k-disjoint set on plane and planar graphs. In *The 28th International Workshop on Graph-Theoretic Concepts in Computer Science(WG 2002)*, page to appear. Springer, Lecture Notes in Computer Science, Berlin, 2002.

Fault-Free Hamiltonian Cycles in Pancake Graphs with Conditional Edge Faults

Ping-Ying Tsai [a,*] Jung-Sheng Fu [b] Gen-Heuy Chen [a]

[a] *Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, ROC*

[b] *Department of Electronics Engineering, National United University, Miaoli, Taiwan, ROC*

Key words: pancake graph, Hamiltonian cycles, Hamiltonicity, fault tolerance, Cayley graph, conditional edge faults.

1. Introduction and notations

Let G be a *graph* with the *vertex set* $V(G)$ and the *edge set* $E(G)$. Unless otherwise stated, we follow [12] for definitions and notations. A path (or cycle) in a graph G is called a *Hamiltonian path* (or *Hamiltonian cycle*) if it contains every vertex of G exactly once. A graph is called *Hamiltonian* if it has a Hamiltonian cycle. A graph is called *Hamiltonian connected* if every two vertices of G are connected by a Hamiltonian path. We use $\delta(G)$ to denote the minimal vertex degree of G, and we use $d_{u,v}$ to denote the distance between two vertices u and v.

The n-dimensional pancake graph, denoted by $\wp_n$, is a graph with the vertex set $V(\wp_n) = \{a_1a_2\cdots a_n | a_1a_2\cdots a_n$ is a permutation of $1, 2, \ldots, n\}$, and the edge set $E(\wp_n) = \{(a_1a_2\cdots a_i\cdots a_n, b_1b_2\cdots b_i\cdots b_n) | a_1a_2\cdots a_n, b_1b_2\cdots b_n \in V(\wp_n), 2 \leq i \leq n$, where $b_j = a_{i-j+1}$ if $1 \leq j \leq i$ and $b_j = a_j$ if $i < j \leq n\}$. The pancake graph is an instance of Cayley graphs [1]. $\wp_3$ and $\wp_4$ are illustrated in Fig. 1. It is easy to see that $\wp_n$ is an $(n-1)$-regular graph with $n!$ vertices. The pancake graph was introduced (and named) from the famous "pancake problem" whose answer is exactly the diameter of the corresponding pancake graph [5]. The diameter of $\wp_n$ is bounded above by $\frac{3(n+1)}{2}$ [6]. It is still an open problem to compute the exact diameter of the pancake graph. The pancake graph is vertex symmetric [1], but not edge symmetric [9]. Some other properties of pancake graphs can be found in [9–11].

For convenience, we use $\langle u \rangle_i$ to denote the ith leftmost digit of a vertex u, i.e., $\langle u \rangle_i = a_i$ if $u = a_1a_2\cdots a_n$, where $1 \leq i \leq n$. The edge $(a_1a_2\cdots a_n, a_ka_{k-1}\cdots a_2a_1a_{k+1}a_{k+2}\cdots a_n)$ is referred to as a *k-dimensional edge*, where $2 \leq k \leq n$. We use $N^{(k)}(u)$ to denote the neighbor of a vertex $u \in V(\wp_n)$ which is

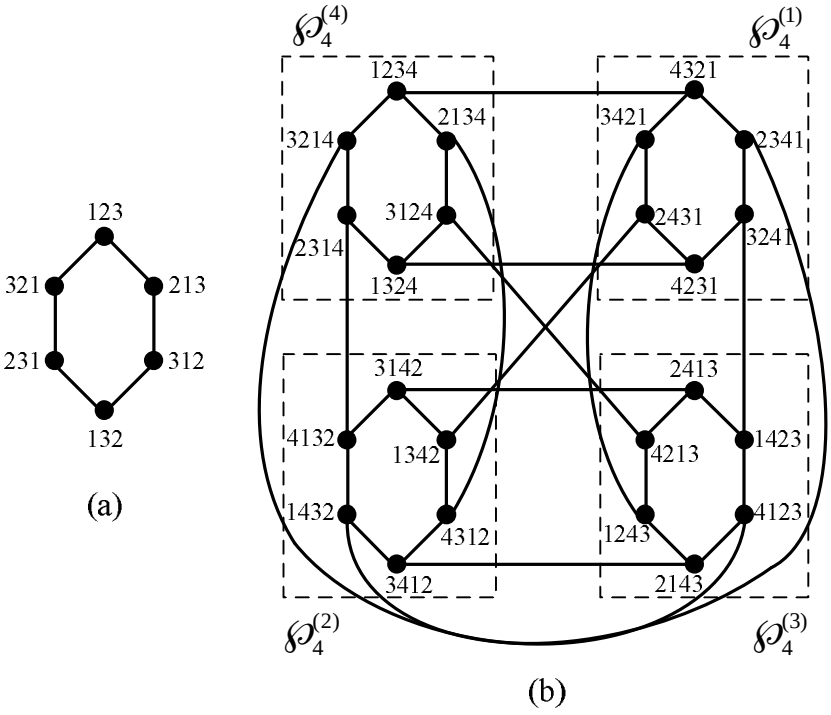


Fig. 1. Topologies of pancake graphs. (a) $\wp_3$ and (b) $\wp_4$.

connected to u by a k-dimensional edge, and $E^{(k)}(\wp_n)$ to denote the set of all k-dimensional edges in $\wp_n$. It can be observed from Fig. 1 that $\wp_4$ consists of four embedded $\wp_3$'s, denoted by $\wp_4^{(1)}$, $\wp_4^{(2)}$, $\wp_4^{(3)}$, and $\wp_4^{(4)}$. In general, $\wp_n$ comprises n embedded $\wp_{n-1}$'s: $\wp_n^{(r)}$ for $1 \leq r \leq n$, where $\wp_n^{(r)}$ is the subgraph of $\wp_n$ induced by those vertices u with $\langle u \rangle_n = r$. For $I \subseteq \{1, 2, \ldots, n\}$, we let $\wp_n^I$ denote the subgraph of $\wp_n$ induced by $\bigcup_{r \in I} V(\wp_n^{(r)})$.

An *interconnection network* (*network* for short) is usually represented by a graph where vertices represent processors and 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 pancake graph 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 [1]. Since faults may occur to networks, it is significant to consider faulty networks. Many fundamental problems such as diameter, routing, broadcasting, gossiping, and embedding were solved on various faulty networks. Two fault models were adopted before. One is the *random fault model* [4, 7], which assumes that the faults may occur everywhere without any restriction. The other is the *conditional fault model* [2, 3], which assumes that the fault distribution is subject to some constraints, e.g., two or more non-faulty links incident to each node. It is more difficult to solve problems under the conditional fault model than the random fault model. No previous work on the pancake graph considered the conditional fault model. If the random fault model is adopted, $\wp_n$ can tolerate at most $n-3$ edge faults, while retaining a fault-free Hamiltonian cycle [8]. We use $F(\subseteq E(\wp_n))$ to denote the set of edge faults in $\wp_n$. For $p \neq q$, we use $\tilde{E}_{p,q}(\wp_n)$ to denote the set of n-dimensional edges in $\wp_n$ that connect $\wp_n^{(p)}$ and

$\wp_n^{(q)}$. Some known results of $\wp_n$ are listed as follows.

Lemma 1.([8]) $|\tilde{E}_{p,q}(\wp_n)| = (n-2)!$ for all $1 \leq p \neq q \leq n$, where $n \geq 3$.

Lemma 2.([8]) $\wp_n - F$ is Hamiltonian as $|F| \leq n-3$, and Hamiltonian connected as $|F| \leq n-4$, where $n \geq 4$.

Lemma 3.([8]) Suppose that $u, v \in V(\wp_n)$ and $\langle u \rangle_n \neq \langle v \rangle_n$, where $n \geq 5$. For any $I \subseteq \{1, 2, \ldots, n\}$ and $|I| \geq 2$, if $\wp_n^{(r)} - F$ is Hamiltonian connected for all $r \in I$ and $|\tilde{E}_{i,j}(\wp_n) - F| \geq 3$ for all $i, j \in I$ and $i \neq j$, there exists a Hamiltonian path between u and v in $\wp_n^I - F$.

Lemma 4.([8]) Suppose that $u, v \in V(\wp_n^{(r)})$ and $u \neq v$, where $1 \leq r \leq n$ and $n \geq 4$. If $d_{u,v} \leq 2$, then $\left\langle N^{(n)}(u) \right\rangle_n \neq \left\langle N^{(n)}(v) \right\rangle_n$.

2. Results

First, we show two properties for $\wp_n$.

Lemma 5. Suppose that $e_1, e_2 \in E(\wp_4)$ and $e_1 \neq e_2$. There exists a Hamiltonian cycle in $\wp_4 - \{e_2\}$ that contains e_1.

Lemma 6. Suppose that $s, t \in V(\wp_n)$, $\langle s \rangle_1 = \langle t \rangle_1$, and $s \neq t$, where $n \geq 4$. For every $(x, y) \in E(\wp_n)$ with $\{x, y\} \cap \{s, t\} = \phi$, there exists a Hamiltonian path between s and t in $\wp_n$ that contains (x, y).

As a consequence of these lemmas, we obtain our main theorem. It is the first result on the fault tolerance of the pancake graph under the conditional fault model. Assuming that there were two or more non-faulty edges incident to each vertex, we show that $\wp_n$ contained a fault-free Hamiltonian cycle, even if there were up to $2n-7$ edge faults, where $n \geq 4$.

Theorem 1. $\wp_n - F$ is Hamiltonian if $|F| \leq 2n-7$ and $\delta(\wp_n - F) \geq 2$, where $n \geq 4$.

References

[1] S. B. Akers and B. Krishnamurthy. A Group-Theoretic Model for Symmetric Interconnection Networks. *IEEE Transactions on Computers*, vol. 38 (4), pp. 555-566, 1989.

[2] Y. A. Ashir and I. A. Stewart. Fault-Tolerant Embedding of Hamiltonian Circuits in k-ary n-cube. *SIAM Journal on Discrete Mathematics*, vol. 15 (3), pp. 317-328, 2002.

[3] M. Y. Chan and S. J. Lee. On the Existence of Hamiltonian Circuits in Faulty Hypercubes. *SIAM Journal on Discrete Mathematics*, vol. 4 (4), pp. 511-527, 1991.

[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*, vol. 44 (4), pp. 302-310, 2004.

[5] W. H. Gates and C. H. Papadimitriou. Bounds for Sorting by Prefix Reversal. *Discrete Mathematics*, vol. 27, pp. 47-57, 1979.

[6] M. H. Heydari and I. H. Sudborough. On the Diameter of the Pancake Network. *Journal of Algorithms*, vol. 25, pp. 67-94, 1997.

[7] S. Y. Hsieh. Embedding Longest Fault-Free Paths onto Star Graphs with More Vertex Faults. *Theoretical Computer Science*, vol. 337, pp. 370-378, 2005.

[8] C. N. Hung, H. C. Hsu, K. Y. Liang and L. H. Hsu. Ring Embedding in Faulty Pancake Graphs. *Information Processing Letters*, vol. 86, pp. 271-275, 2003.

[9] A. Kanevsky and C. Feng. On the Embedding of Cycles in Pancake Graphs. *Parallel Computing*, vol. 21, pp. 923-936, 1995.

[10] C. K. Lin, H. M. Huang, and L. H. Hsu. The Super Connectivity of the Pancake Graphs and the Super Laceability of the Star Graphs. *Theoretical Computer Science*, vol. 339, pp. 257-271, 2005.

[11] K. Qiu, S. G. Akl, and H. Meijer. On Some Properties and Algorithms for the Star and Pancake Interconnection Networks. *Journal of Parallel and Distributed Computing*, vol. 22, pp. 16-25, 1994.

[12] D. B. West. *Introduction to Graph Theory* (2nd Edition). Prenctice Hall, Upper Saddle River, 2001.

Digraph Embedding on $\mathbf{T}_h$

Ardeshir Dolati *

Department of Mathematics, Shahed University, PO Box: 18151-159, Tehran, Iran.

Key words: graph embedding, upward embedding, spherical diraphs, horizontal torus.

---

abstract An upward embedding of a digraph (directed graph) on the plane or a surface is an embedding of its underlying graph so that all directed edges are monotonic and point to a fixed direction. Such embedding in some literature is called upward drawing without crossing of edges. Upward embedding testing on the plane and sphere are NP-Complete problems (cf. [6, 8]). In this paper we study the problem of upward embedding of digraphs on the horizontal torus which we refer to it by $\mathbf{T}_h$. We shall present a characterization of all digraphs that admit upward embedding on $\mathbf{T}_h$. We also show that it is not possible to find a polynomial time algorithm for upward embedding testing of a given digraph on $\mathbf{T}_h$.

1. Introduction

An upward embedding of a digraph D on an embedded surface S is an embedding of its underlying graph on the surface such that all arcs are represented by monotonic curves that point to a fixed direction. A necessary condition for a digraph to have an upward embedding on a surface is that it has no directed cycle–it is acyclic. In this paper we deal with upward embedding on a special embedding of ring torus which we call it horizontal torus and refer to it by $\mathbf{T}_h$.

There are major differences between graph embedding and upward embedding of digraphs. For instance, all genus one orientable surfaces are topologically homeomorphic to a ring torus, which in turn, from the point of view of graph embedding is equivalent to $\mathbf{T}_h$. But a digraph with an underlying graph with genus one, may have an upward embedding on the vertical torus, and may fail to have an upward embedding on the horizontal torus (cf., [4]). While the question that whether an undirected graph has an embedding on a fixed surface has a polynomial time algorithm [5, 10], there exist polynomial time algorithms for upward embedding testing of some special cases such as three connected [1], single source [2, 9], and outerplanar [11] on the plane. Also there exists a polynomial time algorithm for upward embedding testing of three connected single source digraphs [3] on the sphere. However, in general, upward embedding testing on the plane and also on the sphere is NP-Complete [6, 8].

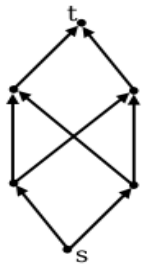


Fig. 1. An SNP-digraph

There is a polynomial time algorithm to decide whether a single source and single sink digraph has an upward embedding on $\mathbf{T}_h$ [4]. In this paper we shall present a characterization of all digraphs that admit upward embedding on $\mathbf{T}_h$. We also show that it is not possible to find a polynomial time algorithm for upward embedding testing of a given digraph on $\mathbf{T}_h$.

2. Main results

In this section after introducing some definitions and notations we present the main results. Here in this paper by equivalence relation $\mathbf{R}$ and horizontal torus $\mathbf{T}_h$ we mean those defined in [4]. By a digraph D we mean a pair $D = (V, A)$ of vertices V, and arcs A. A *source* of D is a vertex with no incoming arc. A *sink* of D is a vertex with no outgoing arc. By *SNP-digraph* we mean a digraph that has an upward embedding on the sphere but it has no upward embedding on the plane. A single source and single sink SNP-digraph is depicted in Figure 1

Dolati et al. [4] showed that an acyclic digraph has an upward embedding on $\mathbf{T}_h$ if it has exactly one source, exactly one sink and the underlying graph of the subdigraphs induced on equivalence classes of its arcs with respect to the relation $\mathbf{R}$ are planar so that at most one of them is an SNP-digraph.

In the following we show that by adding new arcs, if necessary, any digraph that admits upward embedding on $\mathbf{T}_h$ can be extended to an acyclic digraph which satisfies the above conditions.

Theorem 1. *A digraph has an upward embedding on $\mathbf{T}_h$ if and only if by adding new arcs, if necessary, it can be extended to an acyclic single source and single sink digraph whose subdigraphs induced on the equivalence classes of its arcs with respect to $\mathbf{R}$ are planar and at most one of them is an SNP-digraph.*

Now we want to show that it is not possible to find a polynomial time algorithm for upward embedding testing of a given digraph on $\mathbf{T}_h$. To this end we define the *source-in-graph* of a digraph D which we refer to it by $SI(D)$, as follow:

suppose that $D = (V, A)$ is a digraph. Let $\{s_{i_1}, \ldots, s_{i_m}\}$ be the set of its sources

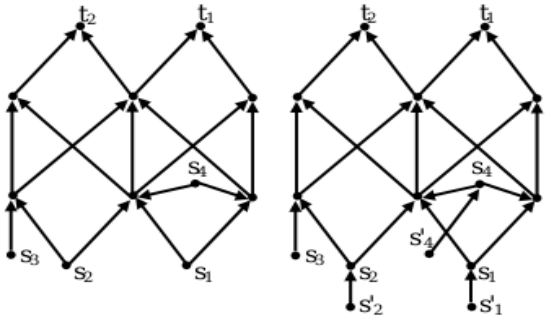


Fig. 2. A digraph D and its $SI(D)$

whose outgoing arcs are more than one . To build the $SI(D)$ from D, we add the set of vertices $\{s'_{i_1}, \ldots, s'_{i_m}\}$ and the set of arcs $\{(s'_{i_j}, s_{i_j})|j = 1, \ldots, m\}$ to it (see Figure 2).

By Theorem 1 we solve one of the open problems that presented in [4], by the following we solve another open problem about upward embedding on $\mathbf{T}_h$ in the same reference.

Theorem 2. *Suppose that D is a digraph and D' is a single source and single sink SNP-digraph whose source and sink are s' and t', respectively. Let S and T be the set of sources and the set of sinks of $SI(D)$ respectively. D has an upward embedding on the sphere if and only if there exist $s' \in S$ and $t' \in T$ so that the resulting digraph from identifying sources s and s' and identifying sinks t and t' of D' and $SI(D)$ has an upward embedding on $\mathbf{T}_h$.*

Corollary 1. *It is not possible to find a polynomial time algorithm for upward embedding testing of a given digraph on $\mathbf{T}_h$.*

References

[1] **Bertolazzi P., Di Battista G., Liotta G.**, and **Mannino C.,** (1994) Upward drawing of triconnected digraphs, *Algorithmica*, 12(6): pp. 476-497.

[2] **Bertolazzi P., Di Battista G., Mannino C.**, and **Tamassia R.,** (1998) Optimal upward planarity testing of single source digraphs, *SIAM J. Comput.* 27(1): pp. 132-169.

[3] **Dolati A., Hashemi S.M.,** (2008) On the sphericity testing of single source digraphs, *Discrete Mathematics*, 308 (11) pp. 2175-2181.

[4] **Dolati A., Hashemi S.M.,** and **Khosravani M.,** (2008) On the Upward Embedding on the Torus, *Rocky Mountain Journal of Mathematics* , 38 (1) pp. 107-121.

[5] **Filotti I.S., Miller G.L.**, and **Reif J.,** (1979) On determining the genus of a graph in $O(v^{O(g)})$ steps, in *Proceeding of the 11th Annual Symposium on Theory of Computing, ACM Press, New York*, pp. 27-37.

[6] **Garg A., Tamassia R.,** (1994) On the Computational Complexity of Upward and Rectilinear Planarity Testing (Extended Abstract), *R. Tamassia and I.C. Tollis(eds.), Graph Drawing '94, Lecture Notes in Computer Science*,Vol. 894, Springer, Berlin, pp. 286-297.

[7] **Hashemi S. M.,** (2001) Digraph Embedding, *Discrete Mathematics*, **233** pp. 321-328.

[8] **Hashemi S. M., Kisielewicz A.,** and **Rival I.,** (1998) The Complexity of Upward Drawings on Spheres, *Order*, **14**, pp. 327-363.

[9] **Hutton M.** and **Lubiw A.,** (1993) Upward drawing of single source acyclic digraph, in: it W. T. Trotter(ed.) Planar Graphs(DIMACS, Series in Discrete Mathematics and Theoretical Computer Science), Vol.9: pp. 41-54.

[10] **Mohar B.,** (1999) A linear time algorithm for embedding graphs on an arbitrary surfaces, *SIAM J. Discrete Math.* 12(1): pp. 6-26.

[11] **Papakostas A.,** (1994) Upward planarity testing of outerplanar dags, in: *R.Tamassia and I. C. Tollis(eds.), Graph Drawing '94, Lecture Notes in Computer Science 894, Springer*, pp. 298-306.

Network design

Room B, Session 1

Tuesday 13, 10:30-12:00

An Application of Network Design with Orientation Constrains

Alberto Caprara [a,*] Emiliano Traversi [a] Joerg Schweizer [b]

[a] *DEIS, Università di Bologna*

[b] *DISTART-Trasporti, Università di Bologna*

Key words: Network Design, Graph Orientation, Integer Linear Programming, Benders Decomposition, Personal Rapid Transit

1. Abstract

We address the problem of orienting the edges of an undirected graph so as to minimize the sum of the distances between a given set of origin-destination pairs in the resulting directed graph. The problem originates from the design of Personal Rapid Transit (PRT) networks. We consider an Integer Linear Programming (ILP) formulation with variables associated with the orientation of the arcs, and variables associated with the arcs in the paths in the directed graph between origin-destination pairs. Given that the direct solution of this natural formulation is impractical even for small instances, we propose a branch-and-cut approach based on Benders decomposition, reporting preliminary experimental results on arising from PRT networks.

2. Motivation and previous work

This work is motivated by the requirements to design optimal, large scale Personal Rapid Transit (PRT) networks for entire urban areas. PRT is an innovative type of public transport [1], with the first system planned to operate in public by the end of 2008 at the new terminal 5 of London-Heathrow airport. PRT is composed of a fleet of fully-automated and electrically-driven vehicles for up to 6 passengers, running on a dedicated network of one-way guide-ways with small dimensions. Similar to Taxis, PRT vehicles are available on-demand and 24 hours a day. The access stations are off-line, ensuring that all vehicles can reach their pre-programmed destinations without transfers or intermediate stops. PRT is seen as a truly sustainable urban mobility alternative that offers high-quality, emission-free and low energy-usage transportation which is accessible to and affordable for all social groups. This

is why PRT has been chosen as the exclusive transport system within the sustainable, completely energy self-sufficient city of the Masdar ("The Masdar-Initiative", Dubai, United Emirates), covering a 5x5 kilometers area with approximately 100 stations and 40 kilometers of guideways. Capacity limits of PRT systems is a crucial issue and is vital to the feasibility of large-scale networks. In principle, exceeding of capacity limits can be avoided in three ways: (i) by reducing headways between vehicles, (ii) by an intelligent, congestion-avoiding vehicle routing, (iii) by an network that is optimized for a-priory known trip demand patterns. The present work is concerned with the last option, defining models in order to find an optimized layout for a non trivial PRT network: starting from a network of initially undirected links, travel costs and a demand matrix between origin- and destination-nodes, the proposed method will orient all the links so as to minimize the travel costs in the resulting directed network.

Due to the obvious similarities between PRT and Automated Guided Vehicle systems (AGV), many previous works originate from AGV applications: Kaspi et al. [4] and Langevin et al. [5] solved instances taken from the AGV systems, using ILP models for the capacitated version and solving them by branch and bound. However, this approach becomes impractical when one considers dense networks with a large number of origin-destination (OD) pairs. Johnson and Pieroni [6] mentioned a branch-and-price approach for solving the uncapacitated version, but without giving any detailed description of the method.

3. Problem formulation and complexity

Let $G = (V, E)$ be an undirected graph, with a length ℓ_e and a capacity $c_e = c_{i,j}$ associated with each edge $e = \{i, j\} \in E$. Moreover, let $R \subseteq \{(i, j) : i, j \in V, i \neq j\}$ be a set of origin-destination pairs, with a demand d_r associated with each origin-destination pair $r = (s_r, t_r) \in R$. We let an *orientation* of G be a directed graph $D = (V, A)$ such that each arc $(i, j) \in A$ corresponds to an edge $\{i, j\} \in E$ and, for each edge $\{i, j\} \in E$, at most one of the arcs (i, j) and (j, i) is in A. We address the problem of finding an orientation D of G, along with a path in D joining each source-destination pair, so as to minimize the weighted sum of the lengths of these paths, where the weight of the path joining each origin-destination pair r is equal to its demand d_r, and the length of each arc $a = (i, j)$ is equal to $\ell_a := \ell_{i,j}$. In the capacitated version, we have the additional constraint that, for each arc $a = (i, j)$ of D, the overall demand of the origin-destination pairs whose path uses arc a does not exceed the arc capacity $c_a := c_{i,j}$. Note that, if the capacity constraint is not imposed, the problem simply calls for an orientation of the edges of G so that the weighted sum of the shortest-path distances between origin-destination pairs in the resulting directed graph D is minimized.

Although the problem has been addressed previously in the literature, we could not find any explicit statement about its complexity. First of all, note that, in case the capacity constraint is imposed, even finding a feasible solution to the problem is

easily seen to be difficult.

Proposition 1 *Testing if the problem considered has a feasible solution is NP-complete.*

Proof. In case $c_e = 1$ for $e \in E$, the problem has a solution if and only if G contains $|R|$ edge-disjoint paths, one from s_r to t_r for $r \in R$. This is well known to be NP-complete. □

On the other hand, in case the capacity constraint is not imposed, finding a feasible solution is easy, though not entirely trivial.

Proposition 2 *([2]) In case the capacity constraint is not imposed, testing if the problem considered has a feasible solution can be done in linear time.*

Proof. Without capacity constraint, the problem has a solution if and only if there exists an orientation of the edges of G such that, for $r \in R$, there exists a directed path from s_r to t_r. This can be tested in linear time by the algorithm in [2]. □

(The algorithm in [2] can also be applied to a mixed graph, in which some of the edges are already oriented.) The above results, based on well-known facts, leave open the complexity of the problem without capacity constraints. This is easily settled by using an old (and not-so-well-known) result by [3].

Proposition 3 *The problem considered is NP-hard even in case the capacity constraint is not imposed.*

Proof. In [3], it is shown that the following problem is NP-complete: given $G = (V, E)$, find an orientation D of G of diameter 2, i.e., such that each node can be reached from each other node by a path with at most two arcs. Given an instance of this problem, we define the instance of our problem (without the capacity constraint) on the same G in which $R := \{(i, j) : i, j \in V, i \neq j\}$, i.e., every ordered pair of nodes is an origin-destination pair, all demands $d_r := 1$ and all edge lengths $\ell_e := 1$. Note that, for each of the $|V|(|V| - 1)/2$ node pairs (i, j), considering the two origin-destination pairs (i, j) and (j, i), in every orientation D of G one of the paths will have weight 1, whereas the other one will have weight at least 2. This proves that the optimal value of our problem is at least $3/2|V|(|V| - 1)$. Moreover, the optimal value is exactly $3/2|V|(|V|-1)$ if and only if there exists an orientation of diameter 2, which shows that by finding an optimal solution to our problem we can solve the problem of [3]. □

The natural ILP formulation of our problem is the following. For convenience, let $\overline{A}$ denote the set of the possible arcs arising from orientations of the edges in E. Moreover, for a node $i \in V$, let $\delta^+(i)$ and $\delta^-(i)$ denote, respectively, the set of arcs in $\overline{A}$ exiting from and entering in V. The ILP formulation contains binary variables $x_a \equiv x_{i,j}$, equal to one if the arc $a \equiv (i, j) \in \overline{A}$ is present in D, i.e., if edge $\{i, j\}$ is oriented from node i to node j, and binary variables $y^r_a \equiv y^r_{i,j}$, equal to one if the path joining origin s_r to destination t_r uses arc $a \equiv (i, j) \in \overline{A}$. The model reads:

$$\min \sum_{r \in R} \sum_{a \in \overline{A}} d^r \ell_a y_a^r, \tag{1}$$

$$x_{i,j} + x_{j,i} \leq 1, \qquad \forall \{i,j\} \in E, \tag{2}$$

$$\sum_{a \in \delta^+(i)} y_a^r - \sum_{a \in \delta^-(i)} y_a^r = \begin{cases} 1, & \text{if } i = s_r \\ -1, & \text{if } i = t_r \\ 0, & \text{otherwise} \end{cases}, \forall i \in V, \forall r = (s_r, t_r) \in R, \tag{3}$$

$$y_{i,j}^r \leq x_{i,j}, \qquad \forall (i,j) \in \overline{A}, \forall r \in R, \tag{4}$$

$$\sum_{r \in R} d^r y_a^r \leq c_a, \qquad \forall a \in A, \tag{5}$$

$$x_a, y_a^r \in \{0,1\}, \qquad \forall a \in A, \forall r \in R. \tag{6}$$

Constraints (2) impose that D is an orientation of G. Equations (3) guarantee that the arcs a with $y_a^r = 1$ define a path from s_r to t_r in D, whereas inequalities (4) link the y and the x variables. Finally, the capacity constraints (5) are not present in the uncapacitated version.

4. Solution approach and preliminary results

Even without the capacity constraints (5), the direct solution of ILP (1)-(6) by a general-purpose ILP solver quickly becomes impractical as the size of G grows. On the other hand, in the uncapacitated version, if constraints (4) are removed, the problem decomposes into $|R| + 1$ independent subproblems. Based on this, in our approach we solve the Linear Programming (LP) relaxation of the ILP by a Benders decomposition approach, with a Master Problem with the x variables along with auxiliary variables β_r expressing the weighted length of the path from s_r to t_r in D, the objective function being $\sum_{r \in R} \beta_r$. The constraints of the Master Problem are (2), along with *optimality constraints* of the form $\beta_r \geq a^T x + b$ and *feasibility constraints* of the form $a^T x \geq b$ that are originated from the solution of the Slave Problems. The latter, one for each origin-destination pair $r \in R$, are defined by objective function (1) (without the summation over r) and constraints (3) and (4), the value of the x variables being fixed by the current solution of the Master Problem. Rather than adding Benders' cuts in a standard way, we add *Pareto-optimal* cuts using the procedure defined by Magnanti and Wong [8].

In our preliminary experimental results we considered realistic instances of the uncapacitated version of the problem, associated with a grid network in which 50% of the nodes are stations, and each ordered pair of stations is an origin-destination pair in R. Moreover, the edge lengths ℓ_e are uniformly distributed in $[0.5, 1.5]$ and the demands d_r are uniformly distributed in $[0, 50]$. In the table above, we compare the times to solve the LP relaxation (1)-(6) by a direct approach as a single LP (column *Direct*) and by our approach based on Benders decomposition (*Bend*), imposing a time limit of one hour. We also report the optimal LP value (*LP val*) and the time required by the linear-time algorithm by [2] (*Tarj*), that finds a feasible

Instance	$\|V\|/\|R\|$	Direct	Bend	LP val	Tarj	Heur
Grid5x5	25/156	1	4	10563.605	0	13658.12
Grid6x5	30/210	3	7	20417.285	1	25649.4
Grid6x6	36/306	9	11	32241.3	2	41978.9
Grid7x6	42/420	24	23	47637.86	3	67605.18
Grid7x7	49/600	66	48	63329.925	5	89069.78
Grid8x7	56/756	143	107	94863.065	10	136386.09
Grid8x8	64/992	290	167	136226.865	19	191177.26
Grid9x8	72/1260	534	374	165318.37	27	252931.22
Grid9x9	81/1640	970	561	211137.705	42	314165.59
Grid10x9	90/1980	2980	895	28515.275	52	408750.55
Grid10x10	100/2450	Tlim	1064	371514.9	83	559278.14
Grid11x10	110/2970	Tlim	3661	469287.495	139	800488.2

solution without taking into account the objective function, along with the associated heuristic value (*Heur*). The table shows the faster increase in the LP solution time taken by the direct approach with respect to ours, and the fact that the heuristic solution quality is not terribly bad (considering that this is essentially a random feasible solution), with a gap with respect to the LP value of about 40%.

In the full paper, we will report results on larger instances, considering heuristic and exact enumerative algorithms based on the LP relaxation.

Together with our model we are developing a PRT micro-simulator with the aim to test and compare the effectiveness of our approach using dynamic vehicle models. Our future work will be focused on (i) collecting and solving real-world PRT instances; (ii) solving the problem with capacity constraints, still by Benders decomposition by using additional auxiliary variables; and (iii) solving the problem with additional constraints that may arise in PRT applications, such as limits on the number of arcs that can be present in D.

References

[1] Anderson, J.E. et al. (1998). Special issue: emerging systems for public transportation. Journal of Advanced Transportation, 32, 1-128.

[2] Chung, F.R.K., Garey, M.R., Tarjan, R.E. (1985). Strongly connected orientations of mixed multigraphs. Networks, 15, 477-484.

[3] Chvatal, V., Thomassen, C. (1978). Distances in orientations of graphs. Journal of Combinatorial Theory Ser. B, 24, 61-75.

[4] Kaspi, M. and Tanchoco, J.M.A. (1990). Optimal flow path design of unidirectional AGV systems. International Journal of Production Research, 28, 1023-1030.

[5] Langevin, A., Riopel, D., Savard G., Bachmann, R. (2004). A multy-comodity network desing approach fo automated guided vehicle systems. INFOR, 2, 113-123.

[6] Johnson, E.L., Pieroni A. (1983). A linear programming approach to the optimum network orientation problem. Presented at NETFLOW 83: International Workshop on Network Flow Optimization Theory and Practice, Pisa, Italy.

[7] Benders, J.F. (1962). Partitioning procedures for solving mixed variables programming problems. Num. Math, 4, 238-252.

[8] Magnanti, T.L., Mireault, P., Wong, R.T.(1986) Tailoring Benders domposition for uncapacitated network design. Mathematical Programming Study, 26, 112-154

Classbased Detailed Routing in VLSI Design

C. Schulte [a,*] T. Nieberg [a]

[a] *Research Institute for Discrete Mathematics*
University of Bonn, Lennéstr. 2, D-53113 Bonn

Key words: Routing, VLSI, Shortest Path

1. Routing in VLSI Design

Modern, highly complex integrated circuits cannot be designed without the use of methods of discrete mathematics: tools based on efficient algorithms are needed to cope with the ever more demanding requirements of a highly automated design process [1]. VLSI (very large-scale integrated) design combines many classical combinatorial optimization problems with practical applications, and usually huge instance sizes.

In the talk, we look at the VLSI routing problem, i.e. the task of connecting different points (metal shapes on the chip called pins) within the chip area by wires so that they are electrically equivalent. Traditionally, this problem is solved on an incomplete 3-dimensional grid graph. The graph is obtained from a complete grid structure in the area by removing parts that are reserved for internal circuit structures, power supply, or already wired parts. These removed areas are called blockages as they have to be avoided by the wires placed during routing. Pins and blockages of such a chip are placed so that they are aligned to this grid structure, and wiring follows the edges between adjacent grid vertices. The grid pitch, i.e. the spacing between two parallel lines, is chosen so that specific rules (called design rules) regarding minimum distance and spacing between wires are satisfied.

Next to this grid graph, an instance of the routing problem consists of a list of nets. A net is a collection of pins that have to be connected by metal wiring. Different nets have to be connected by disjoint paths. The problem at hand thus amounts to finding vertex disjoint Steiner trees for each net. Due to the size of actual instances—a grid graph with more than 10^{11} vertices, millions of nets—this problem is decomposed into smaller problems, that are then solved sequentially. Each net is divided into two-point connections (possibly by adding Steiner points), and then solved by constructing shortest paths between them successively. These resulting shortest-path problems on a grid graph are computationally tractable also from a practical perspective by specialized versions of Dijkstra's algorithm and sophisticated data structures [2].

(a) (b) (c)

Fig. 1. Examples of minimum distance and minimum length violations: In (a) the distance between a wire and pin is not sufficient. In (b) and (c) we have two incident edges which both do not have the required length. To achieve a valid wiring at least one of the edges must be long enough.

However, in recent technologies, blockages and pins of circuits, some of which need to be connected by routing, have become a lot smaller than the grid pitch. As a result, pins that form source and target of a path are no longer alligned with respect to a possible routing grid; we call these *offgrid*. Adjusting the routing grid by making it finer results in two major problems: the size of the grid itself would become intractable in memory, and even more important, the minimum distance and spacing rules avoided by using a routing grid with 'built-in' sufficient spacing would have to be taken into account explicitely. This would result in an unacceptable increase in runtime.

1.1 Offgrid-Pinaccess

A tractable solution to the routing problem with offgrid sources and targets, is to still use the grid graph for the larger distances to be covered, and to locally construct small paths from the respective pins to nearby grid points. However, simply connecting a pin to the nearest point(s) on the grid-graph usually results in infeasible wiring due to the restrictions on minimum length and spacing for the wiring (see Figure 1).

Therefore, we developed an algorithm that computes simple paths from pins to grid points not violating any design rules. The main idea behind this approach is to first identify and remove parts of the pin where starting a wire would cause a violation with the pin itself regardless of its length. Then, from the remaining parts of the pin we construct paths in each direction respecting the minimum lengths in order to avoid design rule violations until we reach an ongrid point.

After deleting the paths that result in minimum distance errors to other wires already present or blockages, the endpoints of the remaining paths are passed as source resp. target points to the ongrid path search. When an ongrid path between such points has been found, we simply have to look up the corresponding offgrid paths and append one of them.

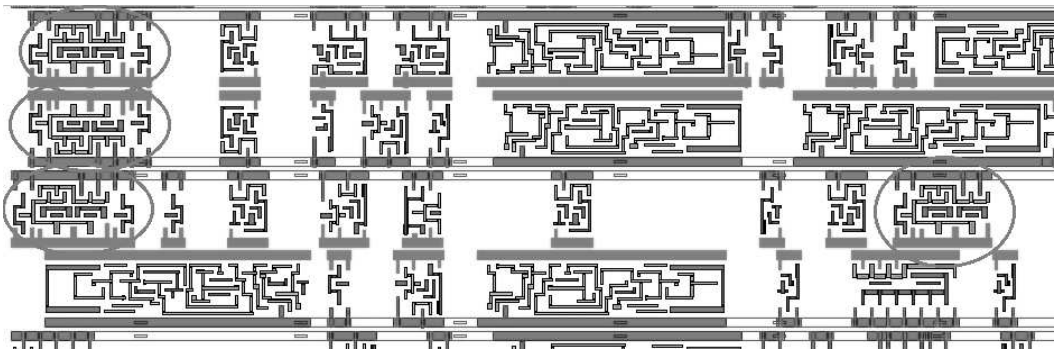


Fig. 2. Circuits placed on the chip area. Circuits with equal geometric shapes are highlighted, these are combined into the same equivalence class. Note that some circuits of the same class are mirrored (on the left).

2. Equivalence Classes of Circuits

Generally speaking, an integrated circuit maps a boolean function to hardware. As such, it consists mainly of transistors that have been placed on the chip area in earlier steps of the VLSI design process. A single building block that maps a simple boolean function by some transistors is called circuit, and in other words, the routing problem has to connect the pins of these smaller circuits. A circuit can be seen as a collection of pins and blockage that is placed within close proximity on the chip area.

Although we have millions of circuits placed on huge chips, each of them containing several pins that have to be connected, there actually are only a few thousand different prototype circuits. As a result, the same configuration of pins and blockages can be found on many different locations on the chip area, possibly rotated or mirrored (see figure 2 for an example). Since also the non-circuit blockages, e.g. resulting from the power supply, are very uniformly placed, we are able to exploit this local redundancy by collecting geometrically equal areas and circuits into equivalency classes.

The benefit of these classes is that we now do not have to compute offgrid paths for millions of pins separately anymore. Instead it now suffices to work on the pins of only one representative circuit of each class. All paths for pins of other circuits can be easily deduced from the already constructed ones. This of course results in runtime improvents and becomes absolutely necessary when applying sophisticated and time consuming methods to construct offgrid paths.

On the other hand, these classes have to be computed as well. This poses a new problem, namely how to collect the circuits into classes, and we designed a sweepline approach to do so efficiently.

With this approach, there are several trade-offs involved:

- Computing large classes results in fewer offgrid paths to be constructed, but

many of these paths become infeasible as some blockage information has to be omitted to create these large classes. Also, the problem on deciding which blockage information to include and to exlude in the creation of the classes has to be solved.

- Computing fine-grained classes, e.g. by taking all blockage information into account, results in a high number of classes and thus in a high number of offgrid paths to be constructed.

3. Conclusions

In the talk, we discuss several issues involved in gridless detailed routing, the interface between gridded and gridless approaches, and provide solutions to avoid many of the problems posed in the context of computer aided design of the latest chips. We especially focus on the trade-off given by first grouping similar structures into classes in order to avoid unnecessary computations during pinaccess versus the cost of computing the classes and coping with incomplete information.

References

[1] B. Korte, D. Rautenbach, J. Vygen. *BonnTools: Mathematical innovation for layout and timing closure of systems on a chip*. Proc. of the IEEE 95 (2007), p. 555–572.

[2] S. Peyer. *Shortest Paths and Steiner Trees in VLSI Routing*. Dissertation, University of Bonn (2007).

A Simple 3-Approximation of Minimum Manhattan Networks

Bernhard Fuchs [a,*] and Anna Schulze [b]

[a] *Abteilung Algorithmik*
TU Braunschweig

[b] *Zentrum für Angewandte Informatik Köln,*
Universität Köln

Key words: minimum Manhattan networks, approximation algorithms, rectilinear routing, VLSI design

1. Introduction

Connecting a given set of points with minimum total length is a key problem in VLSI design. The wires are allowed to run in two perpendicular directions. Minimum Steiner trees minimize the total wire length. Manhattan networks impose an additional constraint. In contrast to Steiner trees they must contain a *shortest* path between each pair of points. Given a set P of n points in the plane, a *Manhattan network* of P is a network that contains a rectilinear shortest path between every pair of points of P. A *minimum Manhattan network* is a Manhattan network of minimum total length. See Figure 1 (a) and (b) for an example.

It is unknown whether it is NP-hard to construct a minimum Manhattan network. The best approximations published so far are a combinatorial 3-approximation algorithm in time $O(nlogn)$ by Benkert et al. [1], and an LP-based 2-approximation algorithm by Chepoi et al. [2]. Kato et al. [4] proposed a 2-approximation with running time $O(n^3)$, however the proof of the correctness seems to be incomplete [1]. Seibert and Unger [5] presented an approximation algorithm and claimed that it yields a 1.5-approximation. As remarked by Chepoi et al. [2] both the description of the algorithm and the performance guarantee are somewhat incomplete and not fully understandable.

We present a new combinatorial 3-approximation for this problem in time $O(nlogn)$. In contrast to the approximation of Benkert et al. [1] our algorithm and also the analysis of the approximation ratio is quite easy. We will discuss similarities and differences below.

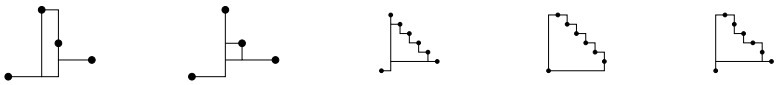


Fig. 1. (a) A Manhattan network. (b) A minimum Manhattan network. (c)-(e) Different staircase boundaries for the same staircase.

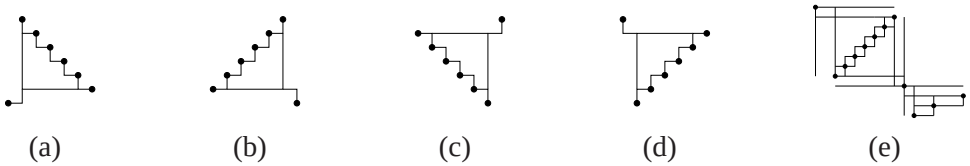


Fig. 2. (a) to (d) The four different cases of staircases. (e) Edges inserted by the sweep steps.

2. Definitions

Each pair of points p and q spans a unique closed axis-parallel rectangle $R(p, q)$ with p and q as corners. We call two points $p, q \in P$ (w. l. o. g. $p_x \leq q_x$ and $p_y \leq q_y$) *x-neighbored* if there is no further point r with $p_x < r_x < q_x$ and y-neighbored if there is no further point r with $p_y < r_y < q_y$. Obviously for two x-neighbored points p and q a minimum Manhattan networks needs to contain line segments of length $|p_y - q_y|$.

Almost all approximation algorithms for minimum Manhattan networks use staircases, but the definition of a staircase is not standardized. To get a clearer definition we define only one of four symmetric cases of a staircase shown in Figure 2. We define the staircase type as shown in Figure 2 (a).

Definition 1 *A* staircase *consists of a sequence of points $(v_1, \ldots, v_k)$ and two* base points b^x, b^y. *For each sequence point v_i, $i = 1, \ldots, k$, the x-base point b^x is the x-neighbored point in the third quadrant of v_i. The y-base point b^y is the y-neighbored point in the third quadrant of v_i. Two points belong to the same staircase if they have the same base points.*

Our algorithm partitions the global Manhattan network problem into disjoint local Manhattan network problems for staircases by inserting edges which separate the staircases of each other. That is, we construct a boundary for each staircase which is defined as follows:

Definition 2 *A* staircase boundary *for a sequence $(v_1, \ldots, v_k)$ of a staircase with base points b^x and b^y is defined in the following way*: *For any consecutive sequence points v_i and v_{i+1}, $1 \leq i \leq k-1$, the staircase boundary contains a shortest path between the two points. Furthermore, the staircase boundary contains a shortest path between v_1 and b^x and between v_k and b^y.*
The area contained in the staircase boundary is called staircase area.

The boundary of a staircase is not unique. See Figures 1 (c) to (e) for examples of staircase boundaries.

3. A 3-Approximation of Minimum Manhattan Networks

Our 3-approximation algorithm for minimum Manhattan networks proceeds in two steps. In the first step we compute a basic set of edges in each of the two dimensions. These edges ensure that only sequence points of staircases remain unconnected to the appropriate base points and that the edges constitute a staircase boundary with staircase area of size at most the one of the staircase area defined by edges of a minimum Manhattan network. In the second part we compute Manhattan networks for the staircases. The general approach to partition the problem in a set of Manhattan network problems for staircases is used by all combinatorial approaches (see for example [1], [3] or [4]). We now describe our approach in more detail. We first examine the points from bottom to top. For two y-neighbored points p and q considered by this sweep we insert the horizontal boundary edges of the rectangle $R(p, q)$ into our network. Afterwards we perform an analogous sweep from left to right. See Figure 2 (e) for an example of such a sweep in the two directions. The up to now identified edges contain a boundary for each staircase. We now identify all staircases and compute for each staircase a Manhattan network. On that behalf we use the standard 2-approximation stated for example by Gudmundsson et al. [3]. Altogether, we obtain a Manhattan network for the input points. See Algorithm 1 for a detailed description.

Algorithm 1 MANHATTAN NETWORK APPROXIMATION

Require: A set $P \subseteq \mathbb{R}^2$ of points.

1: Set $CR = \emptyset$ and $MN = \emptyset$.
2: Sweep over the points of P bottom-up. Let p be the currently considered point and q be the previously processed point. Add to CR the horizontal edges of $R(p, q)$.
3: Sweep over the points of P from left to right. Let p be the currently considered point and q be the previously processed point. Add to CR the vertical edges of $R(p, q)$.
4: **for all** Staircases SC **do**
5: Let MSC be a Manhattan network of the staircase SC given the staircase boundary.
6: Set $MN = MN \cup MSC$.
7: **end for**
8: **return** $MN \cup CR$.

The difference to the approach of Benkert et al. [1] lies mainly in the first step. Whereas we include for pairs of x-neighbored points p and q both horizontal edges of $R(p, q)$ into our network, they first include only one of the two edges. Unfortunately, these edges do not guarantee a partition into two disjoint areas for which they can bound the edge length to be inserted separately. Thus, they have to insert further edges. This complicates both the analysis and the algorithm in comparison to our approach.

Our scan procedure yields a 3-approximation outside of staircases and minimizes the staircase areas. Thus, together with the standard 2-approximation for staircases we get a 3-approximation altogether. The analysis of this algorithm is tight. Altogether, we get the following theorem:

Theorem 1 *For a point set $P \subseteq \mathbb{R}^2$ the MANHATTAN NETWORK APPROXIMATION algorithm computes a Manhattan network with total length at most 3 times the length of a minimum Manhattan network for P in time $O(nlogn)$.*

References

[1] Benkert, M., Wolff, A., Widmann, F., Shirabe, T.: The Minimum Manhattan Network Problem: Approximations and Exact Solutions. Computational Geometry: Theory and Applications 35(3), 2006, 188-208.

[2] Chepoi, V., Nouioua, K., Vaxès, Y.: A Rounding Algorithm for Approximating Minimum Manhattan Networks. Theoretical Computer Science 390(1), 2008, 56-96.

[3] Gudmundsson, J., Levcopoulos, C., Narasimhan, G.: Approximating a Minimum Manhattan Network. Nordic Journal of Computing 8(2), 2001, 219-232.

[4] Kato, R., Imai, K., Asano, T.: An Improved Algorithm for the Minimum Manhattan Network Problem. Proc. of the 13th International Symposium on Algorithms and Computations, LNCS, vol. 2518, Springer, 2002, 344-356.

[5] Seibert, S., Unger, W.: A 1.5-Approximation of the Minimal Manhattan Network Problem. Proc. 16th International Symposium on Algorithms and Computation, LNCS, vol. 3827, Springer, 2005, 246-255.

Coloring (I)

Room A, Session 2

Tuesday 13, 13:30-15:00

Exact Graph Coloring via Hybrid Approaches

Stefano Gualandi [a,*] Federico Malucelli [a]

[a] *Dipartimento di Elettronica e Informazione, Politecnico di Milano*

Key words: Graph coloring, Constraint Programming, Semidefinite Programming, Column Generation

---

We consider the classical Minimum Graph Coloring Problem (Min-GCP). Given a graph $G = (V, E)$ and an integer k, a k-coloring of the graph G is a mapping $c : V \rightarrow \{1, \ldots, k\}$, s.t. $c(i) \neq c(j), \forall \{i, j\} \in E$; Min-GCP consists in finding the minimum k such that a k-coloring exists. Min-GCP is NP-hard [5].

Constraint Programming (CP) is a natural choice as a method for checking if a k-coloring exists, since constraint propagation can be exploited quite effectively [1]. Standard CP lacks efficient mechanisms to guide the search towards the optimal region, and to derive effective bounds on the minimum k. For this reason, hybrid methods that integrate CP with standard mathematical programming techniques have been recently investigated; amongst them, two promising hybrid approaches are (i) the exploitation of Semidefinite Programming relaxations into Constraint Programming, and (ii) the so–called Constraint Programming-based Column Generation. We have applied both hybrid approaches to the Min-GCP obtaining interesting results and new insights for integrating the SDP-relaxation into the CP-based Column Generation approach.

The idea of using SDP relaxations within a CP approach to combinatorial optimization problems has been introduced in [9] for the stable set and the maximum clique problems. There is a wide literature on SDP relaxations of the Min-GCP (*e.g.*, see [3]). Here we propose to use the solution of the SDP relaxation of Min-GCP in order to devise effective branching rules within the CP code, breaking ties carefully. The SDP relaxation gives for every pair of vertices the likelihood of coloring them with the same color. We observe that in the case of Min-GCP we deal not with binary variables as in [9], but with integer variables. Different rules for choosing the next vertex to color and the color to assign to it are experimented.

The CP-based Column Generation framework has been introduced in [4] for solving a crew scheduling problem, and it was motivated by the need to model complex numerical and logical constraints. The main idea of CP-based Column Generation is that the pricing subproblem is formulated as a Constraint Satisfaction Problem (CSP) and is solved using Constraint Programming. A regular Column Generation formulation of Min-GCP is introduced in [7], where the master is a classical

set partitioning problem, and the pricing is a maximum weighted independent set problem. In our CP-based formulation, the pricing subproblem is to find a maximal independent set with a weight greater than a given threshold τ, corresponding to find a negative reduced cost variable to enter the basis of the master problem. To make CP efficient in solving the pricing subproblem we use two accelerating techniques introduced in [6]: the first technique consists in using a randomized breaking ties strategies into the CP solver, and the second in using an adaptive threshold τ that changes throughout the iterations of Column Generation.

Since CP-based Column Generation solves the linear relaxations of the integer master problem, to solve exactly Min-GCP we need to implement a branch-and-price algorithm. In branch-and-price, we are faced with the problem of devising effective branching rules. For Min-GCP, an effective strategy is to select at every branching node a couple of vertices (i, j) that are not adjacent, and to either force them to take the same color or to take different colors [7]. This is equivalent to formulate two new graph coloring problems: the first resulting from merging vertex i and j into an additional vertex ij and deleting eventual parallel edges, and the second in adding an edge between vertex i and j. This branching rule allows a recursive use of the same CP-based Column Generation algorithm, but to be effective the couple of vertices (i, j) must be chosen carefully. This is the same issue faced with the CP formulation, and that we have solved with the hybrid CP-SDP approach. Therefore, our new idea is to use again the SDP relaxation of Min-GCP to devise an effective branching rule for the branch-and-price algorithm.

For the experimental campaign, we focus on the instances of Min-GCP that are very challenging for existing exact methods, like DSATUR [2], branch-and-cut [8], and regular branch-and-price [7]. In particular these methods spend plenty of computational time in closing the gap between fractional and integer solutions. Among such instances, Mycielski graphs, although of small size, are particularly challenging, and hence have been chosen as the first benchmark set. Computational experience shows that our hybrid CP-SDP approach outperforms known exact methods on Mycielski graphs. The second set of benchmark are the (n, p) random graphs, with n vertices and p expected density; these graphs have bigger size than Mycielski graphs and for most of them the chromatic number is still unknown. On random graphs, the CP-based Column Generation outperforms the hybrid CP-SDP approach, since it is able to fully exploit the linear lower bounds. In particular, with the CP-based Column Generation we were able to close two open instances (DSJC125.9 and DSJC250.9) and to improve the lower bound of other four instances (DSJC125.5, DSJC250.5, DSJC500.9, DSJC1000.9). At the time of writing, the hybrid SDP-CP-based Column Generation is under testing and validation, but preliminary results suggests that this new hybridization approach is indeed interesting.

References

[1] Nicolas Barnier and Pascal Brisset. Graph coloring for air traffic flow management. In Annals of Operation Research, August 2004, vol 130, pp 163–178, Kluwer, 2004.

[2] Daniel Brelaz. New methods to color the vertices of a graph. Commun. ACM, 22(4):251–256, 1979.

[3] Igor Dukanovic and Franz Rendl. Semidefinite programming relaxations for graph coloring and maximal clique problems. Math. Program., 109(2):345–365, 2007.

[4] Torsten Fahle, Ulrich Junker, Stefan E. Karisch, Niklas Kohl, Meinolf Sellmann, and Bo Vaaben. Constraint programming based column generation for crew assignment. J. Heuristics, 8(1):59–81, 2002.

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

[6] Stefano Gualandi. Enhancing CP-based column generation for Integer Programs. PhD thesis, Politecnico di Milano, Italy, 2008.

[7] Anuj Mehrotra and Michael A. Trick. A column generation approach for graph coloring. INFORMS Journal on Computing, 8:344–354, 1996.

[8] Isabel Mendez-Diaz and Paula Zabala. A cutting plane algorithm for graph coloring. Discrete Applied Mathematics, 156:159–179, 2008.

[9] Willem-Jan van Hoeve. Exploiting semidefinite relaxations in constraint programming. Comput. Oper. Res., 33(10):2787–2804, 2006.

Maximum Profit Wavelength Assignment in WDM Rings [1]

Evangelos Bampas [a,*], Aris Pagourtzis [a], and Katerina Potika [a]

[a] *School of Electrical & Computer Engineering, National Technical University of Athens, Greece*

Key words: maximum path coloring, weighted arc coloring, approximation algorithms

1. Introduction

Wavelength Division Multiplexing (WDM) is a dominating technology in contemporary all-optical networking. It allows several connections to be established through the same fiber links, provided that each of the connections uses a different wavelength. A second requirement is that a connection must use the same wavelength from one end to the other in order to avoid the use of wavelength converters which are costly or slow. In practice, the available bandwidth is limited to few dozens, or at most hundreds, wavelengths per fiber and the situation is not expected to change in the near future. It is therefore impossible to serve a large set of communication requests simultaneously. It thus makes sense to consider the problem of satisfying a maximum profit subset of requests, where profits may represent priorities or actual revenues related to requests. In our model, requests are undirected, which corresponds to full-duplex communication. We describe a request by its connection path and its profit, and formulate the problem in graph-theoretic terms as follows:

MAXIMUM PROFIT PATH COLORING PROBLEM (MAXPR-PC)
Input: a graph G, a set of paths $\mathcal{P}$, a profit function $w : \mathcal{P} \to \mathbf{R}$ and a number of available colors k.
Feasible solution: a set of paths $\mathcal{P}' \subseteq \mathcal{P}$ that can be colored with k colors so that no overlapping paths are assigned the same color.
Goal: maximize $\sum_{p \in \mathcal{P}'} w(p)$.

[1] Research supported by PENED 2003 project, cofinanced 75% of public expenditure through EC – European Social Fund, 25% of public expenditure through Ministry of Development – General Secretariat of Research and Technology of Greece and through private sector, under measure 8.3 of Operational Programme "Competitiveness" in the 3rd Community Support Programme. We also acknowledge funding from the National Technical University of Athens, through PEBE 2007 Basic Research Support Programme.

Here we study MAXPR-PC in rings with undirected requests and we present a 2-approximation algorithm.

While the cardinality version of the problem (MAXPC) has been studied by several researchers [1, 3], MAXPR-PC has been considered in rather few papers [4–6]. Both MAXPR-PC and MAXPC are $\mathcal{NP}$-hard even in simple networks such as rings and trees; this can be shown by an immediate reduction from the corresponding color minimization problem (see e.g. [1]).

MAXPR-PC in chains is also known as the "weighted k-coloring of intervals" problem, which can be solved exactly as shown by Carlisle and Lloyd [4]. In [6] an algorithm based on linear programming and randomized rounding with approximation ratio 1.49 for MAXPR-PC in rings is presented. Let us note here that, although the algorithm in [6] achieves a better approximation ratio, the algorithm presented here is purely combinatorial, therefore faster and easier to implement. Li et al. [5] study a version of MAXPR-PC where requests are not routed in advance, that is, an appropriate routing and coloring is sought. They also assume directed requests and edge capacities that must be obeyed and present a 2-approximation algorithm for rings.

2. Match and Replace for MAXPR-PC

In this section we present an algorithm for MAXPR-PC in rings. MAXPR-PC in chains can be solved exactly in $O(km\ log m))$ time, using algorithm [4].

In our algorithm, we employ a popular technique used for rings, namely to choose an edge e and remove it from a ring. We call this algorithm Match and Replace; details are given in Algorithm 2. We denote the profit of a set of paths $\mathcal{P}$ with $w(P) = \sum_{p\in\mathcal{P}} w(p)$. Given a set of paths $\mathcal{P}$, the set of paths in $\mathcal{P}$ that are colored with the same color i is called the *i-th color class* of $\mathcal{P}$; we use $\mathcal{P}(i)$ to abbreviate this notion.

Theorem 1 *Match and Replace is a 2-approximation algorithm.*

Proof. Let OPT be the value of any optimal solution of the ring instance, OPT_c be the value of any optimal solution of the instance constrained to path set $\mathcal{P}_c$ and OPT_e be the value of any optimal solution of the instance constrained to path set $\mathcal{P}_e$. Recall that

$$OPT \leq OPT_c + OPT_e \ . \tag{1}$$

Let SOL_c be the value of the solution obtained in step 2 of the algorithm (chain subinstance solution), and SOL be the value of the final solution. Clearly,

$$SOL = SOL_c + w'(M) \tag{2}$$

where $w'(M)$ is the sum of the weights of the edges that belong to the matching M computed in step 5. The instance $(G - e, \mathcal{P}_c, w)$ is solved optimally in step 2.

Algorithm 2 Match and Replace

1: Pick an arbitrary separation edge e of the ring. Let $\mathcal{P}_e$ be the set of paths that use edge e and $\mathcal{P}_c = \mathcal{P} \setminus \mathcal{P}_e$.
2: Color the instance $(G-e, \mathcal{P}_c, w)$ optimally, using the Carlisle-Lloyd algorithm for MAXPR-PC in chains.
3: Let $\mathcal{P}_c(i)$ be the i-th color class of $\mathcal{P}_c, 1 \leq i \leq k$ (note that some color classes may be empty).
4: Construct a weighted bipartite graph $H = (\mathcal{S} \cup \mathcal{P}_e, E)$, with $\mathcal{S} = \{\mathcal{P}_c(i) : i = 1, \ldots, k\}$. For every pair $(\mathcal{P}_c(i), q) \in \mathcal{S} \times \mathcal{P}_e$, define path set $\mathcal{P}_c(i)^q$ such that $\mathcal{P}_c(i)^q \subseteq \mathcal{P}_c(i)$ and $\forall p \in \mathcal{P}_c(i)^q, p$ and q overlap (that is $\mathcal{P}_c(i)^q$ consists of those paths in $\mathcal{P}_c(i)$ that overlap q). If $w(q) - w(\mathcal{P}_c(i)^q) > 0$ then we add edge $(\mathcal{P}_c(i), q)$ to H with weight $w'(\mathcal{P}_c(i), q) = w(q) - w(\mathcal{P}_c(i)^q)$.
5: Find a maximum weight matching M in H.
6: **for each** edge $(\mathcal{P}_c(i), q) \in M$ **do**
7: uncolor all paths in $\mathcal{P}_c(i)^q$ and color path $q \in \mathcal{P}_e$ with color i.
8: **end for**

Therefore, taking also into account Eq. 2 we have that

$$OPT_c = SOL_c \leq SOL \ . \tag{3}$$

Let $\mathcal{S}_M$ be the subset of $\mathcal{S}$ consisting of $\mathcal{P}_c(i)$'s that are matched by M. Similarly, let $\mathcal{P}_{e,M}$ be the paths in $\mathcal{P}_e$ that participate in M. Finally, let K be the set of the k most profitable paths of $\mathcal{P}_e$. We will now show that

$$OPT_e = w(K) \leq SOL \ . \tag{4}$$

For the sake of analysis we will examine a solution SOL' that Match and Replace would have computed if it had chosen a matching M' of a subgraph H' of H in step 5. Bipartite graph H' has the same node set and the same edge weight function as H, but only a subset of the edges of H, namely for every pair $(\mathcal{P}_c(i), q)$: edge $(\mathcal{P}_c(i), q)$ is in H', if $w(q) - w(\mathcal{P}_c(i)) > 0$ and $q \in K$. Let M' be a maximum matching in H', and let $\mathcal{S}_{M'}$ and $\mathcal{P}_{e,M'}$ be defined analogously for M' as for M. Similar to Eq. 2

$$SOL' = SOL_c + w'(M') \ . \tag{5}$$

Note that $SOL_c = w(\mathcal{S})$ and also that $w'(M') = w(\mathcal{P}_{e,M'}) - \sum_{(P(i),q)\in M'} w(\mathcal{P}_c(i)^q) = w(\mathcal{P}_{e,M'}) - \sum_{(\mathcal{P}_c(i),q)\in M'} [w(\mathcal{P}_c(i)) - w(\mathcal{P}_c(i)^{\neg q})] = w(\mathcal{P}_{e,M'}) - w(\mathcal{S}_{M'}) + \sum_{(\mathcal{P}_c(i),q)\in M'} w(\mathcal{P}_c(i)^{\neg q})$, where $\mathcal{P}_c(i)^{\neg q}$ consists of these paths in $\mathcal{P}_c(i)$ that do not overlap with q. Equation 5 may then be rewritten as follows: $SOL' = w(\mathcal{S} \setminus \mathcal{S}_{M'}) + w(\mathcal{P}_{e,M'}) + \sum_{(\mathcal{P}_c(i),q)\in M'} w(\mathcal{P}_c(i)^{\neg q})$. We observe that $\mathcal{P}_{e,M'} \subseteq K$ and therefore $w(\mathcal{P}_{e,M'}) + w(K \setminus \mathcal{P}_{e,M'}) = w(K)$, so the last sum can be expanded in the following way:

$$SOL' = w(\mathcal{S} \setminus \mathcal{S}_{M'}) + w(K) - w(K \setminus \mathcal{P}_{e,M'}) + \sum_{(\mathcal{P}_c(i),q)\in M'} w(\mathcal{P}_c(i)^{\neg q}) \ . \tag{6}$$

Observe also that for any $\mathcal{P}_c(i) \notin \mathcal{S}_{M'}$ and $q \notin \mathcal{P}_{e,M'}$, there must be no edge

between them in H', hence $w(\mathcal{P}_c(i)) \geq w(q)$. Moreover, $w(\mathcal{S} \setminus \mathcal{S}_{M'})$ and $w(K \setminus \mathcal{P}_{e,M'})$ are sums with the same number of terms because $|K| = |\mathcal{S}| = k$ and $|\mathcal{S}_{M'}| = |\mathcal{P}_{e,M'}|$. These observations imply that $w(\mathcal{S} \setminus \mathcal{S}_{M'}) - w(K \setminus \mathcal{P}_{e,M}) \geq 0$, therefore Eq. 6 yields $SOL' \geq w(K)$. Since H' is a subgraph of H, M' is a matching also for H, probably not a maximum one, therefore $w'(M) \geq w'(M')$ which implies, from Eq. 2 and 5, that $SOL \geq SOL'$. Combining this last inequality with $SOL' \geq w(K)$ we obtain Eq. 4. By Eq. 3 and 4, SOL is an upper bound on both OPT_e and OPT_c, which together with Eq. 1 gives $SOL \geq \frac{OPT}{2}$.

Computing a solution for the chain subinstance takes $O(kmlogm)$. Graph H has $O(m)$ nodes (we assume that $k < m$) and $O(km)$ edges, therefore maximum weighted matching of H takes $O(m^2(k + logm))$ time. Therefore the total time complexity is $O(m^2(k + logm))$. $\Diamond$

References

[1] Wan, P.J., Liu, L.: Maximal throughput in wavelength-routed optical networks. In: Multichannel Optical Networks: Theory and Practice. Volume 46 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, AMS (1998) 15–26

[2] Erlebach, T., Jansen, K.: Maximizing the number of connections in optical tree networks. In: Proceedings of ISAAC '98. LNCS 1533 (1998) 179–188

[3] Nomikos, C., Pagourtzis, A., Zachos, S.: Satisfying a maximum number of pre-routed requests in all-optical rings. Computer Networks **42**(1) (2003) 55–63

[4] Carlisle, M., Lloyd, E.: On the k-coloring of intervals. Discrete Applied Mathematics **59** (1995) 225–235

[5] Li, J., Li, K., Wang, L., Zhao, H.: Maximizing profits of routing in WDM networks. Journal of Combinatorial Optimization **10**(2) (2005) 99–111

[6] Caragiannis, I.: Wavelength management in WDM rings to maximize the number of connections. In: STACS. (2007) 61–72

Cellular Radio Resource Allocation Problem

Andrea Abrardo [a] Paolo Detti [a,*] Gaia Nicosia [b] Andrea Pacifici [c] Mara Servilio [d]

[a] *Università degli Studi di Siena, Dipartimento di Ingegneria dell'Informazione, Siena, Italia*

[b] *Dipartimento di Informatica e Automazione, Università di "Roma Tre", via della Vasca Navale 79, 00146 Roma, Italia*

[c] *Dipartimento di Ingegneria dell'Impresa, Università degli Studi Tor Vergata, Roma, Italia*

[d] *Università degli Studi di L'Aquila, Dipartimento di Informatica, via Vetoio, I-67010 Coppito, L'Aquila, Italia*

Key words: radio resource allocation, network flow, heuristic algorithm.

1. Introduction

In this paper, the problem of allocating radio resources to users in the downlink of an OFDMA telecommunication system in the multi-cell scenario is addressed.

Orthogonal frequency division multiple access (OFDMA), based on multi-carrier technology, has been widely accepted as the most promising radio transmission technology for next generation wireless systems due to its robustness to channel distortions and granular resource allocation capability.

In a multi-carrier system the transmitted bitstream is divided into many different substreams that are sent over many different sub-channels, called sub-carriers. OFDMA systems envisage the assignment of a number of sub-carriers and the relative transmission format to users on the basis of the experimented link quality. Interference phenomena limit the number of users transmitting on the same radio resources, i.e., sub-carriers. The transmission power required by each user to transmit on a given resource depends on the *set of users* assigned to that resource. The objective is to assign radio resources so as to minimize the overall transmission power while providing a given transmission rate to each user (hence, satisfying some fairness criterion among users).

* Corresponding author.

OFDMA supports differentiated quality-of-service through the assignment of a different number of sub-carriers to different users. Moreover, for fixed or portable applications where the radio channels are slowly varying, an intrinsic advantage of OFDMA over other multiple access methods is its capability to exploit the multi-user diversity embedded in diverse frequency-selective channels. In fact, propagation channels are independent for each user and thus the sub-carriers that are in a deep fade for one user may be good ones for another.

Several papers have recently focused on the problem of optimum channel allocation of OFDMA cellular systems, and some of them have also considered the joint scheduling-allocation problem [1–4].

In this work, the problem of allocating sub-carriers in the downlink of an OFDMA system in a multi-cell scenario is addressed. A complexity analysis for general and particular cases is given and solution methods are proposed. In particular, exact and heuristic algorithms based on minimum cost network flow models are provided.

2. Problem statement

The allocation problem we address here can be described as follows. We are given a set of m radio resources, the sub-carriers, and a set of users. The users are partitioned into k *cells* where each cell h contains n_h users, $1 \leq h \leq k$. For each user i, we denote by $c(i)$ the cell of user i. If we set a certain target spectral efficiency η_i for user i, the transmission requirements correspond to a certain number of sub-carriers $r_i = R_i/\eta_i$, where R_i is the transmission rate required by user i, and η_i is set in a such a way that r_i is integer.

In general, users belonging to different cells can share the same sub-carrier, while interference phenomena do not allow two users in the same cell to transmit on the same sub-carrier. However, the power required for the transmission on a given sub-carrier increases with the number of users transmitting on that sub-carrier. More precisely, let S_j be the set of users which are assigned to (i.e., that are transmitting on) the same sub-carrier j. The transmission powers $p_i(j)$ requested by users in S_j on sub-carrier j satisfy the following system.

$$\begin{aligned} & p_i(j) = A_i(j) + \textstyle\sum_{\substack{\ell \in S_j \\ \ell \neq i}} B_i^{c(\ell)}(j) p_\ell(j) && i \in S_j \\ & p_i(j) \geq 0 && i \in S_j \end{aligned} \tag{1}$$

where $A_i(j)$ and $B_i^{c(\ell)}(j)$ are given data taking into account the target signal-interference-ratio, the channel gain of user i on sub-carrier j, and the channel gain between user i and the base station of cell $h \neq c(i)$ on sub-carrier j.

In the following, given the set S_j of users transmitting on sub-carrier j, we denote

by $T_j(S_j)$ the transmission power required by j, i.e.,

$$T_j(S_j) = \sum_{i \in S_j} p_i(j).$$

Note that power $p_i(j)$ increases as the interference term increases and that the interference term depends on the set of users, other than i, which are assigned to the same sub-carrier. Note also that, System (1) could not have a feasible solution. On the other hand, if only user i is assigned sub-carrier j, i.e., the interference term is null, then $p_i(j) = A_i(j)$.

A *feasible radio resource allocation* consists in assigning sub-carriers to users in such a way that (i) for each user i, r_i sub-carriers are assigned to it, (ii) the users in the same cell are not assigned to the same sub-carrier, (iii) given the set S_j of users assigned to sub-carrier j, System (1) has a feasible solution, for all the sub-carriers j. The problem, that we call Cellular Radio Resource Allocation Problem (CRP), consists of determining a feasible radio resource allocation that minimizes the total transmission power, i.e., the sum of the transmission powers required by all the users. Note that, a necessary condition for a CRP instance to be feasible is $m \geq \max\{\max_i\{r_i\}, \max_h\{n_h\}\}$.

We show that CRP is, in its general form, strongly NP-hard. If we are using the expressions of System (1) to compute the transmission powers required, we may prove that CRP is strongly NP-hard even when there are only 3 cells and the sub-carriers are identical (i.e., data do not depend on the specific sub-carrier j). It is worthwhile to note that relaxing those expressions by allowing more general power consumption models, it is possible to show that CRP is strongly NP-hard even when there are only two radio resources.

3. A heuristic approach for CRP

When the transmission power T_j of any subcarrier j can be decomposed in a certain way, illustrated hereafter, CRP can be solved exactly in polynomial time.

Suppose T_j is comprised of a fixed cost part depending on the set of assigned users S_j plus a *convex* variable cost part which depends only on the number $|S_j|$ of users assigned to resource j. In particular, let $f_{ij} \in \mathbb{R}_+$ be the fixed cost of assigning user i to resource j and $g_j(\cdot)$ a convex function representing the variable cost part. Then the transmission power required by resource j, when a set S_j of users is assigned to it, is:

$$T_j(S_j) = g_j(|S_j|) + \sum_{i \in S_j} f_{ij}. \tag{2}$$

When the transmission powers are of the form illustrated in (2), it is possible to formulate CRP as a minimum cost flow problem on a special network G, and hence to efficiently solve it.

Since any feasible flow on G provides an assignment of users to sub-carriers, we can heuristically use the same approach in the general case by approximating the expressions of the transmission powers in System (1) in order to obtain a cost structure satisfying Equation (2). Such an approximation process may produce an assignment which is not necessarily feasible in the original problem (since a set of users assigned to a resource may require negative power values). However, infeasibilities can be dealt with by suitable modifications of the derived assignments.

Preliminary computational experiments on instances with 7 cells 16 resources and 28 users (each requiring 4 resources) show that a heuristic derived by the approach described above is able to obtain quite good solutions in few msec. In fact, the quality of the solutions, from the point of view of the total transmission power, is comparable with that obtained by a branch&bound algorithm based on a MILP formulation after few minutes of computation.

References

[1] A. Abrardo, A. Alessio, M. Moretti, P. Detti, ŞCentralized Radio Resource Allocation for OFDMA Cellular Systems,Ť IEEE ICC 2007, 24-28-th June, Glasgow, Scotland.

[2] S. B. Jeong, S. Kim, H. Lee, ŞData traffic scheduling algorithm for multiuser OFDM system with adaptive modulation considering fairness among users,Ť Computers & Operations Research, vol. 32, pp. 1723-Ű1737, 2005.

[3] C. Wong, R. Cheng, K. Lataief, and R. Murch, ŞMultiuser OFDM with adaptive subcarrier, bit and power allocation,Ť IEEE Journal on Selected Areas in Communications, vol. 17, no. 10, pp. 1747–1758, October 1999.

[4] Y. Zhang and K. Letaief, ŞEnergy-efficient MAC-PHY resource management with guaranted QoS in wireless OFDM networks,Ť in Proc. IEEE ICC 2005, Seoul, Korea.

Graph optimization (I)

Room B, Session 2

Tuesday 13, 13:30-15:00

A Local Approximation Algorithm for Maximum Weight Matching

Tim Nieberg *

Research Institute for Discrete Mathematics
University of Bonn, Lennéstr. 2, D-53113 Bonn

Key words: Maximum Weight Matching, Local Algorithm, Approximation

1. Introduction

In a graph $G = (V, E)$, a subset of edges $M \subset E$ is called a *matching* if no two edges of M share a common vertex. With each edge $e \in E$, we associate a weight $w_e \in \mathbb{R}$, and for a subset $S \in E$, we set $w(S) := \sum_{e \in S} w_e$ as its weight. The aim of the Maximum Weight Matching Problem is to find a matching M^* with maximum weight over all matchings.

An algorithm that for all graphs returns a matching M whose weight $w(M)$ is at most a factor of ρ away from the weight $w(M^*)$ of an optimal matching is said to have an approximation ratio of ρ. We present a local, distributed algorithm that constructs a matching with $(1 - \varepsilon)$ approximation ratio for any $\varepsilon > 0$, and that results in an approach with polylogarithmic expected running time.

For this algorithm, we use the standard local message passing model [3]. The network is an undirected graph $G = (V, E)$, and two nodes $u, v \in V$ in the network are adjacent, i.e. $(u, v) \in E$, whenever there is a bidirectional communication channel connecting u and v. For simplicity, we assume a synchronous communication model: time is divided into rounds, and in each round, every node can send a message to each of its neighbors in G.

Such local algorithms are specifically useful in a large-scale distributed communication network settings such as the internet and wireless sensor networks. They respect the limitations of each node in the network only being able to communicate with its direct neighbors in the network. Solving optimization problems on these graphs by first collecting the topology at a central processing point, computing a solution, and then reporting this solution to the nodes again leads to a high communication overhead, both in terms of time and message size. Moreover, by the time a

* Corresponding author.

centralized solution is computed and reported back, the topology may already have changed.

Given a matching $M \subset E$, we define another matching $S \subset E \setminus M$ to be an *augmentation* (for M). For such an augmentation S, we denote by $M(S) \subset E$ all edges in M, that have a vertex in common with an edge from S. It is easy to see that $(M \setminus M(S)) \cup S$ again forms a matching in G, and we say that this resulting matching is obtained by augmenting M with S. We denote by $\text{gain}_M(S) = w(S) - w(M(S))$ the gain of augmenting M with S. The size of an augmentation is the number of edges contained in S. Considering the set $M(S) \cup S \subset E$, we call an augmentation S *connected* if $M(S) \cup S$ is a single component in G.

For a subset $V' \subset V$ of nodes, we use $G(V')$ to denote the subgraph induced by V' in G. Furthermore, a subset of nodes $I \subset V$ is called independent set if $G(I)$ contains no edges. We call such a set *maximal* if it cannot be extended by additional nodes. Let $\Gamma_r(v)$ be the set of nodes $u \in V$ which have distance at most r from $v \in V$.

2. The l-Augmentation Graph G'

The algorithm presented works on a structure that we call *augmentation graph*, and which is defined and locally constructed as follows.

Definition 1 *The l-augmentation graph $G' = (V', E')$ (of a graph G with respect to a matching M) is defined as the intersection graph of connected augmentations of size at most l in G: The nodes V' of G' are all connected augmentations of size at most l, and two such augmentations are connected by an edge if they have at least one node (from G) in common.*

For each such augmentation, we call the node with the lowest identifier its representative (in G). Note that any node $u \in V$ may represent multiple augmentations from the l-augmentation graph. Every node only needs to construct the connected augmentations it is part of and communicate the augmentations it represents. Therefore, in order to locally construct G', each node $v \in V$ needs to have knowledge about $\Gamma_{O(l)}(v)$ in G which it can obtain in $O(l)$ communication rounds.

Any communication in G' is now mapped to the representatives in G, and passing a message along a single edge in G' takes $O(l)$ rounds in G. We now restrict our attention to this l-augmentation graph $G' = (V', E')$. In total, we have $|V'| = O(n^{2l})$.

3. An Algorithm to Improve a Matching

Starting with an empty matching, we iteratively call Algorithm 3 a certain number of times in order to improve the current matching.

After having constructed the l-augmentation graph based on the matching M given

Algorithm 3 IMPROVEMATCHING

Input: $G = (V, E)$, weights $w_e, e \in E$, matching M in G, $l \in \mathbb{N}$
Output: Matching M' in G

1. Construct l-augmentation graph $G' = (V', E')$
2. $\mathcal{A} := \varnothing$
3. $V^{(1)} := V'$
4. **for** $t := 1$ **to** $\lceil log_2 l^2 n \rceil$ **do**
6. $W := \{v \in V^{(t)} \mid \Gamma_1(v) \cap \{u \in V^{(t)} \mid \text{gain}(u) > 2 \cdot \text{gain}(v)\} = \varnothing\}^{(*)}$
6. Calculate maximal independent set I in $G'(W)$
7. $\mathcal{A} := \mathcal{A} \cup I$
8. $V^{(t+1)} := V^{(t)} \setminus \Gamma_1(I)^{(*)}$
9. **endfor**
10. $M' := M$ augmented by augmentations represented in $\mathcal{A}$

$^{(*)}$ Here, $\Gamma_1(.)$ is taken w.r.t. G'.

as input, the loop (4.–9.) constructs a set $\mathcal{A}$ of augmentations. By construction, the set I that is added to $\mathcal{A}$ each time is an independent set. Therefore, the set $\mathcal{A}$ constructed in Algorithm 3 is an independent set in G', and we can augment M by the local augmentations contained in $\mathcal{A}$ in parallel. We obtain:

Theorem 1 *[4] Let $T_{MIS}(m)$ denote the time needed to locally construct a maximal independent set on a graph with m nodes.*
Then, Algorithm 3 can be realized by a local, distributed approach that requires $O(l + log\,(l^2 n) \cdot T_{MIS}(n^{O(l)}))$ communication rounds.

For the analysis of Algorithm 3, we look at the difference between $w(M)$ and $w(M')$, in other words, the gain of $\mathcal{A}$. In particular, we are interested in the improvement with respect to an optimal solution M^*.

For this overall gain, we can state the next theorem, which follows an idea presented in [1] and that is adapted to the local, distributed setting in [4].

Theorem 2 *Receiving a matching M as input, Algorithm 3 returns a matching M' with*

$$w(M') \geq w(M) + \frac{1}{8l} \cdot \left(\frac{l-1}{l} w(M^*) - w(M) \right),$$

where M^* is a matching of maximum weight in G.

4. A $(1 - \varepsilon)$-Approximation Algorithm

As a corollary to Theorem 2, we would like to point out that, starting with $M = \varnothing$, a single invocation of Algorithm 3 results in a local approach that yields a constant factor approximation of the Maximum Weight Matching problem for any $l > 1$.

Moreover, setting $l = O(1/\varepsilon)$ and invoking Algorithm 3 $O(1/\varepsilon)$ times ($\varepsilon > 0$) results in a local $(1-\varepsilon)$-approximation algorithm:

Theorem 3 *[1,4] Let $l \in \mathbb{N}$. Calling Algorithm 3 $O(l)$ times returns a matching M of weight at least $(1 - O(1/l)) \cdot w(M^*)$.*

The overall number of rounds needed to obtain such a solution is then $O(l^2 + l \cdot log(l^2 n) \cdot T_{\text{MIS}}(n^{O(l)}))$.

5. Conclusions

The presented Algorithm 3 yields a local, distributed algorithm that constructs a $(1-\varepsilon)$-approximation of a Maximum Weight Matching in a graph, and does so in $O(\frac{1}{\varepsilon^2} \cdot log\, n \cdot T_{\text{MIS}}(n^{O(l)}))$ communication rounds. Taking, e.g., the well-known local algorithm of Luby [2] with randomized time of $O(log\, n)$ to compute a maximal independent set, we can obtain an $O(\frac{1}{\varepsilon^3} log^2 n)$ randomized time algorithm.

References

[1] S. Hougardy, D.E. Vinkemeier, *Approximating weighted matchings in parallel*, Information Processing Letters 99 (2006), p. 119–123.

[2] M. Luby, *A simple parallel algorithm for the maximal independent set problem*. SIAM Journal on Computing 15 (1986), p. 1036–1053.

[3] D. Peleg, *Distributed Computing. A Locality-Sensitive Approach*. SIAM, (2000).

[4] T. Nieberg, *A Local Approximation Algorithm for Maximum Weight Matching*. Technical Report, to appear.

An Exact Approach for solving the Balanced Minimum Evolution Problem

Roberto Aringhieri [a] Chiara Braghin [a] Daniele Catanzaro [b,*]

[a] *DTI, Università degli Studi di Milano, via Bramante 65, 26013 Crema, Italy*

[b] *GOM, Université Libre de Bruxelles, Boulevard du Triomphe CP 210/01, 1050, Brussels, Belgium*

Key words: network design, computational biology, phylogenetics, minimum evolution

---

Molecular phylogenetics studies the hierarchical evolutionary relationships among organisms (also called *taxa*) by means of molecular data (e.g., DNA or protein sequences). These relationships are typically described by means of weighted trees, or phylogenies, whose leaves represent taxa, internal vertices the intermediate ancestors, edges the evolutionary relationships between pairs of taxa, and edge weights the *evolutionary distances* (i.e., measures of the dissimilarity) between pairs of taxa [6]. Molecular phylogenetics provides several criteria for selecting one phylogeny from among plausible alternatives [3]. One of the most important criteria is the Minimum Evolution (ME) criterion [8, 11, 12]: it states that, given a set Γ of n taxa and the corresponding $n \times n$ symmetric matrix $\mathbf{D} = \{d_{ij}\}$ of evolutionary distances, the optimal phylogeny for Γ is the one whose sum of edge weights, estimated from $\mathbf{D}$, is minimal. The biological justification at the core of the ME criterion is based on the fact that, in absence of convergent or reverse evolution [2], the true phylogeny of Γ has an expected sum of edge weights smaller than any other possible phylogeny compatible with $\mathbf{D}$.

Phylogenies satisfying the ME criterion are determined by solving a *Minimum Evolution Problem* (MEP), generally $\mathcal{NP}$-Hard [3], which can be stated as follows. Consider an connected, unweighted, undirected *phylogenetic graph* $G = (V, \mathcal{E})$, where $V = V_e \cup V_i$ is the set of vertices. V_e is the set of n *leaves* representing the n taxa in Γ, and V_i the set of $(n-2)$ *internal vertices* representing the common ancestors. By analogy, $\mathcal{E} = \mathcal{E}_e \cup \mathcal{E}_i$ is the set of $\frac{3}{2}(n-1)(n-2)$ edges, $\mathcal{E}_e$ is the set of *external edges*, i.e., the set of edges with one extreme being a leaf, and $\mathcal{E}_i$ is the set of *internal edges*, i.e., the set of edges with both extremes being internal vertices. A *phylogeny* T of the set Γ is any spanning tree T of G such that each internal vertex has degree three, and each leaf has degree one. Denote $\mathcal{E}(T)$ as the set of edges of a phylogeny T, $\mathcal{T}$ as the set of all the possible $(2n-5)!!$ phylogenies of Γ (where $n!!$ is the double factorial of n) [6], and assume that a weight function $f : \mathcal{E}(T) \rightarrow \Re$

* Corresponding author.

is given. Denote $\mathbf{w}$ as the $(2n-3)$-vector of edge weights associated to T, and let $L(T,\mathbf{w})$ be the *length* of T, i.e., the sum of the associated edge weights. Then, the minimum evolution problem consists in:

$$\min_{(T,\mathbf{w})} L(T,\mathbf{w}) \text{ s.t. } f(\mathbf{D},T,\mathbf{w}) = 0, \mathbf{T} \in \mathcal{T}, \mathbf{w} \in \Re_{0^+}^{(2n-3)} \tag{1}$$

where $f(\mathbf{D},T,\mathbf{w})$ is a function correlating the distance matrix $\mathbf{D}$ with the phylogeny T and edge weights $\mathbf{w}$. Defining the function $f(\mathbf{D},T,\mathbf{w})$ means specifying an edge weight estimation model, i.e., a model to compute edge weights starting from the knowledge of $\mathbf{D}$ and T [3]. Thus, a version of MEP is completely characterized by specifying the functions $L(T,\mathbf{w})$ and $f(\mathbf{D},T,\mathbf{w})$.

Several versions of MEP are known in the literature (see [3] for a recent survey), each one characterized by its own set of assumptions on the function $f(\mathbf{D},T,\mathbf{w})$. The most recent version is the Balanced Minimum Evolution (BME) problem [4, 5] which is based on Pauplin's edge weight estimation model [10]. Pauplin proved that under this edge weight estimation model the length of a phylogeny T can be computed as:

$$L(T) = \sum_{i=1}^{n}\sum_{j=1}^{n} 2^{-\tau_{ij}} d_{ij} \tag{2}$$

where τ_{ij} is the number of edges belonging to the path between taxa i and j in T. Hence, solving MEP under Pauplin's edge weight estimation model (i.e., solving BME) means to minimize the function (2) with respect to all the possible phylogenies T in G. To the best of our knowledge, to date the only attempt at solving BME is represented by heuristic approaches [5]. In the remaining of the paper, we show an exact approach to tackle instances of BME exploiting the tree isomorphism [1] and the combinatorial properties of the function (2) [7, 9, 13].

Several authours [7, 9, 13] studied the combinatorial properties of the function (2) and evidenced the relationship between BME and the Traveling Salesman Problem (TSP). Specifically, defined $G_\Gamma(\Gamma,E,w)$ as a complete, undirected, weighted graph whose vertices are taxa in Γ and whose weights $w_{ij} = d_{ij}$, for all $i,j \in \Gamma$, the authors proved that the length of the optimal solution of BME is equal to half-time the length of the shortest hamiltonian circuit in G_Γ. In other words, the shortest hamiltonian circuit in G_Γ identifies the way in which the taxa are ordered in the optimal phylogeny [7, 9]. Note that, for any fixed hamiltonian circuit H in G_Γ there exist 2^{n-2} possible phylogenies having the taxa order identified by H. Hence, a possible way to solve exactly an instance of BME would consist in enumerating all those 2^{n-2} possible phylogenies whose taxa order is identified by the shortest hamiltonian circuit in G_Γ. Note that, once the instance of TSP is solved, this approach has the benefit of reducing the space of solutions of BME from $(2n-5)!!$ to 2^{n-2}. However, since the length of isomorphic phylogenies are equivalent, the restriction of such enumeration only to non-isomorphic phylogenies would drastically decrease (see [6]) the dimension of the solution space of BME and improve the overall running time of the algorithm. This is the fundamental idea of our enumerative algorithm whose pseudocode is reported in Figure 1.

procedure `BMESolver`(Γ: set of taxa, $\mathbf{D}$: distance matrix)
$C \leftarrow$SolveTSP(Γ,$\mathbf{D}$);
T^* =NULL; $min_{val} = \infty$;
for Any possible non-isomorphic phylogeny T **do**
  for any pair of unassigned leaves i and j in T **do** Compute τ_{ij};
  $T_C \leftarrow$AssignLeaves(C,T)
  if $min_{val} \geq L(T_C)$ **then** $min_{val} = L(T_C)$; $T^* = T_C$
  for Any clockwise shift R of C on T **do**
    $T_R \leftarrow$AssignLeaves(R,T)
    if $min_{val} \geq L(T_R)$ **then** $min_{val} = L(T_R)$; $T^* = T_R$
  end for
end for
return T^* and min_{val}
end-procedure

Fig. 1. BMESolver pseudo-code.

Given an instance I of BME, denote T^*, respectively min_{val}, as the optimal phylogeny of I, respectively the optimal length of T^*. Our algorithm starts solving TSP on G_Γ; the solution C so obtained identifies the taxa order in T^*. The algorithm proceeds by enumerating all the possible non-isomorphic phylogenies following the algorithm described in [1]: for each non-isomorphic phylogeny T, the algorithm assigns to each leaf a taxa as shown in Figure 2a; then it computes the value (2) and the eventual minimum is stored. Since the TSP solution only identifies the taxa order in T^*, the next $(n-1)$ algorithm steps consist in a clockwise rotation of taxa assignment to leaves (see Figure 2b) and the computation of the corresponding new value (2). When all the possible non-isomorphic phylogenies have been enumerated, the algorithm ends returning the optimal phylogeny found.

Preliminary computational results show the effectiveness of our algorithm for solving small and medium size instances.

References

[1] R. Aringhieri, P. Hansen, and F. Malucelli. Chemical trees enumeration algorithms. *4OR*, 1:67–83, 2003.

[2] W. A. Beyer, M. Stein, T. Smith, and S. Ulam. A molecular sequence metric and evolutionary trees. *Mathematical Biosciences*, 19:9–25, 1974.

[3] D. Catanzaro. The minimum evolution problem: Overview and classification. *Networks*, In print, 2008.

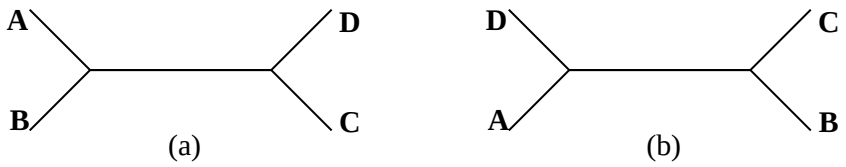


Fig. 2. (a) assignment of taxa to leaves, (b) a clockwise rotation

[4] R. Desper and O. Gascuel. Fast and accurate phylogeny reconstruction algorithms based on the minimum evolution principle. *Journal of Computational Biology*, 9(5):687–705, 2002.

[5] R. Desper and O. Gascuel. Theoretical foundations of the balanced minimum evolution method of phylogenetic inference and its relationship to the weighted least-squares tree fitting. *Molecular Biology and Evolution*, 21(3):587–598, 2004.

[6] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, Sunderland, MA, 2004.

[7] G. H. Gonnet, C. Korostensky, and S. Benner. Evaluation measures of multiple sequence alignments. *Journal of Computational Biology*, 7(1-2):261–276, 2000.

[8] K. K. Kidd and L. A. Sgaramella-Zonta. Phylogenetic analysis: Concepts and methods. *American Journal of Human Genetics*, 23:235–252, 1971.

[9] C. Korostensky and G. H. Gonnet. Using traveling salesman problem algorithms for evolutionary tree construction. *Bioinformatics*, 16(7):619–627, 2000.

[10] Y. Pauplin. Direct calculation of a tree length using a distance matrix. *Journal of Molecular Evolution*, 51:41–47, 2000.

[11] A. Rzhetsky and M. Nei. Statistical properties of the ordinary least-squares generalized least-squares and minimum evolution methods of phylogenetic inference. *Journal of Molecular Evolution*, 35:367–375, 1992.

[12] A. Rzhetsky and M. Nei. Theoretical foundations of the minimum evolution method of phylogenetic inference. *Molecular Biology and Evolution*, 10:1073–1095, 1993.

[13] C. Semple and M. Steel. Cyclic permutations and evolutionary trees. *Advances in Applied Mathematics*, 32(4):669–680, 2004.

Heuristic and exact approaches to the Quadratic Minimum Spanning Tree Problem

Roberto Cordone [a,*] Gianluca Passeri [a]

[a] *DTI, Università degli Studi di Milano, via Bramante 65, 26013 Crema, Italy*

Key words: Minimum spanning tree problem, Quadratic programming, Tabu Search, Branch-and-bound

---

The problem Consider a connected undirected graph $G = (V, E)$ with $|N| = n$ vertices and $|E| = m$ edges, a linear cost function $c : E \to \mathbb{N}$ defined on the edges and a quadratic cost function $q : E \times E \to \mathbb{N}$ defined on the pairs of edges. The *Quadratic Minimum Spanning Tree Problem* (*QMSTP*) requires to determine a spanning tree $T = (V, X)$ minimizing the sum of all linear costs for the edges in X plus all quadratic costs for the pairs of edges in X. Without loss of generality, we assume $q_{ij} = q_{ji}$ and $q_{ii} = 0$ for all $i, j \in N$.

Let $x_i = 1$ if edge i belongs to the solution, $x_i = 0$ otherwise, and $E(S)$ denote the set of edges with both ends in $S \subseteq V$. A formulation for the *QMSTP* is:

$$\min z = \sum_{i \in E} c_i x_i + \sum_{i \in E} \sum_{j \in E} q_{ij} x_i x_j \tag{1}$$

$$\sum_{i \in E} x_i = n - 1 \tag{2}$$

$$\sum_{i \in E(S)} x_i \leq |S| - 1 \qquad S \subseteq V : |S| \geq 2 \tag{3}$$

$$x_i \in \{0, 1\} \qquad i \in E \tag{4}$$

The *QMSTP* has applications in network design, when interference costs between links have to be considered. Despite its neat structure, it has been seldom considered in the literature: two greedy algorithms have been proposed in [4] and compared to a genetic algorithm in [5]. Two other genetic algorithms have been proposed to solve a fuzzy variation of the problem [1, 2].

The *QMSTP* is $\mathcal{NP}$-hard in the strong sense and not approximable (unless $\mathcal{P} = \mathcal{NP}$), even if $c_i = 0$ for all $i \in E$ and $q_{ij} \in \{0, 1\}$ for all $(i, j) \in E \times E$. The number of solutions, given by Kirchoff's theorem [3], sharply increases with n, unless the graph is very sparse. It is n^{n-2} for complete graphs.

* Corresponding author.

The heuristic algorithms We have implemented three constructive greedy algorithms and a Tabu Search. The constructive heuristics approximate the quadratic objective function with a linear one, thus reducing the *QMSTP* to an auxiliary *Minimum Spanning Tree Problem* (*MSTP*):

$$\min \tilde{z} = \sum_{i \in E} \tilde{c}_i x_i \;\text{ subject to } (2-4), \text{ with } \quad \tilde{c}_i \approx c_i + \sum_{j \in E} q_{ij} x_j, \forall i \in E$$

The *Minimum Contribution Method* (*MCM*) estimates $\tilde{c}_i$ as the linear cost c_i plus the $(n-2)$ minimum quadratic costs between i and the other edges (notice that $\sum_{j \in E} q_{ij} x_j$ includes exactly $(n-2)$ non-zero terms in all feasible solutions). The rationale is that, if i is selected, these edges are more likely to enter the solution.

The *Average Contribution Method* (*ACM*) estimates $\tilde{c}_i$ as the linear cost c_i plus $(n-2)$ times the average quadratic cost between i and the other edges:

$$\tilde{c}_i = c_i + (n-2)\,\frac{\sum_{j \in E} q_{ij}}{m-1} \tag{5}$$

The *Sequential Fixing Method* (*SFM*) updates step by step estimate (5). Let X' be the current subset of chosen edges and $F \subseteq E \setminus X'$ the current subset of unchosen edges e which can be feasibly added to X', i.e. such that $\{e\} \cup X'$ is acyclic. Then, $\tilde{c}_i$ is the linear cost c_i plus the quadratic costs between i and the edges in X', plus the number of edges to complete a spanning tree times the average quadratic cost between i and the edges in F.

$$\tilde{c}_i = c_i + \sum_{j \in X'} q_{ij} + (n-2-|X'|)\,\frac{\sum_{j \in F} q_{jk}}{|F|} \tag{6}$$

At the first step $X' = \emptyset$, $F = E \setminus \{i\}$ and (6) reduces to (5). The *ACM* and the *SFM* have been proposed in [4], with slightly different (and less consistent with the underlying rationale) expressions for $\tilde{c}_i$.

The Tabu Search algorithm is based on a natural neighbourhood: each move adds a new edge and removes one of the $n' \leq n-1$ edges from the resulting loop. Suitable data structures allow to scan, for each of the $(m-n+1)$ edges to be added, only the n' edges which can be feasibly removed. Each move is evaluated in $O(1)$ by maintaining in memory coefficient $D_i = c_i + \sum_{j \in X} q_{ij}$ for all $i \in E$. In fact, when exchanging edges $i \in X$ and $i' \in E \setminus X$, the objective function varies by $\delta_{ii'} = D_{i'} - D_i - 2q_{ii'}$. Since the update of coefficients D_i after performing the chosen move takes $O(m)$ time, the overall complexity of a neighbourhood exploration is $O(mn)$.

The tabu mechanism forbids recently removed edges to get into the solution and recently added edges to get out of it. Of course, a tabu move improving the best known result is performed anyway (*aspiration criterion*). Notice that (apart from very sparse graphs) the number of edges out of the solution exceeds the number

of edges inside it. To guarantee a comparable strength to the two tabus, the length of the prohibition (*tabu tenure*) for the insertion is larger than that for the removal. Moreover, the tabu tenures vary (inside suitable ranges) according to the results of the search: they increase for each improvement in the objective function and decrease for each worsening.

The exact algorithm We have also implemented a branch-and-bound algorithm based on the relaxation of the quadratic objective function to a linear approximated one. The relaxed subproblem is, thus, a *MSTP*. For each edge i, the linear estimate $\hat{c}_i$ includes the linear cost c_i, the quadratic costs with the already fixed edges and a suitable number of cheapest quadratic costs with the unfixed edges. This approximation differs from equation (6) for two features: 1) in the second term, the subset of already fixed edges replaces the subset of already chosen ones; 2) in the third term, the cheapest quadratic costs replace the average cost (as in the *MCM*).

The solution of the relaxed subproblem is a spanning tree, which, evaluated by the original objective function, provides an upper bound at each node. An initial upper bound is also given by the *SFM* followed by Tabu Search.

Branching is performed by fixing an edge in or out of the solution. The branching edge is the cheapest unfixed one which belongs to the solution of the relaxed *MSTP*. The visit strategy combines a *best-upper-bound-first* strategy (visit the open branching node with the best upper bound) to the more usual *best-lower-bound-first* strategy (visit the open branching node with the best lower bound): the algorithm starts with the former, switching to the latter when the upper bound does not improve for a predefined number of branching nodes. The reason is that in the upper levels of the branching tree the lower bound is often bad, whereas the heuristic solution gives stronger information on the problem.

The computational results We have generated a benchmark of 60 random instances with different features, one instance for each combination of the following features: size (from 10 to 30 vertices by steps of 5), density (33%, 67% or 100%), linear costs (uniformly generated random integers in $[1; 10]$ or $[1; 100]$), quadratic costs (uniformly generated random integers in $[1; 10]$ or $[1; 100]$). The experimental campaign has been performed on a 2.2 GHz PC with 3 GB of RAM and all algorithms have been implemented in C language.

All greedy constructive heuristics run in less than one second. The *SFM* obviously outperforms the other ones, providing better or equal results in 57 cases out of 60, but only in 4 cases this is also the best known result overall.

The Tabu Search runs for $100\,000$ iterations with a tabu tenure varying in $[3; 8]$ for the edge removal and $[5; 12]$ for the edge insertion. It is robust enough to make the initialization heuristic nearly always irrelevant (of course, the number of iterations required to find the best known result strongly depends on it). The computational time ranges from less than one second ($n = 10$) to less than one minute ($n = 30$).

The branch-and-bound solves in less than one minute all instances with $n = 10$ and all sparse instances (33% density) with $n = 15$, plus a single instance with $n = 15$

and 67% density. The other instances could not be solved in one hour, exhibiting remarkable final gaps. In fact, the hardness of the problem sharply increases with the number of edges. The same few optimal results and similar conclusions have been obtained by applying CPLEX to the standard linearization of formulation (1-4).

References

[1] J. Gao and M. Lu. Fuzzy quadratic minimum spanning tree problem. *Applied Mathematics and Computation*, 164:773–788, 2005.

[2] J. Gao, M. Lu, and L. Liu. Chance-constrained programming for fuzzy quadratic minimum spanning tree problem. In *Proceedings of the 2004 IEEE International Conference on Fuzzy Systems*, pages 983–987, Piscataway, NJ, USA, July 2004. IEEE Press.

[3] G. Kirchhoff. Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme geführt wird. *Annelen der Physik und Chemie*, 72: 497–508, 1847.

[4] W. Xu. On the quadratic minimum spanning tree problem. In M. Gen and W. Xu, editors, *Japan-China International Workshop on Information Systems*, pages 141–148, 1995.

[5] G. Zhou and M. Gen. An effective genetic algorithm approach to the quadratic minimum spanning tree problem. *Computers & Operations Research*, 25(3):229–237, July 1998.

Appendix

Theorem 1 *The* QMSTP *is $\mathcal{NP}$-hard in the strong sense and not approximable (unless $\mathcal{P} = \mathcal{NP}$), even if $c_i = 0$ for all $i \in E$ and $q_{ij} \in \{0, 1\}$ for all $(i, j) \in E \times E$.*

Proof Given an instance of *SAT*, build the following instance of *QMSTP*: graph $G = (V, E)$ has a vertex y_i for each clause and a vertex x_j for each Boolean variable; an edge is given for each pair of vertices (y_i, x_j) such that literal x_j or $\bar{x}_j$ appears in clause i and for each pair of vertices (x_j, x_{j+1}). The linear cost function is identically zero, while $q_{ee'} = 1$ if $e = (y_i, x_j)$, $e' = (y_l, x_j)$ and variable x_j appears affirmed in clause i and negated in clause l or viceversa; $q_{ee'} = 0$ for all other edges.

This instance of *QMSTP* has a solution of zero cost if and only if there is a spanning tree such that all pairs of its edges have zero quadratic cost. The edges (y_i, x_j) can be interpreted as the use of variable x_j to satisfy clause i. Therefore, the zero cost trees exactly correspond to the consistent assignments satisfying all clauses.

As the cost of any other feasible solution is ≥ 1, any polynomial algorithm with a constant approximation guarantee would necessarily find the optimal solution. Hence, *QMSTP* does not belong to $\mathcal{APX}$.

Polyhedra and formulations

Room A, Session 3

Tuesday 13, 15:30-17:00

On the cardinality constrained matroid polytope

Rüdiger Stephan *

Institute for Mathematics, Strasse des 17. Juni 136, 10623 Berlin, Germany

Key words: Matroid polytope, Cardinality constraints, Facets, Separation problem

1. Introduction and main results

In [4] it was shown how one can combine integer characterizations for cycle and path polytopes and Grötschel's cardinality forcing inequalities [3] to give facet defining integer representations for the cardinality restricted versions of these polytopes. Motivated by this work, we apply the same approach on the matroid polytope. It is well known that the so-called rank inequalities together with the nonnegativity constraints provide a complete linear description of the matroid polytope (see Edmonds [2]). By essentially adding the cardinality forcing inequalities, we obtain a complete linear description of the cardinality constrained matroid polytope which is the convex hull of the incidence vectors of those independent sets that have a feasible cardinality. Moreover, we show how the separation problem for the cardinality forcing inequalities can be reduced to that for the rank inequalities. We give also necessary and sufficient conditions for a cardinality forcing inequality to be facet defining.

Given a matroid $M = (E, \mathcal{I})$ with rank function r and a weighting $w_e \in \mathbb{R}$ on the elements, the maximum weight independent set problem $\max w(I) := \sum_{e \in I} w_e,\ I \in \mathcal{I}$ can be solved to optimality with the greedy algorithm. Moreover, the *matroid polytope* $P_{\mathcal{I}}(E)$, that is, the convex hull of the incidence vectors of independent sets $I \in \mathcal{I}$, is determined by the so-called rank inequalities and the nonnegativity constraints (see Edmonds [2]), i.e., $P_{\mathcal{I}}(E)$ is the set of all points $x \in \mathbb{R}^E$ satisfying

$$\begin{aligned} \sum_{e \in F} x_e &\leq r(F) \text{ for all } \emptyset \neq F \subseteq E, \\ x_e &\geq 0 \qquad \text{for all } e \in E. \end{aligned}$$

Here, for any $I \subseteq E$ we set $x(I) := \sum_{e \in I} x_e$. The rank inequality associated with F is facet defining for $P_{\mathcal{I}}(E)$ if and only if F is closed and inseparable (see Edmonds [2]).

* Corresponding author.

Let $c = (c_1, \ldots, c_m)$ be a finite sequence of integers with $0 \leq c_1 < c_2 < \cdots < c_m$. Then, the *cardinality constrained independent set polytope* $P^c_{\mathcal{I}}(E)$ is defined to be the convex hull of the incidence vectors of the independent sets $I \in \mathcal{I}$ with $|I| = c_p$ for some $p \in \{1, \ldots, m\}$. The associated optimization problem $\max w(I), I \in \mathcal{I}$, $|I| = c_p$ for some $p \in \{1, \ldots, m\}$ can be solved in polynomial time (for instance with Lawler's weighted matroid intersection algorithm [5] applied to $M_1 := M_2 := M$, see the item "Open Questions" on page 3).

Grötschel [3] gave a polyhedral analysis of the underlying basic problem of cardinality restrictions that enables us to provide a complete linear description of $P^c_{\mathcal{I}}(E)$. Given a finite set B and a cardinality sequence $c = (c_1, \ldots, c_m)$, the set $\text{CHS}^c(B) := \{F \subseteq B : |F| = c_p \text{ for some } p\}$ is called a *cardinality homogenous set system*. Consequently, when $M = (E, \mathcal{I})$ is the trivial matroid, i.e., all $F \subseteq E$ are independent sets, then $\mathcal{I} \cap \text{CHS}^c(E) = \text{CHS}^c(E)$. Thus, cardinality constrained matroids are a generalization of cardinality homogenous set systems.

The polytope associated with $\text{CHS}^c(B)$, namely the convex hull of the incidence vectors of elements of $\text{CHS}^c(B)$, is completely described by the *trivial inequalities* $0 \leq z_e \leq 1$, $e \in B$, the *cardinality bounds* $c_1 \leq \sum_{e \in B} z_e \leq c_m$, and the *cardinality forcing inequalities*

$$\begin{aligned} &(c_{p+1} - |F|) \sum_{e \in F} z_e \;-\; (|F| - c_p) \sum_{e \in B \setminus F} z_e \leq c_p(c_{p+1} - |F|) \\ &\quad \text{for all } F \subseteq B \text{ with } c_p < |F| < c_{p+1} \text{ for some } p \in \{1, \ldots, m-1\}. \end{aligned} \tag{1}$$

Result 1. In the full paper we will show by case-by-case enumeration that the system

$$\begin{aligned} &(c_{p+1} - r(F))x(F) - (r(F) - c_p)x(E \setminus F) \;\leq\; c_p(c_{p+1} - r(F)) \\ &\quad \text{for all } F \subseteq E \text{ with } c_p < r(F) < c_{p+1} \text{ for some } p \in \{0, \ldots, m-1\}, \end{aligned} \tag{2}$$

$$x(E) \;\geq\; c_1, \tag{3}$$

$$x(E) \;\leq\; c_m, \tag{4}$$

$$x(F) \leq\; r(F) \qquad \text{for all } \emptyset \neq F \subseteq E, \tag{5}$$

$$x_e \;\geq\; 0 \qquad \text{for all } e \in E \tag{6}$$

completely describes $P^c_{\mathcal{I}}(E)$.

Of course, each $x \in P^c_{\mathcal{I}}(E)$ satisfies $c_1 \leq x(E) \leq c_m$. The cardinality forcing inequality $\text{CF}_F(x) := (c_{p+1} - r(F))x(F) - (r(F) - c_p)x(E \setminus F) \leq c_p(c_{p+1} - r(F))$ associated with F, where $c_p < r(F) < c_{p+1}$, is valid as can be seen as follows. The incidence vector of any $I \in \mathcal{I}$ of cardinality at most c_p satisfies the inequality, since $r(I \cap F) \leq c_p$:

$$\begin{aligned}(c_{p+1} - r(F))\chi^I(F) - (r(F) - c_p)\chi^I(E \setminus F) &\leq (c_{p+1} - r(F))\chi^I(F) \\ &\leq (c_{p+1} - r(F))c_p.\end{aligned}$$

The incidence vector of any $I \in \mathcal{I}$ of cardinality at least c_{p+1} satisfies also the inequality, since $r(I \cap F) \leq r(F)$ and thus $r(I \cap (E \setminus F)) \geq c_{p+1} - r(F)$:

$$\begin{aligned}&(c_{p+1} - r(F))\chi^I(F) - (r(F) - c_p)\chi^I(E \setminus F) \\ \leq\ &(c_{p+1} - r(F))r(F) - (r(F) - c_p)\chi^I(E \setminus F) \\ \leq\ &(c_{p+1} - r(F))r(F) - (r(F) - c_p)(c_{p+1} - r(F)) \\ =\ &c_p(c_{p+1} - r(F)).\end{aligned}$$

However, it is not hard to see that some incidence vectors of independent sets I with $c_p < |I| < c_{p+1}$ violate the inequality.

Result 2. In the most cases, namely if $c_p > 0$ and $c_{p+1} < r(E)$, a cardinality forcing $\mathrm{CF}_F(x) \leq c_p(c_{p+1} - r(F))$ is facet defining if and only if F is closed. Here we only show necessity. Assume that F is not closed. Then, there is some $e \in E \setminus F$ such that $r(F \cup \{e\}) = r(F)$. Consequently, $c_p < r(F \cup \{e\}) < c_{p+1}$, and $\mathrm{CF}_F(x) \leq c_p(c_{p+1} - r(F))$ is the sum of the valid inequalities $\mathrm{CF}_{F\cup\{e\}}(x) \leq c_p(c_{p+1} - r(F \cup \{e\}))$ and $-x_e \leq 0$. In the full paper we also give necessary and sufficient conditions for the remaining cases ($c_p = 0$ or $c_{p+1} = r(E)$).

Result 3. The separation problem for the cardinality forcing inequalities (2) can be solved in polynomial time by tracing back to the separation problem of the rank inequalities. To get an idea of the transformation, let $x^* \in \mathbb{R}^E$ be any nonnegative vector. The separation problem for the class of cardinality forcing inequalities consists of checking whether or not

$$\begin{aligned}&(c_{p+1} - r(F))x^*(F) - (r(F) - c_p)x^*(E \setminus F) \ \leq\ c_p(c_{p+1} - r(F)) \\ &\text{for all } F \subseteq E \text{ with } c_p < r(F) < c_{p+1} \text{ for some } p \in \{0, \ldots, m-1\}.\end{aligned}$$

When one assumes that x^* satisfies all rank inequalities (5), then one can show that x^* violates the cardinality forcing inequality associated with F' if and only if $\frac{1}{\delta}x^*(F') - r(F') > \epsilon$ for appropriate $\delta, \epsilon > 0$. The latter problem can be approached with Cunningham's separation routine for the rank inequalities [1].

Open questions. It stands to reason to investigate the intersection of two matroids with regard to cardinality restrictions. It is well-known, if an independence system $\mathcal{I}$ defined on some ground set E can be described as the intersection of two matroids $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$, then the optimization problem $\max w(I),\ I \in \mathcal{I}$ can be solved in polynomial time, for instance with Lawler's weighted matroid intersection algorithm [5]. This algorithm solves also the cardinality constrained version $\max w(I),\ I \in \mathcal{I} \cap \mathrm{CHS}^c(E)$, since for each cardinality $p \leq r(E)$ it generates an independent set I of cardinality p which is optimal among all independent sets J of cardinality p. Thus, from an algorithmic point of view the problem is well studied. However, there is an open question regarding the associated polytope. It is well

known, $P_{\mathcal{I}}(E) = P_{\mathcal{I}_1}(E) \cap P_{\mathcal{I}_2}(E)$, that is, the non-cardinality constrained independent set polytope $P_{\mathcal{I}}(E)$ is determined by the nonnegativity constraints $x_e \geq 0$, $e \in E$, and the rank inequalities $x(F) \leq r_j(F)$, $\emptyset \neq F \subseteq E$, $j = 1, 2$, where r_j is the rank function with respect to $\mathcal{I}_j$. We do not know, however, whether or not $P^c_{\mathcal{I}}(E) = P^c_{\mathcal{I}_1}(E) \cap P^c_{\mathcal{I}_2}(E)$ holds. So far, we have not found any counterexample contradicting the hypothesis that equality holds.

References

[1] W. H. Cunningham, *Testing membership in matroid polyhedra.* J. Comb. Theory, Ser. B 36 (1984), 161-188.

[2] J. Edmonds, *Matroids and the greedy algorithm.* Math. Program. 1 (1971), 127-136.

[3] M. Grötschel. *Cardinality homogeneous set systems, cycles in matroids, and associated polytopes.* In: Grötschel, Martin (ed.), *The sharpest cut. The impact of Manfred Padberg and his work.* MPS/ SIAM Series on Optimization, 2004, pp. 99-120.

[4] V. Kaibel and R. Stephan. *On cardinality constrained cycle and path polytopes*. Technical report, Zuse Institute Berlin, 2007. Available at http://opus.kobv.de/zib/volltexte/2007/1024/.

[5] E. L. Lawler, *Matroid intersection algorithms.* Math. Program. 9 (1975), 31-56.

On the Facial Structure of the Common Edge Subgraph polytope [1]

Gordana Manić [a,*] Laura Bahiense [b] Cid de Souza [c]

[a] *Universidade Federal do ABC, SP, Brazil*

[b] *Universidade Federal do Rio de Janeiro, RJ, Brazil*

[c] *Universidade Estadual de Campinas, SP, Brazil*

Key words: Maximum common subgraph problem, mapping problem, graph isomorphism, polyhedral combinatorics, branch&cut algorithm

---

Introduction. It is necessary in many applications to compare objects represented as graphs and to determine the degree of the similarity between them. This is often accomplished by formulating the problem as the one involving the maximum common subgraph between the graphs being considered. We consider here the **Maximum Common Edge Subgraph Problem (MCES)** defined as follows. Given two graphs G and H with $|V_G| = |V_H|$, find a common subgraph of G and H (not necessary induced) with the maximum number of *edges*. Graphs are assumed to be simple, finite and undirected. As usual, we denote by V_G (resp. E_G) the set of vertices (resp. edges) of a given graph G.

The $\mathrm{M}CES$ problem was introduced by Bokhari in [1]. Since the $\mathrm{M}CES$ problem comes from parallel programming environments, G is usually referred to as the *task interaction graph*, and H as the *processors graph*. Vertices in G represent tasks (its edges join pairs of tasks with communication demands) and vertices in H are processors, a pair of processors being joined by an edge when they are directly connected. The problem consists in assigning, i.e. mapping, each task to one processor in such a way that the number of neighboring tasks assigned to connected processors is maximized.

The $\mathrm{M}CES$ problem is also of particular interest since it generalizes the graph isomorphism problem. Furthermore, the maximum common subgraph problems has become increasingly important in matching $2D$ and $3D$ chemical structures [5].

---

* Corresponding author.

[1] Work supported by grants 301732/2007-8, 478470/2006-1, 472504/2007-0 and 473726/2007-6 from *Conselho Nacional de Desenvolvimento Científico e Tecnológico*, grant 2007/170.479 from *Fundação Carlos Chagas Filho de Amparo àă Pesquisa do Estado do Rio de Janeiro* and grants 03/09925-5 and 06/01817-7 from *Fundação de Amparo à Pesquisa do Estado de São Paulo*. This research was developed while the first author was a Postdoctoral fellow at the *Universidade Estadual de Campinas, SP, Brazil*.

Note that if $|V_G| \neq |V_H|$, a suitable number of dummy vertices have to be inserted into the smaller graph in order to obtain an instance of $\mathrm{M}CES$. If $|V_G|$ and $|V_H|$ are not required to be equal, we obtain a problem which is $\mathrm{A}PX$-hard, while the $\mathrm{M}CES$ problem is only known to be $\mathrm{N}P$-hard [3]. There have been many attempts to devise useful algorithms for $\mathrm{M}CES$. Some of them approximate the solution of the $\mathrm{M}CES$ problem, while others give the exact solution for a specialized set of graphs or graphs of moderate size. But most of the approaches to $\mathrm{M}CES$ propose heuristic procedures intended for particular architectures [1, 2].

Our contribution. We present a new integer programming formulation for the $\mathrm{M}CES$ problem and carry out a polyhedral investigation of this model. A number of valid inequalities are identified, most of which are facet-defining. Those inequalities were incorporated to a branch&cut algorithm for the $\mathrm{M}CES$ problem. We report on our computational experiments, which show the contribution of the inequalities we found here.

A new integer programming formulation. The only polyhedral study of the $\mathrm{M}CES$ problem so far was done by Marenco in [3, 4]. The integer programming formulation presented by this author has variables y_{ik}, for $i \in V_G$, $k \in V_H$, which are 1 in a feasible solution if i is mapped to k, and 0 otherwise. Furthermore, his model also has variables x_{ij}, for $ij \in E_G$, which are 1 if exists $kl \in E_H$ such that i is mapped to k and j to l, and 0 otherwise. The main idea of our new model is to create variables that represent the assignment of edges of G to the edges of H. More formally, apart from variables y_{ik}, we also include variables c_{ijkl}, for all $ij \in E_G$ and $kl \in E_H$ which are 1 if ij is mapped to kl, and 0 otherwise. We present now our monotonous integer programming model for the $\mathrm{M}CES$ problem.

$$\max \sum_{ij \in E_G} \sum_{kl \in E_H} c_{ijkl} \tag{1}$$

$$\sum_{k \in V_H} y_{ik} \leq 1, \forall i \in V_G \qquad \sum_{i \in V_G} y_{ik} \leq 1, \forall k \in V_H \tag{2}$$

$$\sum_{kl \in E_H} c_{ijkl} \leq \sum_{k \in V_H} y_{ik}, \forall ij \in E_G \qquad \sum_{ij \in E_G} c_{ijkl} \leq \sum_{i \in V_G} y_{ik}, \forall kl \in E_H \tag{3}$$

$$\sum_{j \in N(i)} c_{ijkl} \leq y_{ik} + y_{il}, \forall i \in V_G, \forall kl \in E_H \tag{4}$$

$$\sum_{l \in N(k)} c_{ijkl} \leq y_{ik} + y_{jk}, \forall ij \in E_G, \forall k \in V_H$$

$$c_{ijkl} \in \{0,1\}, \forall ij \in E_G, \forall kl \in E_H \qquad y_{ik} \in \{0,1\}, \forall i \in V_G, \forall k \in V_H \tag{5}$$

Inequalities in (2) force that every vertex of G is mapped to at most one vertex of H; and that for every vertex of H, there is at most one vertex of G mapped to it. Similar inequalities for edges are in (3). First inequality in (4) forces that for a fixed vertex i from G and a fixed edge kl from H, if some edge incident to i is mapped to kl, then i is mapped either to k or to l. Second inequality in (4) is analogous. Note that we work with the monotonous model since the proofs of facet-defining inequalities are substantially easier than in the model given in [3]. This is because the monotone polytope associated to the above formulation can be easily shown to be full-dimensional.

We present now some valid inequalities and facets that we found for the polytope P given by the convex hull of the integer solutions of the integer programming model (1)–(5). Using standard techniques from Polyhedral Combinatorics, we were able to show that inequalities (3) and (4) from our model define facets. The next theorem shows that a class of valid inequalities for the model given in [3], which involves vertex degrees, is part of a broader family of facet-defining inequalities in our model. To state this result we denote by $N(i)$ the set of all neighbours of a given vertex i.

Theorem 1 *Let i be a fixed vertex from G, k a fixed vertex from H, $I \subseteq N(i)$ and $K \subseteq N(k)$. Then, following inequalities are valid and define facets.*

$$\sum_{j\in I}\sum_{l\in K} c_{ijkl} \leq \min\{|I|,|K|\}y_{ik} + \sum_{p\in K} y_{ip} = |I|y_{ik} + \sum_{p\in K} y_{ip},\ \textit{if } |I| < |K|. \quad (6)$$

$$\sum_{j\in I}\sum_{l\in K} c_{ijkl} \leq \min\{|I|,|K|\}y_{ik} + \sum_{p\in I} y_{pk} \leq |K|y_{ik} + \sum_{p\in I} y_{pk},\ \textit{if } |I| > |K|. \quad (7)$$

It is worth noting that we obtained a facet that generalizes the result of Theorem 1. For given edges ij in G, and kl in H, it bounds the number of edges in G incident to ij that can be mapped to edges incident to kl in H.

In the next theorem we introduce a facet-defining inequality where the benefit of having an extended formulation including the variables c_{ijkl} becomes apparent. More precisely, we are able to express the following very simple inequality which can not be written in the model given in [3].

Theorem 2 *Let G' be an induced subgraph of G and M a maximal matching in H. If $|V_{G'}| = 2p+1$ and G' has an hamiltonian cycle, then inequality $\sum_{ij\in E_{G'}}\sum_{kl\in M} c_{ijkl} \leq p$ is valid. If $|M| \geq p+1$, then it defines a facet.*

Using the following theorem that explores the structure of the graphs given as input instances, we obtained better upper bounds for some instances.

Theorem 3 *Let $\mathcal{S}$ be a fixed graph. Let furthermore k_G (resp. k_H) be the maximum number of edge disjoint subgraphs in G (resp. in H) such that each of those subgraphs is isomorphic to $\mathcal{S}$. If $k_G \geq k_H$, then inequality $\sum_{e\in E_G}\sum_{w\in E_H} c_{ew} \leq |E_G| - (k_G - k_H)$ if $|E_G| \leq |E_H|$, is valid.*

Finally, we note that by lifting technique, we obtained a few stronger valid inequalities than presented in [3]. We omit them here due to space limitation.

Computational results. The polyhedral investigation described earlier was the starting point of our branch and bound ($\mathbf{B}\&\mathrm{B}$), as well as branch and cut ($\mathbf{B}\&\mathrm{C}$) algorithms. We used the same 71 instances used by Marenco, 16 of which are very small, with less than 10 vertices each. Other 19 instances have 20 vertices each, 9 instances have at least 30 vertices. The largest instance has 36 vertices. All the graphs are quite sparse and present a high degree of symmetry, with most of them being regular graphs.

We used XPRESS as the Integer Programming solver and the MOSEL language to code our programs. A fast polynomial time algorithm was designed to sepa-

rate inequalities (6) and (7). Besides, a routine was implemented that separates the inequality in Theorem 2, but only for $p = 1$ and $p = 2$. We also added *a priori* in the model all the inequalities from Theorem 3 for when $\mathcal{S}$ is a k-cycle and $k \in \{3, 4, 5\}$. Finally, another feature of our algorihtm was the implementation of a simple, though efficient, heuristic based on the solutions of the linear relaxations computed during the the enumeration.

Our $\mathbf{B}\&\mathrm{C}$ algorithm outperformed the standard $\mathbf{B}\&\mathrm{B}$ algorithm. Using our $\mathbf{B}\&\mathrm{C}$ algorithm, we managed to solve 39 instances, compared to the 31 solved by Marenco. Among the unsolved instances, 19 have duality gap of at most 10%, 11 have gap between 10 and 20%, and 2 have gap greater than 20%. Our algorithm is quite fast. Only few instances required more than 10 minutes to be solved and the execution time never exceeded 14 minutes.

Conclusions. We showed that with our extended formulation which include variables that interlaces edges of G with edges of H, we gain on expressiveness with respect to the model given in [3]. We carried out a polyhedral investigation of this new model and presented some valid inequalities and facets. This study led to some advance in obtaining the exact solutions to the $\mathrm{M}CES$ problem using Integer Programming and $\mathbf{B}\&\mathrm{C}$ algorithm.

References

[1] S. Bokhari. On the mapping problem. *IEEE Trans. Comput.*, C-30(3), 1981.

[2] J. Ramanujam F. Ercal and P. Sadayappan. Task allocation onto a hypercube by recursive mincut bipartitioning. *J. of Parallel and Distributed Comp.*, 10, 1990.

[3] J. Marenco. Un algoritmo branch-and-cut para el problema de mapping. Master's thesis, Universidade de Buenos Aires, 1999. Supervisor: I. Loiseau.

[4] J. Marenco. New facets of the mapping polytope. In *CLAIO*, 2006.

[5] J. W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J. of Computer-Aided Molecular Design*, 16:521–533, 2002.

Reformulations in Mathematical Programming: Definitions

Leo Liberti *

LIX, École Polytechnique, F-91128 Palaiseau, France

Key words: Opt-reformulation, narrowing, relaxation, approximation.

1. Introduction

The mathematical programming formulation language is a very powerful tool used to formalize optimization problems by means of parameters, decision variables, objective functions and constraints. Such diverse settings as combinatorial, integer, continuous, linear and nonlinear optimization problems can be defined precisely by their corresponding mathematical programming formulations. Its power is not limited to its expressiveness, but usually allows hassle-free solution of the problem: most general-purpose solution algorithms solve optimization problems cast in their mathematical programming formulation, and the corresponding implementations can usually be hooked into language environments which allow the user to input and solve complex optimization problems easily. It is well known that several different formulations may share the same numerical properties (feasible region, optima) though some of them are easier to solve than others with respect to the most efficient available algorithms. Being able to cast the problem in the best possible formulation is therefore a crucial aspect of any solution process.

When a problem with a given formulation P is cast into a different formulation Q, we say that Q is a reformulation of P. Curiously, the term "reformulation" appears in conjunction with "mathematical programming" over 400,000 times on Google; yet there are surprisingly few attempts to formally define what a reformulation in mathematical programming actually is [1, 7]. Furthermore, there is a remarkable lack of literature reviews on the topic of reformulations in mathematical programming [3]; and even more importantly, very few solution methods consider reformulation-based algorithmic steps (usually, the reformulation is taken to be a pre-processing step) [6]. Although some automatic relaxation software exists [2], there is no equivalent for general reformulations.

In this paper we propose a data structure for storing and manipulation mathematical programming formulations, and several definitions of different types of reformulations, all based on transformations carried out on the proposed data structure. A

* Corresponding author.

(non-exhaustive) list of known reformulations based on these definitions can be found in [4].

2. A data structure for mathematical programs

We refer to a mathematical programming problem in the most general form:

$$\min\{f(x) \mid g(x) \lesseqgtr b \wedge x \in X\}, \tag{1}$$

where f, g are function sequences of various sizes, b is an appropriately-sized real vector, and X is a cartesian product of continuous and discrete intervals. We let $\mathbb{P}$ be the set of all mathematical programming formulations and $\mathbb{M}$ be the set of all matrices.

Definition 1 *Given an alphabet $\mathcal{L}$ consisting of countably many alphanumeric names $N_{\mathcal{L}}$ and operator symbols $O_{\mathcal{L}}$, a mathematical programming formulation P is a 7-tuple $(\mathcal{P}, \mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{C}, \mathcal{B}, \mathcal{T})$, where:*

- *$\mathcal{P} \subseteq N_{\mathcal{L}}$ is the sequence of parameter symbols: each element $p \in \mathcal{P}$ is a parameter name;*
- *$\mathcal{V} \subseteq N_{\mathcal{L}}$ is the sequence of variable symbols: each element $v \in \mathcal{V}$ is a variable name;*
- *$\mathcal{E}$ is the set of expressions: each element $e \in \mathcal{E}$ is a Directed Acyclic Graph (DAG) $e = (V_e, A_e)$ such that:*

(a) $V_e \subseteq \mathcal{L}$ is a finite set

(b) there is a unique vertex $r_e \in V_e$ such that $\delta^-(r_e) = \emptyset$ (such a vertex is called the root vertex)

(c) vertices $v \in V_e$ such that $\delta^+(v) = \emptyset$ are called leaf vertices and their set is denoted by $\lambda(e)$; all leaf vertices v are such that $v \in \mathcal{P} \cup \mathcal{V} \cup \mathbb{R} \cup \mathbb{P} \cup \mathbb{M}$

(d) for all $v \in V_e$ such that $\delta^+(v) \neq \emptyset$, $v \in O_{\mathcal{L}}$

(e) two weightings $\chi, \xi : V_e \to \mathbb{R}$ are defined on V_e: $\chi(v)$ is the node coefficient and $\xi(v)$ is the node exponent of the node v; for any vertex $v \in V_e$, we let $\tau(v)$ be the symbolic term of v: namely, $v = \chi(v)\tau(v)^{\xi(v)}$.

Elements of $\mathcal{E}$ are sometimes called expression trees; nodes $v \in O_{\mathcal{L}}$ represent an operation on the nodes in $\delta^+(v)$, denoted by $v(\delta^+(v))$, with output in $\mathbb{R}$;

- *$\mathcal{O} \subseteq \{-1, 1\} \times \mathcal{E}$ is the sequence of objective functions; each objective function $o \in \mathcal{O}$ has the form (d_o, f_o) where $d_o \in \{-1, 1\}$ is the optimization direction (-1 stands for minimization, $+1$ for maximization) and $f_o \in \mathcal{E}$;*
- *$\mathcal{C} \subseteq \mathcal{E} \times \mathcal{S} \times \mathbb{R}$ (where $\mathcal{S} = \{-1, 0, 1\}$) is the sequence of constraints c of the form (e_c, s_c, b_c) with $e_c \in \mathcal{E}, s_c \in \mathcal{S}, b_c \in \mathbb{R}$: $c \equiv \begin{cases} e_c \leq b_c \text{ if } s_c = -1 \\ e_c = b_c \text{ if } s_c = 0 \\ e_c \geq b_c \text{ if } s_c = 1; \end{cases}$*
- *$\mathcal{B} \subseteq \mathbb{R}^{|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|}$ is the sequence of variable bounds: for all $v \in \mathcal{V}$ let $\mathcal{B}(v) = [L_v, U_v]$ with $L_v, U_v \in \mathbb{R}$;*
- *$\mathcal{T} \subseteq \{0, 1, 2\}^{|\mathcal{V}|}$ is the sequence of variable types: for all $v \in \mathcal{V}$, v is called a continuous variable if $\mathcal{T}(v) = 0$, an integer variable if $\mathcal{T}(v) = 1$ and a binary*

variable if $\mathcal{T}(v) = 2$.

We write $\mathcal{T}(z)$ and respectively $\mathcal{B}(z)$ to mean the sequences of types and respectively bound intervals of the sequence of variables in $z \subseteq \mathcal{V}$. We sometimes refer to a formulation by calling it an *optimization problem* or simply a *problem*. Consider a function $x : \mathcal{V} \to \mathbb{R}^{|\mathcal{V}|}$ (called *point*) which assigns values to the variables. A point x is *type feasible* if: $x(v) \in \mathbb{R}$ when $\mathcal{T}(v) = 0$, $x(v) \in \mathbb{Z}$ when $\mathcal{T}(v) = 1$, $x(v) \in \{L_v, U_v\}$ when $\mathcal{T}(v) = 2$, for all $v \in \mathcal{V}$; x is *bound feasible* if $x(v) \in \mathcal{B}(v)$ for all $v \in \mathcal{V}$; x is *constraint feasible* if for all $c \in \mathcal{C}$ we have: $e_c(x) \leq b_c$ if $s_c = -1$, $e_c(x) = b_c$ if $s_c = 0$, and $e_c(x) \geq b_c$ if $s_c = 1$. A point x is *feasible in* P if it is type, bound and constraint feasible. Denote by $\mathcal{F}(P)$ the feasible points of P. A feasible point x is a *local optimum* of P with respect to the objective $o \in \mathcal{O}$ if there is a non-empty neighbourhood N of x such that for all feasible points $y \neq x$ in N we have $d_o f_o(x) \geq d_o f_o(y)$. A feasible point x is a *global optimum* of P with respect to the objective $o \in \mathcal{O}$ if $d_o f_o(x) \geq d_o f_o(y)$ for all feasible points $y \neq x$. Denote the set of local optima of P by $\mathcal{L}(P)$ and the set of global optima of P by $\mathcal{G}(P)$. If $\mathcal{O}(P) = \emptyset$, we define $\mathcal{L}(P) = \mathcal{G}(P) = \mathcal{F}(P)$.

3. Reformulations

The generic term we employ for a problem Q related to a given problem P by some form of transformation carried out on the formulation of P as defined in Defn. 1 is *auxiliary problem*. Among the several possible auxiliary problem types, four are specially interesting and used quite commonly: transformations preserving all optimality properties (opt-reformulations); transformations preserving at least one global optimum (narrowings); transformations based on dropping constraints, variable bounds or types (relaxations); transformations that are one of the above types "in the limit" (approximations).

Opt-reformulations are auxiliary problems that preserve all optimality information. We define them by considering local and global optima. A local reformulation transforms all optima of the original problem into optima of the reformulated problem, although more than one reformulated optimum may correspond to the same original optimum. A global reformulation transforms all global optima of the original problem into global optima of the reformulated problem, although more than one reformulated global optimum may correspond to the same original global optimum.

Definition 2 *Q is a local reformulation of P if there is a function $\varphi : \mathcal{F}(Q) \to \mathcal{F}(P)$ such that (a) $\varphi(y) \in \mathcal{L}(P)$ for all $y \in \mathcal{L}(Q)$, (b) φ restricted to $\mathcal{L}(Q)$ is surjective. This relation is denoted by $P \prec_\varphi Q$. Q is a global reformulation of P if there is a function $\varphi : \mathcal{F}(Q) \to \mathcal{F}(P)$ such that (a) $\varphi(y) \in \mathcal{G}(P)$ for all $y \in \mathcal{G}(Q)$, (b) φ restricted to $\mathcal{G}(Q)$ is surjective. This relation is denoted by $P \lhd_\varphi Q$. We write $P \prec Q$ (resp. $P \lhd Q$) if there is a φ such that $P \prec_\varphi Q$ (resp. $P \lhd_\varphi Q$). Q is an opt-reformulation of P (denoted by $P < Q$) if $P \prec Q$ and $P \lhd Q$.*

Opt-reformulations can be chained (i.e. applied in sequence) to obtain other opt-reformulations.

Lemma 1 *The relations $\prec, \lhd, <$ are reflexive and transitive.*

Narrowings are auxiliary problems that preserve at least one global optimum. They come in specially useful in presence of problems exhibiting many symmetries: it may then be the huge amount of global optima that is preventing a search from being successful. An example of narrowing is given by the local cuts obtained from the symmetry group of the problem [5]. All opt-reformulations are a special case of narrowings; narrowings can be chained to obtain more complex narrowings. Chaining an opt-reformulation and a narrowing results in a narrowing.

Definition 3 *Q is a narrowing of P if there is a function $\varphi : \mathcal{F}(Q) \to \mathcal{F}(P)$ such that (a) $\varphi(y) \in \mathcal{G}(P)$ for all $y \in \mathcal{G}(Q)$.*

Loosely speaking, a relaxation of a problem P is an auxiliary problem of P with fewer constraints. Relaxations are useful because they often yield problems which are simpler to solve yet they provide a bound on the objective function value at the optimum. The "fundamental theorem" of relaxations states that relaxations provide bounds to the objective function. Opt-reformulations and narrowings are special types of relaxations. Relaxations can be chained to obtain other relaxations; chaining of relaxations with opt-reformulations and narrowings results in other relaxations.

Definition 4 *Q is a relaxation of P if $\mathcal{F}(P) \subsetneq \mathcal{F}(Q)$.*

Approximations are auxiliary problems dependent on a numerical parameter, which approximate as closely as desired other auxiliary problems for some limiting value of the parameter. Since approximations can be defined for all types of auxiliary problems, we can have approximations to opt-reformulations, narrowings, relaxations and approximations themselves. In general, approximations have no guarantee of optimality, i.e. solving an approximation may give results that are arbitrarily far from the optimum. In practice, however, approximations manage to provide solutions of good quality. Opt-reformulations, narrowings and relaxations are special types of approximations. Chaining approximations and other auxiliary problems yields an approximation.

Definition 5 *Q is an approximation of P if there is a countable sequence of problems Q_k (for $k \in \mathbb{N}$), a positive integer k' and an auxiliary problem Q^* of P such that: (a) $Q = Q_{k'}$; (b) for all expression trees $f^* \in \mathcal{O}(Q^*)$ there is a sequence of expression trees $f_k \in \mathcal{O}(Q_k)$ that represent functions converging uniformly to the function represented by f^* (c) for all $c^* = (e^*, s^*, b^*) \in \mathcal{C}(Q^*)$ there is a sequence of constraints $c_k = (e_k, s_k, b_k) \in \mathcal{C}(Q_k)$ such that: (i) the functions represented by e_k converge uniformly to the function represented by e^*; (ii) $s_k = s^*$ for all k; (iii) b_k converges to b.*

References

[1] C. Audet, P. Hansen, B. Jaumard, and G. Savard. Links between linear bilevel and mixed 0-1 programming problems. *Journal of Optimization Theory and Applications*, 93(2):273–300, 1997.

[2] E.P. Gatzke, J.E. Tolsma, and P.I. Barton. Construction of convex relaxations using

automated code generation techniques. *Optimization and Engineering*, 3:305–326, 2002.

[3] L. Liberti. *Reformulation and Convex Relaxation Techniques for Global Optimization*. PhD thesis, Imperial College London, UK, March 2004.

[4] L. Liberti. Reformulation techniques in mathematical programming, November 2007. Thèse d'Habilitation à Diriger des Recherches.

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

[6] N. Mladenović, F. Plastria, and D. Urošević. Reformulation descent applied to circle packing problems. *Computers and Operations Research*, 32(9):2419–2434, 2005.

[7] H. Sherali. Personal communication. June 2007.

Online optimization

Room B, Session 3

Tuesday 13, 15:30-17:00

UFO: Uncertainty Feature Optimization, an Implicit Paradigm for Problems with Noisy Data

N. Eggenberg [a,*] M. Salani [a] M. Bierlaire [a]

[a] *Transp-OR, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland*

Key words: optimization, uncertainty, multi-objective, uncertainty features

1. Introduction

Nowadays, Operations Research tools are widely used to optimize real world problems. The underlying difficulty of real world applications is that most of them are due to uncertainty. As shown in [8] and [15] and references therein, the uncertainty should not be neglected when optimizing. There are three main approaches to cope with uncertainty in operations research.

The first approach is a reactive method, usually called *online* or *recovery* algorithm. In such methods, the uncertainty is neglected at optimization phase, but handled thanks to the re-optimization process. Although these methods are efficient in many applications, their main disadvantage is that no bound for the final solution is provided and that the re-optimization itself and the implementation of the new solution are time-consuming, which is contrasting with the online requirements of immediate reactivity. See [1] for a general survey on online algorithms.

The second general approach is stochastic optimization, where the solution with lowest *expected* cost is wanted. The advantage is that when the solution is implemented several times under identical conditions, we are provided with an approximation on the average cost. However, the method needs the characterization of an uncertainty set and a probabilistic distribution on it. See [8], [10] and [15] for more details on stochastic optimization.

Finally, the latest researches focus on *robust* optimization, [3] and [4]. The advantage of this approach is to find, if it exists, a solution that is feasible for any realization within an uncertainty set, and an upper cost bound is provided. However, this is a conservative worst case approach, which may produce solutions with high costs in average to ensure feasibility on the whole uncertainty set.

* Corresponding author.

The motivations to derive the concept of *uncertainty features* are that on one hand it is difficult to explicitly characterize an uncertainty set and, on the other hand, completely neglecting uncertainty and use purely reactive approaches might lead to bad solutions.

The idea is that instead of explicitly describing the uncertainty, we focus on imposing desired properties, or features, on the solution that are proven to lead to more robust solutions in terms of feasibility or to ameliorate its *recoverability*, i.e. the performance of a recovery algorithm. These features thus involve characteristics of both proactive and reactive methods dealing with robustness and recoverability. Uncertainty is considered *implicitly* through those features: no explicit uncertainty description is required. Clearly, uncertainty features are both problem and recovery algorithm dependent.

To illustrate the concepts, consider the widely studied field of airline scheduling (see [12] for a general survey). An usual technique for a solution to absorb delays is to over-estimate travel times in order to have buffers to absorb possible perturbations. So, a possible uncertainty feature would be to maximize the idle time between successive flights, which allows delay absorption and thus increases the robustness of the solution.
The stochastic model of [16] considers recourse to address the crew scheduling problem. Interestingly, the final conclusion is that the solutions tend to reduce the plane changes of crews. This gives an example of an uncertainty feature, namely measuring how well the routes of the planes match union constraints for the crew. Maximizing this uncertainty feature will, according to the conclusions of [16], increase the recoverability of the schedule.
[11] builds robust schedules for planes that allow crew swappings in case of disruptions. A corresponding uncertainty feature is the number of plane crossings, i.e. when two planes are on the ground at the same time and the same airport and other features can be derived from ad-hoc recovery algorithms ([7]).
We notice that, in many works dealing with robustness or recovery (see e.g. [13], [6], [14]), the solutions of the different approaches tend to a property that is simple, such as the number of plane crossings ([11], [5]), reducing the length of plane rotations ([9]) or increasing idle time ([2]).

2. Framework for Uncertainty Features

We start, without loss of generality, from a general deterministic minimization problem (P) given as

$$z_D^* = \min f(\mathbf{x}) \tag{1}$$
$$a(\mathbf{x}) \leq \mathbf{b} \tag{2}$$
$$\mathbf{x} \in X \tag{3}$$

Note that this problem is prone to noise in the data, but we do not formulate the nature of this noise. We transform (P) in a multi-objective optimization problem by adding an uncertainty feature (that has to be maximized) $\mu(\mathbf{x})$. Objective (1) becomes:

$$[z_D, z_M] = [\min f(\mathbf{x}), \max \mu(\mathbf{x})].$$

Note that the feasibility of solution $\mathbf{x}$ is not affected. When $\mu(\mathbf{x})$ is of the same complexity than $f(\mathbf{x})$ and $a(\mathbf{x})$, then the obtained problem is no more difficult than the initial problem (P).
There are three commonly used methods to solve multi-objective problems. One is the exploration of the Pareto frontier, the set of non dominated solutions. Another approach is to consider a weighted combination of the objectives. A third approach is to optimize one of the objectives and enforce a bound on the second with an additional constraint.

Note that uncertainty features are a simple and intuitive way to implicitly take into account the uncertainty.
In this framework, it is straightforward to consider several uncertainty features at once. In this case, either $\mu(\mathbf{x})$ is a combination of several uncertainty features $\mu_1(\mathbf{x}), \cdots, \mu_m(\mathbf{x})$ or we address a multi-objective problem with $m+1$ objectives $f(\mathbf{x}), \mu_1(\mathbf{x}), \cdots, \mu_m(\mathbf{x})$.

3. Illustrative Examples

The aim of uncertainty features is to avoid the explicit modeling of uncertainty. We show here that it is possible to use that framework to formulate both a stochastic optimization as well as a robust optimization problem.
Let U be the uncertainty set of a stochastic optimization problem, i.e. the set of possible values for the problem data, associate with a probability distribution. Clearly, if we take $\mu(\mathbf{x}) = -\mathbb{E}_{\mathbb{U}}\{f(\mathbf{x})\}$, the expectation of the cost over U, then we obtain a stochastic optimization problem. If we solve the multi-objective by optimizing only the uncertainty feature and bounding the objective $f(\mathbf{x})$ by $+\infty$, we get the usual expected cost minimization of stochastic problems.

Robust optimization can also be formulated as a particular case of uncertainty features.

Let us consider the approach of [4] for linear robust optimization with a unique constraint:

$$z^*_{ROB} = \min \mathbf{c}^T\mathbf{x} \tag{4}$$

$$\sum_j a_j x_j + \beta(\mathbf{x}, \Gamma) \leq b \tag{5}$$

$$\mathbf{x} \in X \tag{6}$$

where $\beta(\mathbf{x}, \Gamma)$ is the characterization of the worst scenario in an uncertainty set

when at most Γ coefficients a_j change. Note that the uncertainty set U contains $J \geq \Gamma$ changing coefficients $\tilde{a}_j \in [a_j - \hat{a}_j, a_j + \hat{a}_j]$ for $j \in J$. The objective is to find the solution of least cost that is feasible for all possible scenarios in U having at most Γ changing parameters. This must hold for the worst scenario, explaining the formulation (4)-(6).
We now show how to derive the above formulation using uncertainty features. The problem we start with is the feasibility problem

$$\min_{\mathbf{x} \in X} \left\{ \sum_j a_j x_j + \beta(\mathbf{x}, J) - b \right\} \tag{7}$$

The solution of this problem gives the worst violation of the constraint when all J parameters are changing. A negative solution of (7) means the solution is robust for all scenarios in U. Hence, the main concern is feasibility with respect to *all* scenarios. The uncertainty feature of a solution is given by its cost, thus we set $\mu(\mathbf{x}) = -\mathbf{c}^T\mathbf{x}$ (cost is to be minimized). In this case, we handle the multi-optimization by maximizing only the uncertainty feature $-\mathbf{c}^T\mathbf{x}$ and constraining the feasibility, imposing it not to exceed $(1+\rho)z^*_{ROB}$. Defining

$$\rho = \begin{cases} \frac{\overline{\beta}(\mathbf{x}, J-\Gamma)}{z^*_{ROB}} - 1 & \text{if } z^*_{ROB} \neq 0 \\ 0 & \text{otherwise.} \end{cases} \qquad \text{where } \overline{\beta}(\mathbf{x}, J-\Gamma) + \beta(\mathbf{x}, \Gamma) = \beta(\mathbf{x}, J).$$

We do not report here the details for brevity. The main result is that, for the linear robust problem with a single constraint, the uncertainty feature problem is equivalent to the robust formulation of [4]. Note that the extension to the case of n constraints is also possible and remarkably, the formulation with uncertainty features allows to estimate the maximal value of Γ_i, the number of varying coefficients in row i, to ensure that a robust solution exists.

In [7] we develop a recovery algorithm for airline scheduling. Uncertainty features improving the algorithms efficiency are, as described in section 1, plane crossings and idle time. Moreover, we intend to derive more specific features based directly on the recovery algorithm's networks structure and eventually consider uncertainty features for crew scheduling and recovery.

We see that uncertainty features allow to fall back to standard methods with particular choices of $\mu(\mathbf{x})$. Their advantage compared to standard methods is that they allow to implicitly exploit the structure of a recovery algorithm without increasing the complexity, as long as the uncertainty features have the same complexity than the deterministic objective and constraint functions.

4. Conclusion

In this paper, we introduce the concept of uncertainty feature to implicitly cope with uncertainty instead of modeling it explicitly with an uncertainty set. We show that the uncertainty features are a generalization of the existing methods for optimization under uncertainty, since, by choosing an uncertainty feature based on an uncertainty set and choosing an appropriate function $\mu(\mathbf{x})$, we retrieve the stochastic or the robust formulations. The advantage on existing methods is the possibility to consider reactive methods implicitly if the uncertainty feature increases recoverability.
The validation of the approach is clearly problem dependent, since different problems do not necessarily have similar structural properties. Furthermore, uncertainty features for recoverability depend on the recovery algorithm. For a specific problem, one has to measure, by simulation, the correlation between a structural property and the solution cost. A good feature is identifiable by a significative negative correlation, i.e. when an increase in one term leads to a significant decrease in the other.
An experimental comparative investigation on knapsack problems is in process, and an application to airline scheduling is planned.

References

[1] S. Albers, *Online algorithms: A survey*, Mathematical Programming, 2003, 97, p. 3-26.

[2] M.A. Al-Fawzana and M. Haouari, *A bi-objective model for robust resource-constrained project scheduling*, International Journal of Production Economics, 2005, 96, p. 175-187.

[3] A. Ben-Tal and A. Nemirovski *Lectures on Modern Convex Optimization, Analysis, Algorithma, and Engineering Applications*, MPS-SIAM Series on Optimization, 2001.

[4] Bertsimas and Sim, *The price of robustness*, OR, 2004, 52, p. 35–53.

[5] F. Bian and al., *Measuring the robustness of airline fleet schedules*, 2004, http://www.cs.nott.ac.uk/ gxk/papers/jdsmista2003sel.pdf.

[6] S. Chebalov and D. Klabjan, *Robust Airline Scheduling: Move-up Crews*, University of Illinois at Urbana-Champaign, February 2004.

[7] N. Eggenberg, M. Salani and M. Bierlaire, *Column generation methods for disrupted airline schedules*, Ecole Polytechnique Fédérale de Lausanne, Tech. Rep. TRANSP-OR 071203.

[8] W. Herroelen and R. Leus, *Project scheduling under uncertainty: Survey and research potential*, EJOR, 2005, 165, p. 289-306.

[9] J.M. Rosenberger, E.L. Johnson and G.L. Nemhauser, *A Robust Fleet Assignment Model with Hub Isolation and Short Cycles*, Transportation Science, 2004, 38, p. 357-368.

[10] Kall and Wallace, *Stochastic Programming*, 1994.

[11] D. Klabjan and al., *Airline Crew Scheduling with Time Windows and Plane-Count Constraints*, Transportation Science, 2002, vol. 36, p. 337-348.

[12] N. Kohl, A. Larsen, J. Larsen, A. Ross and S. Tiourine, *Airline Disruption Management - Perspectives, Experiences and Outlook*, Informatics and Mathematical Modelling, Technical University of Denmark, 2004, IMM-Technical Report-2004-16.

[13] S. Lan, J.-P. Clarke and C. Barnhart, *Planning for Robust Airline Operations: Optimizing Aircraft Routings and Flight Departure Times to Minimize Passenger Disruptions*, Transportation Science, 2006, 40, p. 15-28.

[14] N. Policella, *Robust Scheduling: Analysis and Synthesis of Flexible Solutions*,Universita di Roma "la Sapienza", thesis, 2004.

[15] N. V. Sahinidis, *Optimization under uncertainty: state-of-the-art and opportunities*, Computers and Chemical Engineering, 2004, 28, p. 971-983.

[16] J.W. Yen and J.R. Brige, *A Stochastic Programming Approach to the Airline Crew Scheduling Problem*, Transportation Science, 2006, 40, p. 3-14.

A dynamic service mechanic problem for a housing corporation

Maria L.A.G. Cremers [a,*] Joaquim A.S. Gromicho [b]
Willem K. Klein Haneveld [a] Maarten H. van der Vlerk [a]

[a] *Department of Operations, University of Groningen, The Netherlands*
[b] *Vrije Universiteit, Amsterdam & ORTEC, Gouda, The Netherlands*

Key words: service mechanics, stochastic programming, online optimization

1. Introduction

We consider a dynamic planning problem for the maintenance and repairment services of a housing corporation. Large maintenance activities are typically known well-ahead, while emergency incidents are urgent and unforeseen. The same pool of mechanics is used to serve both kinds of jobs. Furthermore, some jobs will need to be outsourced to subcontractors since the number of own mechanics is not sufficient to serve all jobs. In this service mechanic problem we focus on a decision to make today for the planning period of the next two weeks: which maintenance activities to assign to subcontractors and which ones to own mechanics, while taking into account the unknown emergency incidents which will arise (and need to be served) during the planning period. The decision criterion of the service mechanic problem is to minimize the expected costs of serving all jobs.

In the service mechanic problem five aspects are important.

- Probabilistic information on the emergency incidents is used when making decisions on the maintenance activities.
- We decide whether or not to outsource jobs to subcontractors.
- For each incident, besides the assignment decision it is also decided during which time slot(s) the incident is served. Every incident has its own due time, not later than the end of the planning period.
- Not every mechanic is able to serve every job (depending on skill type).
- The routing of mechanics is not considered.

* Corresponding author.

In the literature, very little attention has been paid to the service mechanic problem described above. While we focus on the uncertainty with respect to emergency incidents, by using probabilistic information, most related papers ignore future incidents when making decisions. Furthermore, we explicitly decide whether or not to subcontract jobs, while most related papers do not even consider the possibility of outsourcing. Instead, they include routing of mechanics / vehicles while we exclude travel times.

For example, Johns [5] and Madsen et al. [7] discuss the problem of scheduling repair men in which future jobs are not considered when deciding on the known jobs. Furthermore, outsourcing is not allowed, only one type of mechanic is considered, and routing is performed. Also in the grocery delivery problem of Campbell and Savelsbergh [3] future jobs are ignored when deciding on the start time and routing of deliveries. The vehicle fleet is homogeneous and although deliveries can be rejected, this is not a decision to be made. Rejected jobs can be regarded to be outsourced to subcontractors afterwards. In the routing and selection problem of Bolduc et al. [2] and Chu [4] outsourcing of jobs is modeled as decisions. However, they study the deterministic variant of the problem with one type of own vehicle (skill type); routing of the own fleet is not considered.

2. Problem description

In the service mechanic problem we consider two kinds of jobs for mechanics: large maintenance activities which are known well before they start and urgent emergency incidents which gradually become known during the planning period. All jobs need to be served.

Each job requires a specific type of mechanic; either a handyman or an expert. Experts are able to serve jobs requiring handymen, but the reverse is not possible. The numbers of available handymen and experts are given and may vary during the planning period of two weeks. In addition to the own mechanics (handymen and experts) subcontractors can be used to outsource any kind of job. We assume that sufficiently many subcontractors are available.

For (maintenance) activities a start and end time is given, as well as the required type and number of mechanics. The data of (emergency) incidents consists of the required type and number of mechanics, the arrival time, the due time (time before which the incident has to be served), and the duration. Locations of the jobs are not necessary since we do not take travel times into account.

The objective of the service mechanic problem is to minimize the expected costs of serving all jobs. First of all, subcontracting today (before the start of the planning period) is less expensive than during the planning period. Furthermore, since the labor costs of the own mechanics have to be paid regardless of whether the mechanics actually work, using own mechanics is free of charge. The costs of subcontracting a job are proportional to the duration of the job and the required number of me-

chanics. Moreover, jobs requiring experts are more expensive to subcontract than those requiring handymen. We assume that the contracts with subcontractors have already been made so that there are no fixed costs for outsourcing.

In addition to the non-preemptive version, we also consider the preemptive service mechanics problem in which interruption of maintenance activities is allowed in order to serve incidents.

3. Two-stage recourse model

To model the service mechanic problem we have developed a two-stage recourse model as known in stochastic programming [1], [6], [8]. The first stage models today when the data of the maintenance activities are completely known and probabilistic information on the emergency incidents is assumed. This is consistent with most two-stage recourse models in which the second stage typically consists of a (static) optimization problem like a mixed-integer linear programming problem. However, our model is non-standard since the second stage is a *dynamic* problem, *simulating* the planning period of two weeks.

In the first stage all maintenance activities for the next two weeks are assigned to mechanics. The assignment of activities to subcontractors is permanent. Thus, these activities can not be reconsidered and therefore do not appear in the second-stage problem. In contrast, the assignment of activities to own mechanics is preliminary since during the planning period it can be reconsidered.

The objective of the service mechanic problem is to find the assignment with minimal (expected) costs. The costs of an assignment consists of two parts: the costs of assigning maintenance activities to subcontractors today (the first-stage costs) and the expected costs of (re-)assigning activities and incidents to subcontractors during the planning period. In the previous section we have already explained that only the costs of subcontracting are considered. To find an assignment with small (expected) costs, we use a genetic algorithm.

In the second stage, for a given first-stage assignment and given realization of the emergency incidents, an event simulation is applied. In this simulation, two online decisions are made. First, after the arrival of an incident a start time has to be determined, in such a way that the due time is respected. Furthermore, all preliminarily assigned activities and emergency incidents need to be assigned to mechanics, either own mechanics or subcontractors, at minimal costs. Although it would probably be best to make both decisions simultaneously, due to restrictions on CPU time the two decisions are made sequentially.

In this paper, we will investigate three different simulation strategies. The strategies differ in the way they find suitable start times for the incidents and whether or not preemption of activities is allowed. In the first strategy, *Simple*, preemption of maintenance activities is not allowed. The start time of incidents is set equal to the

arrival time. We expect this strategy to be fast, but to give results of low quality, compared to the other two strategies. In strategy *Search* preemption of activities is also not allowed. Now, the start time of incidents is set equal to the earliest possible time when there are enough own mechanics available. If this results in exceeding the due time, the start time is set equal to the arrival time and at least one incident or activity is outsourced. To determine which one is assigned to a subcontractor a greedy heuristic is applied. Preemption of maintenance activities is allowed in simulation strategy *Preemptive*. Rules regarding e.g. the number of times an activity can be interrupted and the length of the interruption are determined. Following these rules the start time of incidents is set as early as possible.

4. Current research

A genetic algorithm is used to solve the entire two-stage recourse model. Preliminary results will be presented for all three simulation strategies, based on randomly generated data sets.

References

[1] J.R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, New York, 1997.

[2] M.-C. Bolduc, J. Renaud, and F. Boctor. A heuristic for the routing and carrier selection problem. *European Journal of Operational Research*, 183:926–932, 2007.

[3] A.M. Campbell and M.W.P. Savelsbergh. Decision support for consumer direct grocery initiatives. *Transportation Science*, 39:313–327, 2005.

[4] C.-W. Chu. A heuristic algorithm for the truckload and less-than-truckload problem. *European Journal of Operational Research*, 165:657–667, 2005.

[5] S. Johns. Heuristics to schedule service engineers within time windows. *Journal of the Operational Research Society*, 46:339–346, 1995.

[6] P. Kall and J. Mayer. *Stochastic Linear Programming: Models, Theory, and Computation*. Kluwer Academic Publishers, New York, 2005.

[7] O.B.G. Madsen, K. Tosti, and J. Vælds. A heuristic method for dispatching repair men. *Annals of Operations Research*, 61:213–226, 1995.

[8] A. Ruszczynski and A. Shapiro, editors. *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*. North-Holland, 2003.

Special Cases of Online Parallel Job Scheduling [1]

Johann L. Hurink [a] Jacob Jan Paulus [a,*]

[a] *University of Twente, P.O. box 217, 7500AE Enschede, The Netherlands*

Key words: Scheduling Parallel Jobs, Online Algorithms, Competitive Analysis

1. Introduction

In this talk we consider special cases of online scheduling jobs which require processing on a number of machines simultaneously (parallel jobs). Jobs are characterized by their processing time p_j and the number of machines m_j simultaneously required for processing, and are presented one by one to a decision maker. As soon as a job becomes known, it has to be scheduled irrevocably (i.e. its start time has to be set) without knowledge of successive jobs. Preemption is not allowed and the objective is to minimize the makespan. We study a number of special cases of this online problem.

In contrast to an online algorithm, an *offline* scheduling algorithm has complete knowledge of the list of jobs to construct the optimal offline schedule. This optimal offline objective value is used to measure the quality of online algorithms. An online algorithm is ρ-competitive if for any list of jobs it produces a schedule with makespan at most ρ times the makespan of the optimal offline schedule. An online problem is called semi-online if there is some a priori knowledge of the list of jobs, e.g., the jobs appear in non-increasing order of machine requirement. Because of such knowledge smaller competitive ratios might be obtained.

Using the three-field scheduling problem notation, the considered problem is denoted by $P|\mathrm{online}-\mathrm{list}, m_j|C_{\max}$ see [7]. In the literature the concept of parallel jobs is known by many different names, such as *parallel tasks*, *parallelizable tasks*, *multiprocessor tasks*, *multiple-job-on-one-processor*, and *1-job-on-r-processors*. In some literature the machine requirement m_j of a job is called the width or the size of a job. And in stead of m_j the term $size_j$ or simply s_j is used to denote the parallel machine requirement of job j.

There is a great deal of similarity between $P|\mathrm{online}-\mathrm{list}, m_j|C_{\max}$ and the online

* Corresponding author.

[1] Part of this research has been funded by the Dutch BSIK/BRICKS project.

orthogonal strip packing problem. The orthogonal strip packing problem is a two-dimensional packing problem. Without rotation rectangles have to be packed on a strip with fixed width and unbounded height. The objective is to minimize the height of the strip in which the rectangles are packed. In the online setting one rectangle is presented after the other and has to be assigned without knowledge of successive rectangles. To see the similarity, let each machine correspond to one unit of the width of the strip, and time to the height of the strip. The width of a rectangle j corresponds to the machine requirement of job j and its height to the processing time. Minimizing the height of the strip used is equivalent to minimizing the makespan of the machine scheduling problem. The difference lies in the choice of machines. In $P|\text{online} - \text{list}, m_j|C_{\max}$ any m_j machines suffice for job j, where rectangles can not be split up into several rectangles together having width m_j. Therefore, algorithms for strip packing can be used for parallel job scheduling [5], but in general not the other way around.

2. Overview of Results

We give an overview of the current state of the research on online parallel job scheduling, and its various semi-online versions. The results are summarized in Table 1. The first online algorithm for online parallel job scheduling with a constant competitive ratio is presented in [7] and is 12-competitive. In [12], an improvement to a 7-competitive algorithm is given. This *dynamic waiting algorithm* schedules jobs with a small machine requirement greedily and delays the jobs with a large machine requirement. For the strip packing problem in [1] a 6.99-competitive online algorithm is given under the assumption that jobs have a processing time of at most 1. This *shelf algorithm* groups rectangles of similar height together. The currently best known algorithm is designed by combining the ideas of separating jobs with large and small machine requirement, and using a shelf structure. This results in a 6.6623-competitive algorithm which is independently obtained in [5] and [10], and due to its structure it can be applied to online orthogonal strip packing as well.

For $P|\text{online} - \text{list}, m_j|C_{\max}$ the best known analytical lower bound on the competitive ratio is a bound of 2 resulting from the strip packing problem [2], which applies directly to the parallel job problem with $m \geq 3$. In [6] a tight lower bound of 2 is given for the two machines case. Furthermore, a computerized proof, based on an ILP-formulation, resulting in a lower bound of 2.43 for $P|\text{online} - \text{list}, m_j|C_{\max}$ is given.

Until now, the best known algorithm for the case with 3 machines is the 3-competitive greedy algorithm. In this talk we show an improved algorithm:
Theorem 1 *For $P3|\text{online} - \text{list}, m_j|C_{\max}$ a 2.8-competitive algorithm exists.*

In the literature a number of semi-online variants of online parallel job scheduling are considered. In case the jobs appear in non-increasing order of machine requirement the best known lower bound is 1.88 from classical parallel machine

$P\|\text{online} - \text{list}, m_j\|C_{\max}$		
Model	Lower Bound	Upper Bound
-	2.43, [6]	6.6623, [5, 10]
$m = 2$	2, [6]	2, (Greedy)
$m = 3$	2, [2]	2.8, (This talk)
$3 \leq m \leq 6$	2, [2]	m, (Greedy)
Semi-online $P\|online - list, m_j\|C_{max}$		
Model	Lower Bound	Upper Bound
-non-increasing m_j	1.88, [9]	2.4815, (This talk)
$m = 2$ or 3	$2 - \frac{1}{m}$, [4]	$2 - \frac{1}{m}$ (Greedy)
$m = 4$ or 5	-	2 (Greedy)
-non-increasing p_j	$\frac{5}{3}$, [2]	2, [11]
$m = 2$	$\frac{9}{7}$, [3]	$\frac{4}{3}$, [3]
-non-decreasing p_j	-	-
$m = 2$	$\frac{3}{2}$, [4]	$\frac{3}{2}$, [3]

Table 1
Results on online scheduling of $P|\text{online} - \text{list}, m_j|C_{\max}$

scheduling, i.e. this bound uses only jobs with $m_j = 1$ [9]. Furthermore, for this case in [11] it is shown, that greedy scheduling the jobs is 2.75-competitive and no better than 2.5-competitive. In this talk we show that slightly modifying the greedy algorithm yields a better algorithm.

Theorem 2 *For $P|\text{online} - \text{list}, m_j|C_{\max}$ with jobs appearing in non-increasing order of machine requirement, a 2.4815-competitive algorithm exists.*

Furthermore, we show that for 2 and 3 machines and jobs appearing in non-increasing order of machine requirement the greedy algorithm is $(2 - \frac{1}{m})$-competitive. As we know from classical parallel machine scheduling [4], this is the best possible; these bounds are tight. Finally, we show that for 4 and 5 machines greedy is 2-competitive.

In case the jobs appear in non-increasing order of processing time a greedy algorithm is 2-competitive [11]. The best know lower bound on the competitive ratio is $\frac{5}{3}$ from the strip packing problem [2]. For the two machine case with non-increasing processing times a lower bound of $\frac{9}{7}$ and a $\frac{4}{3}$-competitive online algorithm are known [3]. For the case where jobs appear in non-decreasing order of processing times and two machines, an optimal (best possible ratio) $\frac{3}{2}$-competitive algorithm is given in [3]. Optimality follows from a lower bound from classical parallel machine scheduling [4].

The results, summarized in Table 1, show that in only a few special cases the gap between the lower and upper bound on the competitive ratio is closed. In particular the gap for the general problem $P|\text{online} - \text{list}, m_j|C_{\max}$ is large.

References

[1] B.S. BAKER AND J.S. SCHWARZ (1983), *Shelf Algorithms for Two-Dimensional Packing Problems*, SIAM Journal on Computing, **12**, 3, 508-525.

[2] D.J. BROWN, B.S. BAKER AND H.P. KATSEFF (1983), *Lower bounds for on-line two-dimensional packing algorithms*, Acta Informatica, **18**, 2, 207-225.

[3] W.T. CHAN, F.Y.L. CHIN, D. YE, G. ZHANG AND Y. ZHANG (2007), *On-Line Scheduling of Parallel Jobs on Two Machines*, Journal of Discrete Algorithms (to appear).

[4] U. FAIGLE, W. KERN AND G. TURÀN, *On the performance of online algorithms for partition problems*, Acta Cybernetica, **9**, 107-119.

[5] J.L. HURINK AND J.J. PAULUS (2008), *Online Algorithm for Parallel Job Scheduling and Strip Packing*, Lecture Notes in Computer Science (Proceedings of WAOA 2007) to appear.

[6] J.L. HURINK AND J.J. PAULUS (2008), *Online Scheduling of Parallel Jobs on Two Machines is 2-Competitive*, Operations Research Letters, **36**, 1, 51-56.

[7] JOHANNES, B. (2006), *Scheduling parallel jobs to minimize the makespan*, Journal of Scheduling, **9**, 5, 433-452.

[8] D.S. JOHNSON, A. DEMERS, J.D. ULLMAN, M.R GAREY AND R.L. GRAHAM (1974), *Worst-case performance founds for simple one-dimensional packing algorithms*, SIAM Journal on Computing, **3**, 299-325.

[9] J.F. RUDIN III (2001), *Improved Bound for the Online Scheduling Problem*, The University of Texas at Dallas.

[10] D. YE, X. HAN AND G. ZHANG (2007), *A note on online strip packing*, Journal of Combinatorial Optimization, to appear.

[11] D. YE AND G. ZHANG (2004), *On-line Scheduling of Parallel Jobs*, Lecture Notes in Computer Science (SIROCCO 2004), **3104**, 279-290.

[12] D. YE AND G. ZHANG (2007), *On-line scheduling of parallel jobs in a list*, Journal of Scheduling, **10**, 6, 407-413.

Graph theory (II)

Room A, Session 4

Wednesday 14, 9:00-10:30

Percolation on sparse random graphs with given degree sequence

Nikolaos Fountoulakis [1,*]

School of Mathematics
University of Birmingham
Edgbaston, B15 2TT
United Kingdom

Key words: random graphs, percolation, degree sequences

1. Introduction

Traditionally percolation theory has been the study of the properties of a random subgraph of an infinite graph, that is obtained by deleting each edge of the graph with probability $1-p$ for some $p \in (0,1)$ independently of every other edge. The question that has been mainly investigated is whether the subgraph that is spanned by these edges has an infinite component or not. The classical type of graphs that was studied in percolation theory is the lattice $\mathbb{Z}^d$ in various dimensions $d \geq 2$ (see [8]). Various other types of lattices have also been studied. In each of the above cases the main problem is the calculation of a critical p_c so that if $p < p_c$ then the random subgraph obtained as above has no infinite components, whereas if $p > p_c$ there is an infinite component with probability 1.

In the present work, we study percolation on finite graphs whose number of vertices is large. This problem is old, in the sense that for example a $\mathcal{G}_{n,p}$ random graph is a random subgraph of the complete graph on n vertices, where each edge appears with probability p independently of every other edge. In this context, a question about the appearance of an infinite component is senseless. A somehow analogous question is whether there exists a component of the random subgraph containing a certain proportion of the vertices or as we customarily say a *giant component*. More specifically, if the original graph has n vertices the question now is whether there exists an $\varepsilon > 0$ for which there is a component of the random subgraph that has at least εn vertices with probability $1 - o(1)$ (as $n \to \infty$). Hence, we also ask (quite informally) for the existence of a critical p_c for which whenever $p < (1-\delta)p_c$ then

* Corresponding author.
[1] The author is supported by the EPSRC, grant no. EP/D50564X/1

for every $\varepsilon > 0$ there is no component having at least εn vertices with probability $1 - o(1)$ and whenever $p > p_c(1 + \delta)$ then there is a component with at least εn vertices for some $\varepsilon > 0$ with probability $1 - o(1)$. A classical example of this is the $\mathcal{G}_{n,p}$ random subgraph of K_n, the complete graph on n vertices, where as it was proved by Erdős and Rényi in [5] the critical probability is equal to $1/n$ (see also [2] or [9] for a detailed discussion).

More generally, Bollobás, Kohayakawa and Łuczak in [3] raised the following question: given a sequence of graphs $\{G_n\}$ whose order tends to infinity as n grows, is there such a phase transition? Assume that G_n has $|G_n|$ vertices and e_n edges. For each such n we have a probability space on the set of spanning subgraphs of G_n and the probability of such a subgraph of G_n that has e edges is $p^e(1-p)^{e_n - e}$, where e_n is the number of edges of G_n. Let $G_n(p)$ be a sample from this probability space. Thus we are seeking a p_c such that: if $p < (1-\delta)p_c$, then for every $\varepsilon > 0$ as $n \to \infty$ all the components of $G_n(p)$ have at most $\varepsilon|G_n|$ vertices with probability $1 - o(1)$, and if $p > p_c(1+\delta)$, then there exists $\varepsilon = \varepsilon(p) > 0$ for which the largest component of $G_n(p)$ has at least $\varepsilon|G_n|$ vertices with probability $1 - o(1)$. If the sequence of graphs is $\{K_n\}$, this is simply the case of a $\mathcal{G}_{n,p}$ random graph.

2. The main result

In the present paper, we determine a percolation threshold in the case where the sequence $\{G_n\}_{n\in\mathbb{Z}^+}$ is a sequence of sparse random graphs on n vertices. In particular, for every integer $n \geq 1$, G_n is a uniformly random graph on the set $V_n = \{1, \ldots, n\}$ having a given degree sequence $\mathbf{d}(n) = (d_1, \ldots, d_n)$, i.e. for $i = 1, \ldots n$ vertex i has degree d_i. More formally, a *degree sequence* on the set V_n is a vector $\mathbf{d} = (d_1, \ldots, d_n)$ consisting of natural numbers, where $\sum_{i=1}^n d_i$ is even. We let $2M$ denote this sum, and $M = M(n)$ is the number of edges that $\mathbf{d}$ spans. For a given $\mathbf{d} = \mathbf{d}(n)$, if $\mathbf{d}(n) = (d_1, \ldots, d_n)$ for $n \in \mathbb{Z}^+$, we set $D_i = D_i(n) = |\{j \in V_n \ : d_j = i\}|$, for all $i \in \mathbb{N}$, and $\Delta = \Delta(n) = \max_{1\leq i\leq n}\{d_i\}$. Finally, if G is a graph on V_n, then $D(G)$ denotes its degree sequence.

An *asymptotic degree sequence* is a sequence $(\mathbf{d}(n))_{n\in\mathbb{Z}^+}$, where for each $n \in \mathbb{Z}^+$ the vector $\mathbf{d}(n)$ is a degree sequence on V_n. An asymptotic degree sequence is *sparse*, if for every $i \in \mathbb{N}$, we have $\lim_{n\to\infty} D_i(n)/n = \lambda_i$, for some $\lambda_i \in [0, 1]$, where $\sum_{i\geq 0} \lambda_i = 1$, and moreover

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i\geq 1} i(i-2)D_i(n) = \sum_{i\geq 1} i(i-2)\lambda_i < \infty. \tag{1}$$

The *generating polynomial* of a sparse asymptotic degree sequence is defined as $L(s) = \sum_{i=0}^{\infty} \lambda_i s^i$. We assume that every asymptotic degree sequence $(\mathbf{d}(n))_{n\in\mathbb{Z}^+}$ we work with is such that for every n the set of simple graphs that have $\mathbf{d}(n)$ as their degree sequence is non-empty.

We consider two types of percolation. Firstly, for some $p \in (0, 1)$, each edge of

G_n is present with probability p independently of every other edge. This type of percolation is usually called *bond percolation*, in that we randomly delete the edges (i.e. the bonds) of G_n. This is distinguished from another type of percolation which is called *site percolation*. Here, we go through the vertices of G_n and we make each of them isolated with probability $1-p$, independently of every other vertex (or as we say we delete this vertex). The random subgraph in this case is the spanning subgraph of G_n that does not contain the edges that are attached to the vertices that were deleted. The terms "bond" and "site" percolation have their origins in the percolation theory of infinite graphs (see [8] for an extensive discussion on both types as well as the references therein).

We shall now define the percolation threshold in each of the above cases. Let $G'(n)$ denote the random subgraph that is obtained in either case and let $L_1(G'(n))$ be the lexicographically first component of $G'(n)$ (this is the component that has maximum order and the smallest vertex it contains is smaller than the smallest vertex of every other component of maximum order - the comparison between the vertices is by means of the total ordering on V_n). Starting from the bond percolation we set $p_c^{bond} = \sup\{p \in [0,1] : \ |L_1(G'(n))|/n \xrightarrow{p} 0 \text{ as } n \to \infty\}$ (the symbol $\xrightarrow{p}$ denotes convergence in probability, i.e. we say that $X_n \xrightarrow{p} 0$ if for every $\varepsilon > 0$ we have $\mathbb{P}[|X_n| > \varepsilon] \to 0$ as $n \to 0$). The convergence in probability is meant with respect to the sequence of probability spaces indexed by the set $\mathbb{Z}^+$, where for each $n \in \mathbb{Z}^+$ the probability of a certain spanning subgraph is the probability that this is the subgraph which is spanned by the edges that survive the random deletion of the edges of the random graph G_n. Similarly, in the case of site percolation we define $p_c^{site} = \sup\{p \in [0,1] : \ |L_1(G'(n))|/n \xrightarrow{p} 0 \text{ as } n \to \infty\}$, where $G'(n)$ is now the spanning subgraph of G_n that is the outcome of the deletion of those edges that attached to the chosen vertices, i.e. the vertices that we make isolated. Note that in both cases there are two levels of randomness.

If G_n is a random d-regular graph on V_n, for any fixed $d \geq 3$, the bond percolation threshold has been calculated by Goerdt in [6] and is equal to $1/(d-1)$. Before this, bond percolation in random regular graphs was studied by Nikoletseas, Palem, Spirakis and Yung in [11], where it was proved that the critical probability is at most $32/d$, for d large enough. Also, Nikoletseas and Spirakis in [10] and Goerdt and Molloy [7] study the edge expansion properties of the giant component that remains after the edge deletion process. However, these papers did not provide any analysis on the site percolation process. Our main theorem involves also the latter and is stated as follows:

Theorem 1 *If $(\mathbf{d}(n))_{n\in\mathbb{Z}^+}$ is a sparse asymptotic degree sequence of maximum degree $\Delta(n) \leq n^{1/9}$ and $L(s)$ is its generating polynomial which is twice differentiable at 1 and moreover $L''(1) > L'(1)$, then $p_c^{site} = p_c^{bond} = L'(1)/L''(1)$. Moreover, whenever $p > p_c^{bond}$ ($p > p_c^{site}$, respectively) there exists an $\varepsilon > 0$ such that $|L_1(G'(n))| > \varepsilon n$ with probability $1 - o(1)$.*

The formula for both critical probabilities was obtained by Dorogovtsev and Mendes in [4] using qualitative (i.e. non-rigorous) arguments.

To make the statement of the above theorem slightly clearer, let us consider the

case of bond percolation (the case of site percolation is similar). Let $\mathcal{G}(n)$ be the set of graphs on V_n whose degree sequence is $\mathbf{d}(n)$. Each graph $G \in \mathcal{G}(n)$ gives rise to a probability space which consists of all its spanning subgraphs. In particular, if G has e edges and G' is a spanning subgraph of G that has $e' \leq e$ edges then its probability is $p^{e'}(1-p)^{e-e'}$; let $\mathbb{P}_p^G[\cdot]$ denote this measure. In other words, this space accommodates the outcomes of the bond percolation process applied to G and we call it the *percolation space* of G. For any $\varepsilon \in (0,1)$ we let $g_\varepsilon(G)$ be the set of all spanning subgraphs of G whose largest component has at least εn vertices. This event has probability $\mathbb{P}_p^G[g_\varepsilon(G)]$ in the percolation space of G. Now, assume that $p < p_c^{bond}$. Theorem 1 implies that for any given $\rho \in (0,1)$, the event $\{G \in \mathcal{G}(n) \ : \ \mathbb{P}_p^G(g_\varepsilon(G)) < \rho\}$ occurs with probability $1-o(1)$ in the uniform space $\mathcal{G}(n)$. That is, asymptotically for almost every graph in $\mathcal{G}(n)$ the random deletion of the edges leaves a component of order at least εn with probability no more than ρ. If $p > p_c^{bond}$, then the second part of the theorem implies that there exists $\varepsilon > 0$ such that the event $\{G \in \mathcal{G}(n) \ : \ \mathbb{P}_p^G(g_\varepsilon(G)) > 1-\rho\}$ occurs with probability $1-o(1)$ in $\mathcal{G}(n)$. Hence, as $n \to \infty$ almost all graphs in $\mathcal{G}(n)$ are such that if we apply the bond percolation process to them with retainment probability p, then there is a component having at least εn vertices with probability at least $1-\rho$ (in the percolation space).

3. Sketch of the proof of Theorem 1

The fact that the critical probabilities coincide reflects a behaviour that is similar to that of percolation on an infinite regular tree. Of course in that context the critical probabilities are defined with respect to the appearance of an infinite component that contains the (vertex that has been selected as the) root. Using the fundamental theorem of Galton-Watson processes (see for example [1]), it can be shown that the bond and the site critical probabilities coincide and they are equal to $1/(d-1)$, where d is the degree of each vertex of the tree. Observe that for the case of a random d-regular graph Theorem 1 implies that $p_c^{site} = p_c^{bond} = 1/(d-1)$. This is not a coincidence as it is well-known that a random d-regular graph locally (e.g. at distance no more than i from a given vertex for some fixed i) looks like a d-regular tree.

More generally, the typical local structure of the class of random graphs we are investigating is also tree-like. Note that the ratio $L''(1)/L'(1)$ equals

$$\sum_{i=2}^{\infty}(i-1)\frac{i\lambda_i}{\sum_{j=1}^{\infty} j\lambda_j}. \tag{2}$$

Consider a vertex $v \in V_n$ which has positive degree and let us examine more closely the behaviour of one of its neighbours. It can be shown that the probability that this has degree i is proportional to $iD_i(n)$. In particular, it is almost equal to $\frac{iD_i(n)}{\sum_i iD_i(n)}$ and this tends to $\frac{i\lambda_i}{\sum_{j=1}^{\infty} j\lambda_j}$ as n grows. Moreover, one can show that with probability $1-o(1)$ there are no edges between the neighbours of v. Therefore (2) is the limit

of the expected number of children a neighbour of v has. This scenario is repeated for every vertex in the d-th neighbourhood of v, where d is fixed. More precisely, the vertices which are at distance no more than d induce a tree rooted at v which contains at most $\ln\ln\ n$ vertices, with probability $1 - o(1)$. Suppose that there are t_i vertices of degree i in this tree. Thus for a vertex that is at distance d from v, the probability that it has degree i is proportional to $i(D_i(n) - t_i) = i\lambda_i n(1 - o(1))$. More precisely, it is

$$\frac{i(D_i(n) - t_i)}{\sum_j j(D_j(n) - t_j)}.$$

Since $\Delta \leq n^{1/9}$ and $t_i \leq \ln\ln n$, it follows that $\sum_i it_i \leq \ln\ln n \sum_{i\leq n^{1/9}} i = O(n^{1/3})$. Hence, the limit of the above probability as $n \to \infty$ is again $\frac{i\lambda_i}{\sum_{j=1}^{\infty} j\lambda_j}$ and (2) gives the limiting expected number of children of such a vertex. In other words, the graph that is induced by the vertices which are at distance no more than d from v behaves like the tree of a branching process that started at v, with the ratio $L''(1)/L'(1)$ being the expected progeny of each vertex. Observe here that the condition $L''(1) > L'(1)$ implies that in fact this is a supercritical branching process which yields an infinite tree with probability 1.

Therefore, at least locally either bond or site percolation is essentially percolation on such a random rooted tree. In both types of percolation, if $p < L'(1)/L''(1)$, then the expected number of children of a vertex that survive is $pL''(1)/L'(1) < 1$. Thus the random tree that is developing around v after the random failures of the edges or the vertices will be distributed as the tree of a subcritical branching process. In particular, the tree that surrounds most of the vertices will be cut off from the rest of the graph at a relatively small depth. On the other hand, if $p > L'(1)/L''(1)$ a large proportion from each of these local trees is preserved and moreover they are big enough to guarantee that there are enough edges going out of them. So eventually there is a fair chance that some of them are joined together and form a component of linear order.

References

[1] K.B. Atreya and P.E. Ney, *Branching Processes*, Springer, 1972.

[2] B. Bollobás, *Random Graphs*, Cambridge University Press, 2001.

[3] B. Bollobás, Y. Kohayakawa and T. Łuczak, The evolution of random subgraphs of the cube. *Random Structures and Algorithms* **3** (1992), 55–90.

[4] S.N. Dorogotsev and J.F.F. Mendes, Evolution of networks, *Advances in Physics* **51** (2002), 1079–1187.

[5] P. Erdős and A. Rényi, On the evolution of random graphs, *Publication of the Mathematical Institute of the Hungarian Academy of Sciences* **5** (1960), 17–61.

[6] A. Goerdt, The giant component threshold for random regular graphs with edge faults, *Theor. Comput. Sci.* **259** (2001), 307–321.

[7] A. Goerdt and M. Molloy, Analysis of edge deletion processes on faulty random regular graphs, In *Proceedings of the 4th Latin American Theoretical Informatics Symposium 2000*, 38–47.

[8] G.R. Grimmett, *Percolation*, 2nd edn., Grundlehren der mathematischen Wissenschaften, vol. **321**, Springer-Verlag, Berlin, 1999.

[9] S. Janson, T. Łuczak and A. Ruciński, *Random Graphs*, Wiley Interscience, 2000.

[10] S. Nikoletseas, K. Palem, P. Spirakis and M. Yung, Vertex disjoint paths and multiconnectivity in random graphs: Secure network computing. In *Proceedings ICALP 1994* LNCS 820, 508–519.

[11] S. Nikoletseas and P. Spirakis, Expansion properties of random regular graphs with edge faults. In *Proceedings STACS 1995* LNCS 900, 421–432.

On co-distance hereditary graphs

S. Dubois [a] V. Giakoumakis [a,*] and C.B. Ould El Mounir [a]

[a] *MIS, Université d'Amiens, France*

Abstract

We present a linear time recognition algorithm as well as a 4-expression for calculating the clique-width for the co-distance hereditary graphs which is the complementary class of the well known family of distance hereditary graphs.

Key words: Distance and co-distance hereditary graphs, split decomposition, clique-width, linear recognition algorithm

1 On distance hereditary graphs

For terms not defined here the reader is referred to [1]. Given a graph $G = (V, E)$, V will denote its vertex set, E its edge set and $N(x)$ the neighborhood of $x \in V$. A vertex having exactly one neighbor is called a *pendant* vertex. Two vertices x and y are *twins* iff $N(x) = N(y)$, they are *true* twins iff $(xy) \in E$ and *false* twins iff $(xy) \notin E$. The *distance* between two vertices x and y, denoted by $d_G(x, y)$, is the length of a shortest path between x and y. The class of Distance Hereditary (DH for shortly) graphs have been widely studied and many results have been obtained for these graphs (see [1]). Among them we recall that DH graphs are totally decomposable using *split decomposition*. We recall also that DH graphs are also known as HHDG-free graphs since they can be characterized by four forbidden configurations: the House (i.e. the complementary graph of a chordless chain of 5 vertices or P_5), the Hole (i.e. a chordles cycle of at least five vertices), the Domino (i.e. a cycle of 6 vertices $abcdef$ having exactly one chord cf) and the Gem (i.e. the graph formed by a $P_4 = abcd$ and a universal vertex w.r.t. this P_4). Finally, a graph G is distance hereditary iff for any connected subgraph H of G $d_G(x, y) = d_H(x, y)$ holds for every pair of vertices of H. A *pruning sequence* (S, σ) of G is a total ordering $\sigma[x_1, \ldots, x_n]$ of its vertex set and a sequence $S[s_1, \ldots, s_n]$ of triples, such that for $1 \leq i \leq n-1$ and $i < j$, in the induced subgraph G_i of $G[V \setminus \{x_1, \ldots, x_{i-1}\}]$, s_i is one the following words :(x_i, P, x_j), if $N(x_i) = \{x_j\}$ (x_j is a pendant vertex) or (x_i, F, x_j) (x_i and x_j are false twins) or (x_i, T, x_j) (x_i and x_j are true twins). The pruning sequence is used for the recognition of a DH graph G: starting from a vertex of G we construct successively subgraphs of G by adding true twins, false twins or pendant vertices. In [5] was

* Corresponding author.

proposed a linear time recognition algorithm for DH graphs by constructing the corresponding pruning sequence. But this algorithm recognizes also the domino and the house as DH graphs, problem that was resolved in [3]. The recognition algorithm of DH graphs presented in [3] will be for us the framework for the recognition of a co-distance hereditary graph G. We shall show in the next section that for testing if $\overline{G}$ is distance hereditary we do not need to compute $\overline{G}$ but we can work on G and make the necessary transformations to this algorithm in order to remain in linear time on the size of G. We give below the recognition algorithm in [3] and we enumerate the 6 steps needed these transformations.

Algorithm 1 ([3]). **The pruning sequence of a connected DH graph**
Build the distance levels L_v= $L_1, \ldots, L_k$ from a vertex v of G **(1)**; $j \leftarrow 1$;
For $i = k$ downto 1 **do**
For every connected component C of $G[L_i]$ **(2) Do**
$z \leftarrow$ *Prune-cograph* (C, j) **(3)** ; $j \leftarrow j+ \mid C \mid -1$**; End_For**
Sort the vertices of L_i by increasing *inner* degree **(4)**
For every vertex x of L_i having exactly one neighbor y **(5) do**
$\sigma(j) \leftarrow y$ and $s_j \leftarrow (xPy)$; $j \leftarrow j + 1$**; End_For**
For every vertex $x \in L_i$ taken in increasing inner degree order **Do**;
$y \leftarrow$ *Prune-cograph* $(G[N_{i-1}(x)], j)$ **(6)**; $j \leftarrow j+ \mid N_{i-1}(x) \mid -1$; $\sigma(j) \leftarrow x$ and $s_j \leftarrow (xPy)$; $j \leftarrow j + 1$; **End_For**
End_For

Let us explain some terms used in the above algorithm. Let G be a connected graph and v be a vertex of G. A *distance level* L_v in G is the set $L_1, \ldots, L_k$ of vertices of G such that $x \in L_i$ if $d_G(v, x) = i$. For every vertex x of G and for every integer i such that $1 \leq i \leq k$, we denote by $N_i(x) = N(x) \cap L_i$. The inner degree of x is the cardinality of $N_i(x)$. The algorithm *Prune-cograph* (C, j) constructs the pruning sequence (S, σ) of the cograph corresponding to the connected component C and contracts C to the last vertex z of σ. We must point out that *Prune-cograph* (C, j) works on the cotree $T(C)$ corresponding to the cograph C and by [2] a cotree can been obtained in linear time on the size of the considered cograph.

2 Linear recognition of co-distance hereditary graphs

We shall show now how the recognition algorithm of DH graphs presented in previous section can be transformed in order to recognize a co-distance hereditary (co-DH for shortly) graph G in linear time on the size of G. We assume that G is a connected graph as well as $\overline{G}$. If $\overline{G}$ is not connected we shall work in each connected component of $\overline{G}$. Let us explain now how we can process the 6 steps in algorithm 1 in $\overline{G}$ using the edges of the graph G.

Step 1. ***Algorithm 2 : constructing all distance levels*** L_v ***of a connected component of*** $\overline{G}$
Input : A graph G with n vertices, a list $L = \{1, ..., n\}$ of all vertices of G and an array *index*$[1..n]$ such that index$[i] = 0$ for all i.
Output : The set $L_v = \{L_1, ..., L_k\}$ of distance levels from a vertex v of a connected component of $\overline{G}$.
$i := 0$; Pick an arbitrary vertex x of L, $L_i := x$ and delete it from L.
While L is non empty **Do**

[For every vertex $y \in L_i$, and for every vertex $z \in N(y) \cap L$ increase by 1 index$[z]$. If index$[z]$=$\mid L_i \mid$ move z from L to L_{i+2} and put index$[z] = 0$];
If $L = \emptyset$ then **exit**; $\{L_0 \cup .. \cup L_i$ is a connected component of $\overline{G}\}$
$L_{i+1} := L$; $i := i + 1$; $L := L_{i+1}$
End_While

Complexity of Step 1. When constructing L we construct also an array $A[1..n]$ such that $A[i]$ contains a direct access in the list containing i during the execution of the above algorithm. Hence, we can find in constant time the neighbors of every vertex $y \in L_i$. It is easy to see that the complexity of the above algorithm is linear on the size of G.
Steps 2,3. Since $G[L_i]$ must be a cograph we check this by obtaining in linear time the corresponding cotree T by using the algorithm in [2]. Then we obtain the corresponding cotree as well as the connected components of $\overline{G[L_i]}$ by changing the 0-nodes of T into 1-nodes and its 1-nodes into 0-nodes.
Steps 4,5. We shall sort the vertices of L_i by decreasing inner degree, y will be the vertex whose inner degree will be $\mid L_{i-1} \mid -1$.
Steps 6. Once the vertices of L_i have been sorted by decreasing inner degree, using the array A we find first the non-neighborhood in L_{i-1} of each vertex x of L_i within $O(degree(x))$ complexity and then proceed in an analogous manner described on Steps 2 and 3 above.

It is clear now that we can apply the Algorithm 1 in $\overline{G}$ within linear time complexity on the size of G. It remains a last verification presented in [3] that consists to check if the obtained pruning sequence (S, σ) corresponds to an HD graph. Due to the space limitations of this extended abstract, we leave to the reader to verify that this can be done in linear time on the size of G.

2 Clique width of co-distance hereditary graphs

The well known notion of *clique-width* of a graph G denoted $cwd(G)$, is the minimum number of labels needed for constructing G using four graph operations: labeling by i a new vertex v(denoted $i(v)$), disjoint union of H and H' denoted $H \oplus H'$, $\eta_{i,j}(G)$, $i \neq j$, is the graph obtained by connecting all the vertices labeled i to all the vertices labeled j in G and $\rho_{i,j}(G)$ the graph obtained by renaming i into j in G. An expression obtained from the above four operations using k labels is called a *k-expression*. We denote by $G(t)$ a graph defined by the expression t. In [4], it is proved that every distance hereditary graph, has clique-width at most 3 and a $3-$expression defining it can be obtained in linear time. This expression is constructed as follows: from the pruning sequence (S, σ) associated with a DH graph G we construct a special tree $T(G)$, the *pruning tree*, whose vertices are the vertices of G and whose edges $\{x, y\}$ are labeled l, F or T if there exists s_i in σ such that s_i is (xlf),(xFy) or (xTy) respectively. Let α be a node of $T(G)$, T_α is the set of vertices of G of the sub-tree rooted at a and $G(T_\alpha)$ the subgraph of G induced by the vertices of T_α. Let u and v be two vertices of T_α, then u is a twin descendant of v if all the edges connected u to v are labeled with true or false. Let $\alpha_1, ..., \alpha_l$ be the sons of a ordered from left to right. In [4] it is proved that for every α_i the set of edges relying the vertices of $G(T_{\alpha_i})$ and $G(a \cup T_{\alpha_{i+1}} \cup ... \cup T_{\alpha_l})$ is empty whenever α_i is a false twin son of a and it is formed by all $\{u, v\}$ where

u is a twin descendant of α_i in T_{α_i} and v is either α or a twin descendant of α in $T_{\alpha_{i+1}} \cup ... \cup T_{\alpha_l}$. This result allowed to obtain a $3 - expression$ for a distance hereditary graph by labeling the twin descendants of any node β in T_β with 2 and by 1 all the other vertices of T_β. Using this labeling for the vertices of G and using the pruning tree of $\overline{G}$ we can obtain a $4 - expression$ for G in linear time. For this we shall calculate the expression e_i associated with $G(T_{\alpha_i} \cup T_{\alpha_{i+1}} \cup ... \cup T_{\alpha_l})$ by assuming that we know the $3 - expression\ t_{\alpha_i}$ associated with $G(T_{\alpha_i})$ and the $3 - expression\ e_{i+1}$ associated with $G(T_{\alpha_{i+1}} \cup ... \cup T_{\alpha_l})$. We then have:
1. If a_i is a leaf son of α then
$e_i = \rho_{4\to1}(\rho_{3\to1}(\eta_{1,4}(\eta_{1,3}(\eta_{2,3}(e_{i+1} \oplus (\rho_{2\to4}(\rho_{1\to3}(t_{\alpha_i}))))))))$
2. If a_i is a true twin son of α then $e_i = \rho_{4\to2}(\rho_{3\to1}(\eta_{1,4}(\eta_{1,3}(\eta_{2,3}(e_{i+1} \oplus (\rho_{2\to4}(\rho_{1\to3}(t_{\alpha_i}))))))))$
3. If a_i is a false twin son of α then $e_i = \rho_{4\to2}(\rho_{3\to1}(\eta_{1,4}(\eta_{1,3}(\eta_{2,4}(\eta_{2,3}(e_{i+1} \oplus (\rho_{2\to4}(\rho_{1\to3}(t_{\alpha_i}))))))))$.

It is now to see how we can obtain a $4 - expression$ for G in linear time on the size of G. It follows that many optimization problems have linear solution for co-DH graphs (see [4]).

References

[1] A. Brandstädt, V.B. Le and J. Spinrad, Graph classes: a survey, *SIAM Monographs on Disc. Math. and Applications*, (1999).

[2] D.G. Corneil, Y. Perl, L.K. Stewart, A linear recognition for cographs, *SIAM J. comput*. 14(4), (1985), 926-934.

[3] G. Damiand, M. Habib, C. PaulL, A simple pradigm for graph recognition : Application to cographs and distance hereditary graphs. *Theoretical Computer Sciences*, 263, (2001), 99-111.

[4] M.C. Golumbic, U. Rotics : On the clique-width of some perfect graph classes, *Int. J. Found. Comput. Sci*. 11(3), (2000), 423-443 .

[5] P.L. Hammer, F. Maffray, Completely separable graphs, *Discrete Appl. Math*. 27, (1990), 85-99.

Global r-alliances and total domination

Henning Fernau [a], Juan A. Rodríguez-Velázquez [b,1,*], José M. Sigarreta [c]

[a] *FB 4-Abteilung Informatik. Universität Trier. Trier, Germany*

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

[c] *Department of Mathematics, Popular Autonomous University of the State of Puebla. 21 sur 1103, Colonia Santiago CP 72410, Puebla, Pue., México*

Key words: Computational complexity, domination in graphs, alliances in graphs.

1. Introduction

We show NP-completeness of several graph problems and exhibit some of their combinatorial properties. More specifically, we relate the concepts of alliances in graphs and of domination. We discuss a natural parameter that expresses the strength of an alliance.

General notions. $\Gamma = (V, E)$ denotes a simple graph of order $n = |V|$ and size $m = |E|$. The degree of a vertex $v \in V$ will be denoted by $\delta(v)$. For a non-empty subset $S \subset V$ and a vertex $v \in V$, we denote by $N_S(v)$ the set of neighbors v has in S. We denote the degree of v in S by $\delta_S(v) = |N_S(v)|$. Δ denotes the maximum degree of a graph. The boundary of a set $S \subseteq V$ is defined as $\partial S := \cup_{v \in S} N_{\bar{S}}(v)$.

Total domination. A set $S \subset V$ is a *dominating set* if $\delta_S(v) \geq 1, \forall v \in \bar{S} = V \backslash S$. The *domination number* $\gamma(\Gamma)$ is the minimum cardinality of a dominating set. The concept of total domination was introduced by Cockayne, Dawes and Hedetniemi in [2]: a set $S \subset V$ is a *total dominating set* if $\delta_S(v) \geq 1$, $\forall v \in V$. The *total domination number* $\gamma_t(\Gamma)$ is the minimum cardinality of a total dominating set. The concept of total domination can be extended to multiple domination. That is, a set $S \subset V$ is a *total k-dominating set* if $\delta_S(v) \geq k$, $\forall v \in V$. So, the *total k-domination number* $\gamma_{kt}(\Gamma)$ is the minimum cardinality of a total k-dominating set. Notice that the concept of total 2-domination is different from the concept of

* Corresponding author.

[1] This work was partly supported by the Spanish Ministry of Education through projects TSI2007-65406-C03-01 “E-AEGIS" and CONSOLIDER CSD2007-00004 “ARES", and by the Rovira i Virgili University through project 2006AIRE-09.

double domination introduced by Harary and Haynes in [6]. A set $S \subset V$ is a *double dominating* set if $\delta_S(v) \geq 2$, $\forall v \in \bar{S}$ and $\delta_S(v) \geq 1$, $\forall v \in S$. A *fortiori* every double dominating set is a total 1-dominating set.

Alliances. Alliances in graphs were introduced as a graph-theoretic model of friendship and hostility relationships that might apply to military or similar scenarios. Depending on the character of such an alliance, defensive and offensive groupings were studied (and combinations thereof). We further generalize these notions by considering the *strength* of alliances [9].

A nonempty set of vertices $S \subseteq V$ is called a *defensive r-alliance* in Γ if for every $v \in S$, $\delta_S(v) \geq \delta_{\bar{S}}(v) + r$, where r is the *strength* of the defensive r-alliance, $-\Delta < r \leq \Delta$. A defensive (-1)-alliance is a "*defensive alliance*", and a defensive 0-alliance is a "*strong defensive alliance*" (as defined in [8]). A particular case, called "*global defensive alliance*", was studied in [7]. A defensive r-alliance S is called *global* if S is a dominating set. The *global defensive r-alliance number* $\gamma_r^d(\Gamma)$ is the minimum cardinality of any global defensive r-alliance in Γ.

A non-empty set of vertices $S \subseteq V$ is called an *offensive r-alliance* in Γ if and only if for every $v \in \partial S$, $\delta_S(v) \geq \delta_{\bar{S}}(v) + r$, where $-\Delta + 2 < r \leq \Delta$ denotes the *strength* of the alliance. In particular, an offensive 1-alliance is an "*offensive alliance*", and an offensive 2-alliance is a "*strong offensive alliance*" (as defined in [8]). An offensive r-alliance S is called *global* if S is a dominating set. The *global offensive r-alliance number*, denoted by $\gamma_r^o(\Gamma)$, is defined as the minimum cardinality of a global offensive r-alliance in Γ.

An alliance is called *dual* (sometimes also called *powerful*) if it is both defensive and offensive [8]. Hence, a *global dual alliance* is a global dual (-1)-alliance, i.e., a dominating set that is both a (-1)-defensive alliance and a 1-offensive alliance, and a *global strong dual alliance* is a global dual 0-alliance, i.e., a (0)-defensive global alliance and a 2-offensive alliance. In general, a set $S \subset V$ is a dual r-alliance in Γ if S is both a global defensive r-alliance and an $(r+2)$-offensive alliance in Γ. So, for dual r-alliances, $-\Delta < r \leq \Delta - 2$. The *global dual r-alliance number*, denoted by $\gamma_r^*(\Gamma)$, is defined as the minimum cardinality of a global dual r-alliance in Γ.

2. Global r-alliances

Cami et al. [1] showed NP-completeness for $r = -1$. We were able to modify their construction to show NP-completeness for any fixed r.

Theorem 1 *For all fixed r, the following problem is is NP-complete: Given a graph Γ and a bound ℓ; determine if $\gamma_r^d(\Gamma) \leq \ell$.*

Theorem 2 *For any graph Γ,* $\dfrac{\sqrt{4n+k^2}+k}{2} \leq \gamma_k^d(\Gamma) \leq n - \left\lceil \dfrac{\delta_n - k}{2} \right\rceil$.

Theorem 3 *For any graph Γ,* $\gamma_k^d(\Gamma) \geq \left\lceil \dfrac{n}{\left\lfloor \frac{\delta_1 - k}{2} \right\rfloor + 1} \right\rceil$.

Corollary 1 *For any graph Γ of size m and maximum degrees $\delta_1 \geq \delta_2$, $\gamma_k^d(L(\Gamma)) \geq \left\lceil \frac{m}{\lfloor \frac{\delta_1+\delta_2-2-k}{2} \rfloor + 1} \right\rceil$, where $L(\Gamma)$ denotes the line graph of Γ.*

Theorem 4 *For all fixed r, the following problem is NP-complete: Given a graph Γ and a bound ℓ; determine if $\gamma_r^o(\Gamma) \leq \ell$.*

A set $S \subset V$ is a k-dominating set if for every $v \in \bar{S}$, $\delta_S(v) \geq k$. The *k-domination number* of Γ, $\gamma_k(\Gamma)$, is the minimum cardinality of a k-dominating set in Γ.

Theorem 5 *For any simple graph Γ of order n, minimum degree δ, and Laplacian spectral radius μ_*, $\left\lceil \frac{n}{\mu_*} \left\lceil \frac{\delta+r}{2} \right\rceil \right\rceil \leq \gamma_r^o(\Gamma) \leq \left\lfloor \frac{\gamma_r(\Gamma)+n}{2} \right\rfloor$.*

Theorem 6 *For all fixed r, the following problem is is NP-complete: Given a graph Γ and a bound ℓ; determine if $\gamma_r^*(\Gamma) \leq \ell$.*

Theorem 7 *For any graph Γ of order n, size m and minimum degree δ, $\left\lceil \frac{\sqrt{8m+4n(r+2)+(r+1)^2}+r+1}{4} \right\rceil \leq \gamma_r^*(\Gamma) \leq n - \left\lceil \frac{\delta-r}{2} \right\rceil$.*

3. Total k-domination

We consider the following decidability problem total k-domination (k-TD) for each fixed integer $k \geq 1$: Given $\Gamma = (V, E)$ and an integer parameter ℓ, is there a vertex set D with $|D| \leq \ell$ such that $\delta_D(v) \geq k$ for all $v \in V$? The smallest ℓ such that Γ together with ℓ forms a YES-instance of k-TD is denoted $\gamma_{kt}(\Gamma)$.

Theorem 8 *$\forall k \geq 1$: k-TD is NP-complete.*

Theorem 9 *Every total k-dominating set is a global defensive (offensive) r-alliance, where $-\Delta < r \leq 2k - \Delta$. Moreover, every global dual r-alliance, $r \geq 1$, is a total r-dominating set.*

Corollary 2 *Each total k-dominating set is a global dual r-alliance, where $-\Delta < r \leq 2(k-1) - \Delta$.*

Corollary 3

- *For $-\Delta < r \leq 2k - \Delta$, $\gamma_{kt}(\Gamma) \geq \gamma_r^d(\Gamma)$ and $\gamma_{kt}(\Gamma) \geq \gamma_r^o(\Gamma)$.*
- *For $-\Delta < r \leq 2(k-1) - \Delta$, $\gamma_{kt}(\Gamma) \geq \gamma_r^*(\Gamma)$.*
- *For $k \geq 1$, $\gamma_k^*(\Gamma) \geq \gamma_{kt}(\Gamma)$.*

By Corollary 3 we have that lower bounds for $\gamma_r^d(\Gamma)$, $\gamma_r^o(\Gamma)$ and $\gamma_r^*(\Gamma)$ lead to lower bounds for $\gamma_{kt}(\Gamma)$. Moreover, upper bounds for $\gamma_{kt}(\Gamma)$ lead to upper bounds for $\gamma_r^d(\Gamma)$, $\gamma_r^o(\Gamma)$ and $\gamma_r^*(\Gamma)$.

Concluding Remarks. Let us finally mention that most complexity results presented in this paper for various types of global alliances can be shown in the non-global case, as well. This generalizes earlier results from [4, 5]. Interestingly, we could show (as in [5]) fixed parameter tractability for all mentioned problems, while (as noted above) the obviously related total domination problem is W[2]-hard. (For notions concerning parameterized complexity, we refer to [3].)

References

[1] A. Cami, H. Balakrishnan, N. Deo, and R. Dutton. On the complexity of finding optimal global alliances. *Journal of Combinatorial Mathematics and Combinatorial Computing*, **58** (2006).

[2] E. J. Cockayne, R. Dawes and S. T. Hedetniemi Total domination in graphs, *Networks* **10** (1980), 211–215.

[3] R. G. Downey and M. R. Fellows, *Parameterized Complexity*, Springer, 1999.

[4] O. Favaron, G. Fricke, W. Goddard, S. M. Hedetniemi, S. T. Hedetniemi, P. Kristiansen, R. C. Laskar and D. R. Skaggs. Offensive alliances in graphs. *Discuss. Math. Graph Theory* **24** (2)(2004), 263–275.

[5] H. Fernau and D. Raible, Alliances in graphs: a complexity-theoretic study. Software Seminar SOFSEM 2007, Student Research Forum; SOFSEM Proc. Vol. II. Institute of Computer Science ASCR, Prague, 2007, pp. 61–70.

[6] F. Harary and T. W. Haynes, Double domination in graphs, *Ars Combinatoria* **55** (2000), 201–213.

[7] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning, Global defensive alliances in graphs, *Electron. J. Combin.* **10** (2003), Research Paper 47.

[8] P. Kristiansen, S. M. Hedetniemi and S. T. Hedetniemi, Alliances in graphs. *J. Combin. Math. Combin. Comput.* **48** (2004), 157–177.

[9] K. H. Shafique and R. D. Dutton, Maximum alliance-free and minimum alliance-cover sets, *Congr. Numer.* **162** (2003) 139-146.

Graph optimization (II)

Room B, Session 4

Wednesday 14, 9:00-10:30

Exact Algorithms for the Vertex Separator Problem in Graphs [1]

Victor F. Cavalcante [a] Cid C. de Souza [a,*]

[a] *University of Campinas, Institute of Computing, Campinas SP, Brazil*

Key words: Lagrangian relaxation, cutting planes, Integer Programming, vertex separator, relax-and-cut algorithms.

Introduction. A *vertex separator* in an undirected graph is a subset of the vertices, whose removal disconnects the graph. The Vertex Separator Problem (VSP) was recently studied in [1–3] and can be stated as follows: given a connected undirected graph $G = (V, E)$, with $|V| = n$, an integer $1 \leq b \leq n$ and a cost c_i associated with each vertex $i \in V$, find a partition of V into disjoint sets A, B, C, with A and B nonempty, such that *(i)* E contains no edge (i, j) with $i \in A$, $j \in B$, *(ii)* $\max\{|A|, |B|\} \leq b$, *(iii)* $\sum_{j \in C} c_j$ is minimized.

The VSP is $\mathcal{NP}$-hard and has widespread applicability in network connectivity[1, 3]. It was studied by Balas and de Souza in [1] where a polyhedral investigation is performed. Also, extensive computational results obtained with a branch-and-cut developed by the same authors are reported in [2].

In this work we investigate the usage of Lagrangian techniques in the development of more efficient methods to solve VSP instances to optimality. Recent papers [4–9] report successful applications of the so-called relax-and-cut (R&C) algorithms for discrete optimization problems. These algorithms offer an alternative to strengthen the dual bounds provided by classical Lagrangian relaxations. This is done through the separation and later dualization of valid inequalities within the Lagrangian framework, similar to what happens to cutting planes in Integer Programming (IP). Thus, we decided to focus on the development of R&C algorithms for the VSP.

Results obtained with a very preliminary version of our R&C framework were presented in [3]. Here we discuss algorithmic and implementation issues that allowed

* Corresponding author.

[1] First author supported by a scholarship from CAPES (Brazilian Ministry of Education). Second and corresponding author supported by grants 301732/2007-8, 478470/2006-1, 472504/2007-0 and 473726/2007-6 from *Conselho Nacional de Desenvolvimento Científico e Tecnológico* and grant 03/09925-5 from *Fundação de Amparo à Pesquisa do Estado de São Paulo*.

us to improve on these early results. Below we briefly describe the basic steps and implementation aspects that are relevant to the design of relax-and-cut algorithms.

Relax-and-cut algorithm basics. Suppose we have an IP formulation for certain discrete optimization problem. Assume that we have two sets of constraints, one of which, makes the problem hard, i.e., if we get rid of them, the resulting problem can be solved efficiently. In the classical Lagrangian relaxation scheme, we dualize the set of hard constraints, by penalizing them in the objective function. Given the vector of penalties, we define the Lagrangian Subproblem (LS) with respect to these values, whose optimum is a dual bound for the optimal value of the original problem. The Lagrangian Dual problem seeks the vector which leads to the best dual bound and can be solved, for example, by the subgradient method (SM). Now, at each iteration of the SM, one has to compute an optimal solution of LS, say x^*. It turns out that this solution may not satisfy some of the dualized constraints and this provokes the recalculation of the associated penalties. However, if we know a family $\mathcal{F}$ of strong valid inequalities for the convex hull of the feasible points of the original IP model, we may dualize further inequalities. Suppose that a separation routine for $\mathcal{F}$ is at hand. Then, we can solve the separation problem for $\mathcal{F}$ and x^* and, if a violated inequality is found, we dualize it. This is the chief idea of a R&C algorithm, likewise polyhedral cutting-plane generation is applied in IP. Nevertheless, it is worth noting that separation problems arising in R&C algorithms may be easier than their polyhedral cutting-plane algorithm counterparts, since LS normally has integral valued solutions (cf. [9]).

Implementation strategies. Two strategies to implement R&C algorithms are discussed in the literature. They differ, basically, on the moment at which the new inequalities are dualized. In a Delayed Relax-and-Cut (DR&C), several executions of SM are made and the cuts found during one such execution are added only at the beginning of the next one. In a Non Delayed Relax-and-Cut (NDR&C), typically a single SM run is done and cuts are dualized along the iterations as they are found. See [4, 5, 7–9] for details. Comparison carried out in [8] suggested that NDR&C performed better than DR&C.

An IP formulation for the VSP. We now describe the IP formulation for the VSP presented in [1], on which our Lagrangian relaxation is based. For every vertex $i \in V$, two binary variables are defined: $u_{i1} = 1$ if and only if $i \in A$ and $u_{i2} = 1$ if and only if $i \in B$. For $S \subseteq V$ and $k \in \{1, 2\}$, let $u_k(S)$ denote $\sum(u_{ik} : i \in S)$, $u(S) = u_1(S) + u_2(S)$. In the IP model for the VSP shown below condition (1) forces every vertex to belong to at most one shore. Inequalities (2) prohibits the extremities of an edge to be on distinct shores. Inequalities (3) limit the size of the shores and, at the same time. As observed in [2], if the u_{i1} variables are integer for all $i \in V$, the integrality of the u_2 variables can be dropped from the formulation. Though this observation is not taken into account by our Lagrangian relaxation, it

is relevant for IP solvers.

$$\max \sum_{i \in V} c_i(u_{i1} + u_{i2})$$

$$u_{i1} + u_{i2} \leq 1, \qquad \forall\, i \in V \tag{1}$$

$$u_{i1} + u_{j2} \leq 1,\ u_{j1} + u_{i2} \leq 1, \qquad \forall\, (i,j) \in E \tag{2}$$

$$1 \leq u_k(V) \leq b, \qquad k = 1, 2 \tag{3}$$

$$u_{i2} \geq 0,\ u_{i1} \in \{0,1\}, \qquad \forall\, i \in V. \tag{4}$$

Different Lagrangian relaxations can be devised from the IP formulation above. In this work we evaluate some of them and choose a simple one in which the constraint sets (1) and (2) are dualized. As seen before, R&C algorithms are based on families of valid inequalities whose elements are dynamically dualized. In the VSP case, we use two families of cuts described in [1]. The first class of cutting planes is related to minimal connected dominators (`CD` inequalities). The other class is associated with minimal dominators and a lifting procedure (`LD` inequalities).

The relax-and-cut framework. The algorithm we implemented is structured in three modules. The first one is the R&C algorithm. The output of this module is the set of `CD` and/or `LD` inequalities separated during the execution of the algorithm and the best primal and dual bounds computed. The second one executes a classical Lagrangian procedure based on the model (1)-(4) appended with the cuts returned from the first module. Finally, the third module corresponds to the B&C algorithm in [2] modified to improve the IP model according to the bounds computed by R&C and to separate the cuts returned by the latter in a table lookup fashion. If one is willing to use R&C as a stand alone approach for VSP, the overall algorithm is aborted after the execution of the second module. On condition that a hybrid exact algorithm combining R&C and B&C is to be ran, the second module is disabled, and the output of first module is directly used by the third one.

Computational results. Computational tests were conducted on instances of public domain. The main conclusions of these experiments can be summarized as follows: (A) there is no clear dominance between delayed and non delayed versions of R&C; (B) for dense instances the R&C algorithms compute optimal solutions much faster than the B&C algorithms from [2], outperforming the latter by far in almost all the cases; (C) for sparse graphs our framework is competitive with the B&C algorithms from Balas and de Souza, but we still find few instances where the standard branch-and-bound outperforms both algorithms, confirming the reports in [2]; (D) R&C algorithms alone can rarely solve instances of VSP to optimality. On the other hand, hybrid approaches that use R&C algorithms as a preprocessing phase for a B&C algorithm are well suited to compute optimal VSP solutions; (E) Finally, on the primal side, the R&C algorithm proved to be a very effective heuristic, producing very high quality solutions, often optimal ones, in minute computation times;

Conclusions. The combination of Lagrangian and cutting plane algorithms as proposed by relax-and-cut and its hybridization with a branch-and-cut algorithm for VSP is a promising approach to tackle the problem. The results obtained with our computational experiments turn the framework proposed, to our knowledge, the best exact algorithm available for the VSP to date. Moreover, we believe this technique can also be successfully applied to other combinatorial optimization problems for which a polyhedral study has been conducted. However, from our experience, this might require intensive experimentation and some cleverness on how to combine R&C and B&C algorithms.

References

[1] Balas, E., Souza, C.: The vertex separator problem: a polyhedral investigation. Mathematical Programming **103** (2005) 583–608

[2] Souza, C., Balas, E.: The vertex separator problem: algorithms and computations. Mathematical Programming **103** (2005) 609–631

[3] Cavalcante, V.F., de Souza, C.C.: Lagrangian relaxation and cutting planes for the vertex separator problem. In Chen, B., Paterson, M., Zhang, G., eds.: ESCAPE. Lecture Notes in Computer Science., Springer (2007), v. 4614, 1–482

[4] Calheiros, F., Lucena, A., de Souza, C.: Optimal rectangular partitions. Networks **41** (2003) 51–67

[5] Guignard, M.: Lagrangean relaxation. In Lòpez-Cerdá, M., García-Jurado, I., eds.: Top. Volume 11., Madrid, Spain, Sociedad de Estadística e Investigación Operativa (2003) 151–228

[6] Kliewer, G., Timajev, L.: Relax-and-cut for capacitated network design. In: Proceedings of 13^{th} Annual European Symposium on Algorithms (ESA). Volume 3669., Mallorca, Spain (October 2005) 47–58

[7] Lucena, A.: Steiner problem in graphs: Lagrangean relaxation and cutting-planes. In: COAL Bulletin. Volume 21., Math. Prog. Society (1992)

[8] Lucena, A.: Non delayed relax-and-cut algorithms. Annals of Operations Research **140**(1) (2005) 375–410

[9] Martinhon, C., Lucena, A., Maculan, N.: Stronger k-tree relaxations for the vehicle routing problem. European Journal on Op. Research **158** (2004) 56–71

Inverse Tension Problems

Çiğdem Güler [1,*]

Department of Mathematics, University of Kaiserslautern
67653 Kaiserslautern, Germany

For optimization problems with estimated problem parameters one often knows a priori an optimal solution based on observations or experiments, but is interested in finding a set of parameters, such that the known solution is optimum (a) and the deviation from the initial estimates is minimized (b). The problem of recalculating the parameters satisfying (a) and (b) is known as *inverse optimization problem*.

Among several inverse optimization problems the inverse network flows have been intensely investigated [2, 3, 7, 15, 16]. Ahuja and Orlin [2] derive LP formulations for several inverse network flow problems. In another paper [3] they analyze the combinatorial aspects of inverse minimum cost flow problem under unit weight L_1 and L_∞ norms. Yang et al. [16] study inverse maximum flow and minimum cut problems. A thorough survey study on this topic has been done by Heuberger [10] analyzing different types of inverse and reverse problems that have been considered in the literature. As opposed to network flows, their duals *tension problems* [1] and the corresponding inverse versions have vastly been neglected. Our aim in this study is to fill this blank of the literature and show that the duality relation between tensions and flows is valid for their respective inverse problems, as well. Moreover, this study enlightens the connnection between tension problems and cut problems.

Let $G = (N, A)$ be connected digraph with node set N containing n nodes and arc set A containing m arcs, and a_{ij} represent an arc with tail node i and head node j. A *tension* is a function from A to $\mathbb{R}$ which satisfies Kirchhoff's law for voltages [13]. In other words, a vector $\theta \in \mathbb{R}^A$ is a *tension* on graph G with *potential* $\pi \in \mathbb{R}^N$ such that $\forall (i,j) \in A \quad \theta_{ij} = \pi_j - \pi_i$.

Minimum cost tension problem (MCT) is finding a tension θ satisfying lower ($t_{ij} \in \mathbb{R} \cup \{-\infty\}$) and upper ($T_{ij} \in \mathbb{R} \cup \{+\infty\}$) bounds on each arc such that $\sum_{a_{ij} \in A} c_{ij}\theta_{ij}$ is minimum. In *maximum tension problem (MaxT)*, the graph G contains 2 special nodes, s and t, and an arc $a_{st} \in A$ between these two nodes with bounds $(t_{st}, T_{st}) = (-\infty, \infty)$. The maximum tension problem is finding the maximum tension on arc $a_{st} \in A$ such that the tensions on all arcs satisfy the upper and lower bounds.

---

* Corresponding author.

[1] The research has been partially supported by the Deutsche Forschungsgemeinschaft (DFG), Grant HA 1737/7 "Algorithmik großer und komplexer Netzwerke" and by the Rheinland-Pfalz cluster of excellence "Dependable adaptive systems and mathematical modeling".

Given a feasible tension $\hat{\theta}$ to an instance of a MCT (MaxT), the **inverse minimum cost tension problem** (IMCT) (**inverse maximum tension problem** (IMaxT)) is perturbing the cost vector from c to $\hat{c}$ (bound vectors from T to $\hat{T}$ and/or from t to $\hat{t}$) in a way that $\hat{\theta}$ will become the optimum tension with the perturbed cost vector (bound vectors) while the perturbation $\|c - \hat{c}\|$ ($\|T - \hat{T}\| + \|t - \hat{t}\|$) is minimized according to some norm. We consider rectilinear (L_1) and Chebyshev (L_∞) norms for the inverse minimum cost tension problem, and L_1 norm for the inverse maximum tension problem.

Inverse Minimum Cost Tension Problem Under L_1-Norm

Suppose w_{ij}s are the weights associated with the cost changes on arcs. Then, objective function under L_1 norm is to minimize $\sum_{a_{ij} \in A} w_{ij}|c_{ij} - \hat{c}_{ij}|$. The residual cuts ω_1 and ω_2 are called ***arc-disjoint*** if $\omega_1^+ \cap \omega_2^+ = \emptyset$ and $\omega_1^- \cap \omega_2^- = \emptyset$. Using the arc-disjoint residual cuts, we can show that the inverse minimum cost tension problem under rectilinear norm reduces to solving a *minimum cost tension problem*.

Theorem 1 *Let $Cost(\Omega^*)$ be the minimum cost of a collection of arc-disjoint residual cuts in G. Then, $-Cost(\Omega^*)$ is the optimal objective function value for the inverse minimum cost tension problem under unit weight rectilinear norm.*

If we define the sets of arcs $K := \{a_{ij} \in A : t_{ij} < \hat{\theta}_{ij} < T_{ij}\}$, $L := \{a_{ij} \in A : \hat{\theta}_{ij} = t_{ij}\}$, and $U := \{a_{ij} \in A : \hat{\theta}_{ij} = T_{ij}\}$, then LP corresponding to the inverse minimum cost tension problem under unit weight L_1 norm is:

$$\text{Minimize} \quad \sum_{a_{ij} \in A} c_{ij}(\pi_j - \pi_i) \tag{1}$$

subject to

$$\begin{aligned} -1 \leq \pi_j - \pi_i \leq 1 \quad & \text{for } a_{ij} \in K \\ 0 \leq \pi_j - \pi_i \leq 1 \quad & \text{for } a_{ij} \in L \\ -1 \leq \pi_j - \pi_i \leq 0 \quad & \text{for } a_{ij} \in U \end{aligned}$$

For the nonunit weights case, the LP remains to be a minimum cost tension problem but with the corresponding bounds w_{ij} and $-w_{ij}$ instead of 1 and -1 for the tension.

Inverse Minimum Cost Tension Problem Under L_∞-Norm

Inverse minimum cost tension problem under L_∞ norm has an objective of minimizing $\max_{a_{ij} \in A} w_{ij}|c_{ij} - \hat{c}_{ij}|$. The problem under unit weights reduces to solving a *minimum mean residual cut problem*. In order to achieve this, we exploit the optimality conditions [9] for the tensions and the ϵ-optimality definition by Hadjiat and Maurras [8].

Theorem 2 *Let μ^* denote the mean cost of a minimum mean residual cut in G w.r.t. $\hat{\theta}$. Then, the optimal objective function value for the inverse minimum cost tension problem under l_∞ norm is $max(0, -\mu^*)$.*

Hadjias and Maurras [8] provide a Newton type algorithm to solve the minimum mean residual cut problem. Using their algorithm we can find an optimum solution for the inverse problem in strongly polynomial time. McCormick and Ervolina [12] study max mean cuts and mention that a direct method of calculating max mean cuts as Karp [11] does for minimum mean cycles has not yet been found. Here, we present a direct method to identify the minimum mean cost cut and derive an LP formulation. To the best of our knowledge, this is the first non-iterative method presented in the literature.

Suppose $\omega(i)$ denote a cut with $(S, \bar{S}) = (\{i\}, A\backslash\{i\})$ and $Cost(\omega(i))$ is its corresponding cost. We define a new graph $G' = (N, A')$ with $A' = A \cup \bar{A} = \{a_{ji} : a_{ij} \in A\}$ and supplies/demands of $-Cost(\omega(i))$ on each node $i \in N$. Our goal is to find nonnegative flows on arcs, $\varphi_{ij} \geq 0$ for $a_{ij} \in A'$, such that the supplies and demands are satisfied and the maximum of $\varphi_{ij} + \varphi_{ji}$ for all $a_{ij} \in A$ and $a_{ji} \in \bar{A}$ is minimum. We call this problem **equal network flow problem**.

Theorem 3 *If z^* is the optimum objective fuction value of the equal network flow problem on $G' = (N, A')$, then the cost of the minimum mean cut is $-z^*$. Moreover, the dual LP of the equal network flow problem is a **minimum cost tension problem** on graph $G = (N, A)$.*

Corollary 1 *The minimum mean **residual** cut problem on graph $G = (N, A)$ can be formulated as a minimum cost tension problem on the same graph.*

Inverse Maximum Tension Problem (IMaxT) under L_1-Norm

Given a weight vector w for changing the bounds of the arcs, the inverse maximum tension problem under L_1 norm is minimizing $\sum_{a_{ij} \in A} w_{ij}(|\hat{T}_{ij} - T_{ij}| + |\hat{t}_{ij} - t_{ij}|)$ such that $\hat{\theta}_{st}$ is the maximum tension for the maximum tension problem on $G(\hat{t}, \hat{T})$.

For the maximum tension it is known that there exists a minimum path, which has a length equal to the maximum tension [14]. Using this fact and the properties of minimum path, we can prove the following result.

Theorem 4 *Suppose P^* is the minimum path corresponding to the maximum tension problem in $G(t, T)$. P^{*+} and P^{*-} denote the forward and backward arcs of the path, respectively. The optimum solution of the inverse maximum tension problem w.r.t. unit weight L_1-norm is*

$$T^*_{ij} = \begin{cases} \hat{\theta}_{ij} & \text{if } a_{ij} \in P^{*+} \\ T_{ij} & \text{otherwise} \end{cases} \qquad t^*_{ij} = \begin{cases} \hat{\theta}_{ij} & \text{if } a_{ij} \in P^{*-} \\ t_{ij} & \text{otherwise} \end{cases}$$

Hence, solving the inverse problem is equivalent to solving a maximum tension problem on $G(t, T)$.

Conclusion and Future Work

For the inverse minimum cost and maximum value tension problems under rectilinear and Chebyshev norms we show that the duality relationship between network flows and tensions is also valid. Hence, by a generalization of this approach from

network flows and tensions to flows in regular matroids [4], we can get some insights into dealing with inverse linear programs with totally unimodular matrices. Moreover, it seems that inverse tension problems may have potential for practical applications, especially in scheduling problems. These are currently explored, as well.

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, New Jersey, 1993.

[2] R.K Ahuja and J.B. Orlin. Inverse optimization. *Operations Research*, 49:771–783, 2001.

[3] R.K Ahuja and J.B. Orlin. Combinatorial algorithms of inverse network flow problems. *Networks*, 40:181–187, 2002.

[4] R.E. Burkard and H.W. Hamacher. Minimal Cost Flows in Regular Matroids *Mathematical Programming Studies*, 14:32–47, 1981.

[5] D. Burton and P. Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53:45–61, 1992.

[6] D. Burton and P. Toint. On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming*, 63:1–22, 1994.

[7] C. Güler and H.W. Hamacher. Capacity Inverse Minimum Cost Flow Problem. *Under Revision*.

[8] M. Hadjiat and J.F. Maurras. A strongly polynomial algorithm for the minimum cost tension problem. *Discrete Mathematics*, 165:377–394, 1997.

[9] Horst W. Hamacher. Min cost tensions. *Journal of Information and Optimization Sciences*, 6:285–304, 1985.

[10] Clemens Heuberger. Inverse optimization: A survey on problems, methods, and results. *Journal of Combinatorial Optimization*, 8:329–361, 2004.

[11] R.M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23:309–311, 1981.

[12] S.T. McCormick and T.R. Ervolina. Computing Maximum Mean Cuts. *Discrete Applied Mathematics*, 52:53–70, 1994.

[13] Jean-Marie Pla. An out-of-kilter algorithm for solving minimum cost potential problems. *Mathematical Programming*, 1:275–290, 1971.

[14] R.T. Rockafellar. *Network Flows and Monotropic Optimization*. John Wiley and Sons, New York, 1984.

[15] J. Zhang and Z. Liu. Calculating some inverse linear programming problems. *Journal of Computational and Applied Mathematics*, 72:261–273, 1996.

[16] C. Yang, J. Zhang, and Z. Ma. Inverse maximum flow and minimum cut problems. *Optimization*, 40:147–170, 1997.

On Minimum Reload Cost Paths, Tours and Flows

Edoardo Amaldi [a,*] Giulia Galbiati [b] Francesco Maffioli [a]

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

[b] *Dipartimento di Informatica e Sistemistica*
Università degli Studi di Pavia
Via Ferrata 1, 27100 Pavia, Italy

Key words: network optimization, extended reload cost model, computational complexity.

1. Introduction

We consider several basic network optimization problems under a generalization of the reload cost model introduced in [5]. The concept of reload cost, that is of a cost incurred when two consecutive arcs along a path are of different types, naturally arises in a variety of application contexts. For instance, in transportation it allows to account for the relevant cost of unloading and loading of freight from one carrier to another. Other interesting application areas include telecommunication networks and energy distribution.

In this paper we consider a general model including reload costs as well as regular arc costs. We are given a directed graph $G = (V, A)$ with a non-negative cost $w(a)$ for each arc $a \in A$. Moreover, as in [5], each arc a is assigned a color $l(a)$ out of a finite set L of colors and a non-negative integer *reload cost matrix* $R = \{r_{ll'}\}_{l,l' \in L}$ is given, where entry $r_{ll'}$ is the cost of going from an arc of color l to an arc of color l' ($r_{ll} = 0$ for all l in L). If P is any path in G consisting of k consecutive arcs $a_1, ..., a_k$, respectively of colors $l_1, ..., l_k$, the *reload cost* of P is $r(P) = \sum_{j=1}^{k-1} r_{l_j l_{j+1}}$, that is the sum of the reload costs at its internal nodes, and the *arc cost* of P is $w(P) = \sum_{j=1}^{k} w(a_j)$. Then we define the overall *transportation cost* as $c(P) = r(P) + w(P)$. A similar model involving undirected graphs can also be considered, as pointed out below.

In [5] and [1] complexity results are presented for the problem of finding a spanning tree of minimum reload cost diameter (there are no arc costs). In [2] Gamvros et al. consider the minimum reload cost spanning tree problem. Despite the natural

* Corresponding author.

applicability of the reload cost concept, these are the only references we are aware of.

In this work we investigate optimum paths, tours, and flows under the above general reload cost model. Indeed we believe that considering these classical problems under this model has a clear practical relevance. We focus here on the computational complexity of the problems under consideration.

2. Paths

In this section we consider four problems. *Problem* P1: find a minimum transportation cost path between two given nodes s and t of G. *Problem* P2: find a set of paths from a given node s to the other nodes of G minimizing the sum of their transportation costs. *Problem* P3: find a minimum transportation cost path-tree from s to the other nodes of G. *Problem* P4: find a path-tree from s to the other nodes of G minimizing the maximum among the transportation costs of its paths.

Problem P1 is easily seen to be polynomially solvable. Apply to all nodes of G different from s and t the following splitting procedure: each node v, say having n incoming arcs and m outgoing arcs, is replaced with a complete bipartite graph, oriented from the left to the right shore, having n vertices on the left shore, so that each arc incoming to v arrives to one and only one of these n vertices, and having m vertices on the right shore so that each arc outgoing from v comes out of one and only one of these m vertices. While the original arcs of G maintain their costs, each arc of the complete bipartite graphs, say from node x to node y, is assigned an arc cost equal to the reload cost due to the colors of the arc of G entering in x and of the arc of G outgoing from y. Let H be the resulting graph. It is straightforward to see that a minimum arc cost $s-t$ path in H corresponds to a minimum transportation cost $s-t$ path in G (possibly visiting a node more than once).

Some additional observations are in order. If G has no arc costs, then the above procedure still works yielding a minimum reload cost $s-t$ path in G. When G is an undirected graph, the splitting procedure has to be modified as follows. Each node v must be substituted by a clique containing a number of nodes equal to the degree of v in G, and each edge of G that is incident to v is incident to exactly one node of the clique. Each edge of the clique, say from node x to node y, receives a arc cost equal to the reload cost of going from the color of the edge of G incident to x to that of the edge of G incident to y. Notice that the results for problem P1 can also be obtained using the line-graph of G, instead of resorting to the splitting procedure.

Problem P2 is also polynomially solvable. Apply to G the same splitting procedure of Problem P1 and let H be the resulting graph. Compute a minimum arc cost path-tree T in H with origin s using, say, Dijkstra's algorithm. We show now how the paths in T from s to all vertices of H allow to identify a set of paths from s to all vertices of G, such that the sum of their transportation costs is minimum. For

each node v of G, say having n incoming arcs and m outgoing arcs, we select in the left shore of the bipartite graph replacing v in H, the node closest to s in H; then we use the path in T from s to this node to identify, as done in problem P1, a path in G from s to v. Obviously the resulting set of paths from s to all nodes of G, being a set of minimum transportation cost paths from s to the nodes of G, solves P2. However these paths do not usually form a tree of G. In fact we can show that Problem P3 is NP-hard, via a reduction from MIN SET COVER.

Problem P4 is also NP-hard, even if restricted to graphs having maximum degree 4. Indeed in [1] a polynomial-time reduction from problem 3-SAT-3 to a problem called MIN-DIAM is described. The same reduction, with edges appropriately directed, can be seen as a reduction from 3-SAT-3 to P4, that yields the desired result. Note that the min-max case of problem P2 remains instead solvable in polynomial time, using the same procedure that solves P2.

3. Tours

In this section we summarize some results concerning optimum tour problems. The problem of traversing all edges of an undirected graph G with a tour of minimum arc cost so that each edge is used at least once is the famous Chinese Postman Problem (CPP), which is solvable in polynomial time. If we look for a similar tour of minimum transportation cost, this problem becomes NP-hard. Consider indeed the subproblem in which G is already Eulerian, the arc costs are all zero, and we look for an Eulerian tour of minimum total reload cost. We can show that this problem is NP-hard for directed and undirected graphs. Also the problem of finding a Hamiltonian tour of minimum reload cost in an Hamiltonian graph (or directed graph) G is NP-hard. This can be shown by adapting the original reduction in [3] from the minimum vertex cover problem to the Hamiltonian tour problem.

4. Flows

Given any directed graph with arc capacities, unitary costs on arcs, and unitary reload costs, and an origin s and destination t, an $s-t$ flow of minimum transportation cost can be found in polynomial time. Indeed, by applying the splitting procedure of Section 2 to all nodes except the origin and the destination, the problem reduces to that of finding a minimum (arc) cost $s-t$ flow in the new network H. Note that in this case the line-graph does not help.

Let us now consider the network flow problem variants with multiple origin-destination pairs $\{(s_i, t_i)\}_{1 \leq i \leq q}$, the so-called multicommodity versions. If the flow is unsplittable (must be routed along a single path for each origin-destination pair) and there are capacities constraints on the arcs, the resulting capacitated minimum multicommodity transportation cost flow problem can be shown to be NP-hard even without

arc costs. The reduction is from the shortest arc-disjoint q paths problem, which is shown NP-hard in [4]. If the flow is unsplittable but there are no arc capacities, the problem is solvable in polynomial time because of the previous results about minimum reload cost paths. Given any uncapacitated network G with both arc costs and reload costs, an optimum solution is obtained by just superimposing the set of paths derived by independently applying the splitting procedure of Section 2 for each origin-destination pair.

If the flow is splittable (can be routed along different paths for each origin-destination pair) and there are arc capacities, the minimum multicommodity transportation cost flow problem is polynomially solvable because it can be reduced to that of finding a splittable multicommodity flow of minimum (arc) cost. This is achieved by applying the splitting procedure to all the nodes and by adding to each complete bipartite graph corresponding to an origin (destination) of G a new left shore node S_i (right shore node T_i) connected with arcs from S_i to all nodes of the right shore (from all nodes of the left shore to T_i). All these additional arcs have zero cost and unbounded capacity. Depending on whether the node in the original graph G is an origin s_i or a destination t_i, the corresponding node S_i or T_i is considered as the origin or destination in the new network.

5. Concluding remarks

We have presented a first set of complexity results for several network optimization problems under a natural reload cost model. We are currently investigating more efficient algorithms for the problems that are solvable in polynomial time, and polynomial-time algorithms with worst-case approximation guarantee for those that are NP-hard. Since in some applications it may not be appropriate to mix arc costs and reload costs, it would be interesting to investigate bi-criteria problem versions, where one type of cost is minimized while keeping the other one below a given threshold.

References

[1] G. Galbiati: The complexity of a Minimum Reload Cost Diameter Problem. *Discrete Appl. Math.*, to appear.

[2] I. Gamvros, L. Gouveia, S. Raghavan: Reload Cost Trees and Network Design. *Proc. International Network Optimization Conference* (INOC 2007), Spa, 2007, paper n.117.

[3] M.R. Garey , D.S. Johnson : *Computers and Intractability: A Guide to the theory of NP-Completeness*. W.H. Freeman and company, NY, 1979.

[4] U. Brandes, W. Schlickenrieder, G. Neyer, D. Wagner, K. Weihe: A software package of algorithms and heuristics for disjoint paths in Planar Networks. *Discrete Appl. Math.* 92: 91–110, 1999.

[5] H. Wirth, J. Steffan: Reload cost problems: minimum diameter spaning tree. *Discrete Appl. Math.* 113: 73–85, 2001.

Coloring (II)

Room A, Session 5

Wednesday 14, 11:00-12:30

A decomposition for total-coloring graphs of maximum degree 3

Raphael Machado [a,*] Celina de Figueiredo [a]

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

Abstract

The total chromatic number $\chi_T(G)$ is the least number of colors needed for coloring the elements (vertices and edges) of a graph G in such a way no incident or adjacent elements receive the same color. Deciding whether a graph G is Type 1, that is, $\chi_T(G) = \Delta(G)+1$, is NP-complete [6], even when restricted to cubic bipartite inputs. We develop a decomposition technique for searching for a 4-total-coloring of graphs with maximum degree 3. We use this decomposition tool for extending a result on the total chromatic number of partial-grids, a subclass of bipartite graphs.

Key words: total chromatic number, graph decompositions, partial-grids.

1. Introduction

Let G be a simple graph with vertex-set $V(G)$ and edge-set $E(G)$. The set of the *elements* of G is $S(G) = V(G) \cup E(G)$. Two vertices $u, v \in V(G)$ are *adjacent* if $uv \in E(G)$; two edges $e_1, e_2 \in E(G)$ are *adjacent* if they share a common endvertex; a vertex u and an edge e are *incident* if u is an endvertex of e. A *cut* of a graph G is a set of vertices whose exclusion disconnects G. A cut composed of two adjacent vertices is said to be a K_2*-cut*.

A *total-coloring* of G is a function $\pi : S(G) \to \mathcal{C}$ such that, for **no** pair of adjacent or incident elements $x, y \in S(G)$, it holds $\pi(x) = \pi(y)$. If $|\mathcal{C}| = k$, then π is a k*-total-coloring*. The *total chromatic number* of G, denoted $\chi_T(G)$, is the least k for which G has a k-total-coloring. Clearly, $\chi_T(G) \geq \Delta(G) + 1$, where $\Delta(G)$ is the maximum degree of G; if $\chi_T(G) = \Delta(G) + 1$, then G is *Type 1*. The Total Coloring Conjecture [1, 7], which states that every graph G has total chromatic number $\Delta(G) + 1$ or $\Delta(G) + 2$, is open since 1964.

* Corresponding author.

2. The decomposition technique

We present a decomposition technique for total-coloring graphs of maximum degree 3. The technique decomposes a graph into subgraphs which do not have a K_2-cut. The following lemma, stated without proof, considers the total-coloring of the biconnected components of a graph.

Lemma 1 *Let G be a graph such that all of its biconnected components have an α-total-coloring, where $\alpha \geq \Delta(G) + 1$. Then G itself has an α-total-coloring.*

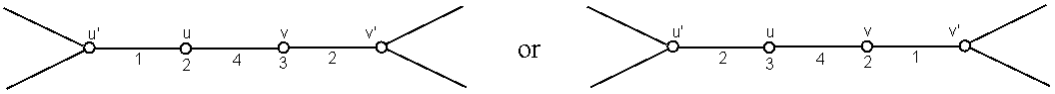
The *K_2-cut-free components of G* are the subgraphs of G which do not have a K_2-cut and are maximal with respect to this property. The following result, which we state without proof, is important to our strategy for total-coloring.

Lemma 2 *Let G be a biconnected graph of maximum degree 3 and $\mathcal{G}$ the collection of its K_2-cut-free components. The intersection graph of $\mathcal{G}$ is acyclic.*

Given a biconnected graph G of maximum degree at most 3 and two adjacent vertices u and v of degree 2 in G, we say that the set $C = \{u, v\}$ is a *frontier candidate*. Let $u' \notin C$ and $v' \notin C$ be the vertices adjacent, respectively, to u and v. We refer to u, v, $u'u$, uv and vv' as the *elements at the frontier-candidate* $\{u, v\}$. Let $\mathcal{F}(G) = \{\{u_1, v_1\}, ..., \{u_r, v_r\}\}$ be the collection of all frontier-candidates of G. We say that a 4-total-coloring π of G is a *frontier-coloring of G* if:

$$\pi(u_i u_i') \neq \pi(v_i v_i'), \text{ and } \pi(\{u_i u_i', v_i v_i', u_i, v_i, u_i v_i\}) = \{1, 2, 3, 4\}$$

for each $\{u_i, v_i\} \in \mathcal{F}(G)$, where $u_i' \notin \{u_i, v_i\}$ and $v_i' \notin \{u_i, v_i\}$ are the vertices adjacent, respectively, to u_i and v_i, for each $i = 1, ..., r$. Observe that the elements at a frontier-candidate $\{u, v\}$ are colored in one of the following ways (except for a permutation of colors):



In the first case, we say that u is the *reference vertex of* $(\pi, \{u, v\})$; in the second case, v is the reference vertex of $(\pi, \{u, v\})$. Now, we state a result that shows how to color a biconnected graph from frontier-colorings of its K_2-cut-free components.

Theorem 1 *Let G be a biconnected graph of maximum degree at most 3. Suppose every K_2-cut-free component G_ℓ of G has two frontier-colorings $\pi_{\ell,a}$ and $\pi_{\ell,b}$ such that, for each frontier-candidate $\{u, v\}$ of G_ℓ, vertex u is the reference vertex of $(\pi_{\ell,a}, \{u, v\})$ if and only if vertex v is the reference vertex of $(\pi_{\ell,b}, \{u, v\})$. Then G is 4-total-colorable.*

Proof (sketch): Let $\mathcal{G} = \{G_1, ..., G_\ell\}$ be the family of the K_2-cut-free components of G and let $\mathcal{I}(\mathcal{G})$ be its intersection graph. Now consider a total order $G_{i_1}, ..., G_{i_\ell}$ on $\mathcal{G}$ given by a breadth first search on the tree $I(\mathcal{G})$ starting at an arbitrary element $G_{i_1} \in \mathcal{G}$ and let $H_j = G[V(G_{i_1}) \cup ... \cup V(G_{i_j})]$. Observe that $H_\ell = G$. By hypothesis, $H_1 = G_{i_1}$ has a frontier-coloring π_1, that is a 4-total-coloring which

satisfies the frontier condition for the set of all of its frontier-candidates. We prove the lemma by induction on j. □

3. Partial-grids

A graph $G_{m\times n}$ with vertex set $V(G_{m\times n}) = \{1, ..., m\} \times \{1, ..., n\}$ and edge set $E(G_{m\times n}) = \{(i, j)(k, l) : |i - k| + |j - l| = 1, (i, j), (k, l) \in V(G_{m\times n})\}$, or a graph isomorphic to $G_{m\times n}$, is a *grid*. A *partial-grid* is an arbitrary subgraph of a grid. Partial-grids are harder to work than grids; for instance, recognition of grids is polynomial [3], but the problem is open for partial-grids [2].

The total chromatic number of grids has been determined [4]. The total chromatic number of partial-grids were determined [4] for all cases where the maximum degree is not 3; if the maximum degree is 3, some cases could be classified: partial grids with at most three maximum degree vertices or with maximum induced cycle of size at most 4 are Type 1. The last step towards a complete classification of partial grids is the case of maximum degree 3.

A graph is c-chordal [5] if it does not have an induced cycle larger than c. In this section, we show that the decomposition technique developed in Section 2 provides a nice method for total-coloring partial-grids with bounded size maximum induced cycle. The applicability of this tool comes from the fact that, for each fixed c, there exists only a finite number of c-chordal K_2-cut-free subgraphs of partial-grids. So, the task of determining the total chromatic number of c-chordal partial-grids of maximum degree 3 may be reduced to that of exhibiting colorings for a finite number of graphs. We use Theorem 1 for classifying as Type 1 all 8-chordal partial grids of maximum degree 3.

Theorem 2 *All partial-grids of maximum degree at most 3 and maximum induced cycle of length at most 8 are 4-total-colorable.*

Proof (sketch): The biconnected components of a partial-grid are itself partial-grids. So, by Lemma 1, we just need to prove the Theorem for biconnected partial-grids. Besides, every K_2-cut-free component of a c-chordal partial-grid is itself a c-chordal partial-grid. So, all we need to do is to exhibit, for each 8-chordal K_2-cut-free partial-grid of maximum degree at most 3, colorings satisfying the conditions of Theorem 1 (see Appendix 4). □

4. Final Remarks

Total-coloring is notably a challenging problem. The total-coloring conjecture is open since 1964, and determining whether a graph is Type 1 is NP-Complete even for very restricted graph classes, such as cubic bipartite graphs. So, it is of great

value developing techniques for total-coloring graphs of maximum degree 3. The proposed decomposition is an original technique, which we apply for total-coloring partial-grids, extending a known result [4] for these graphs.

Future work. Our main goal is to prove the conjecture [4] that all partial-grids of maximum degree 3 are Type 1. We are also investigating new classes for which our approach could be applied. One of those classes are the maximum degree 3 partial d-dimensional grids, which are subgraphs of d-dimensional grids. We observe that the d-dimensional grids can be total-colored using the results of [8]; the particular case of d-cubes was solved independently [4]. Another line of investigation is trying to extend our decomposition tool by using cuts larger than K_2. Those cuts could, hopefully, be applied for total-coloring graphs with maximum degree larger than 3.

References

[1] M. Behzad. Graphs and their chromatic numbers. Ph.D. Thesis, Michigan State University, 1965.

[2] A. Brandstädt, V.B. Le, T. Szymczak, F. Siegemund, H.N. de Ridder, S. Knorr, M. Rzehak, M. Mowitz, N. Ryabova, U. Nagel, D. Bayer. Information System on Graph Class Inclusions v2.0 (http://wwwteo.informatik.uni-rostock.de/isgci/). Last visited 14 September 2007.

[3] G. Burosch and J.-M. Laborde. Characterization of grid graphs. Discrete Math. 87 (1991) 85-88.

[4] C. N. Campos, C. P. Mello. The total chromatic number of some bipartite graphs. Electron. Notes in Discrete Math. 22 (2005) 557-561. (Full paper to appear in Ars Combin.)

[5] D. G. Corneil, F. F. Dragan, E. Köhler. On the power of BFS to determine a graph's diameter. Networks 42 (2003) 209-222.

[6] C. J. H. McDiarmid, A. Sánches-Arroyo. Total colouring regular bipartite graphs is NP-hard. Discrete Math. 124 (1994) 155-162.

[7] V. G. Vizing. On an estimate of the chromatic class of a p-graph. Metody Diskret. Analiz. 3 (1964) 25-30. In Russian.

[8] B. Zmazek and J. Őerovnik. Behzad-Vizing conjecture and Cartesian-product graphs. Appl. Math. Lett. 15 (2002) 781-784.

Appendix. Colorings of all 8-chordal K_2-cut-free partial-grids of maximum degree at most 3

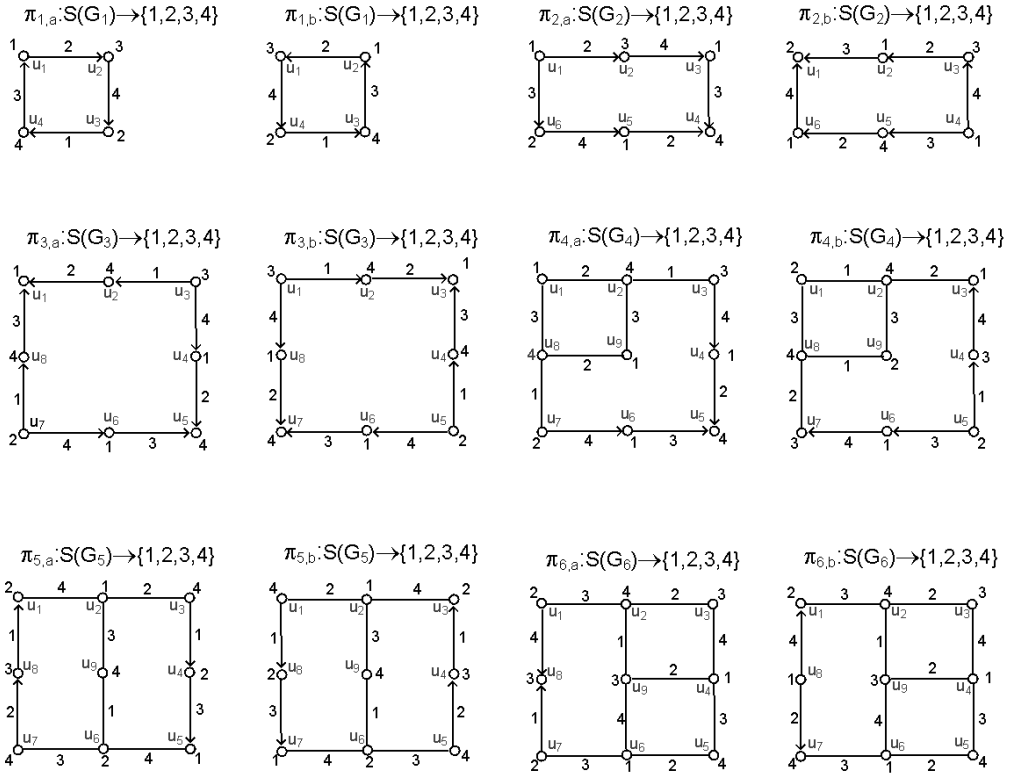


Fig. 1. Total-colorings satisfying the conditions of Theorem 1. (With the aim of indicating the reference vertex at a possible frontier $\{u, v\}$, the edge uv will be represented by an arrow pointing to the reference vertex. We emphasize, nevertheless, that we are **not** dealing with directed graphs.) Observe that there exist six 8-chordal K_2-cut-free partial-grids of maximum degree at most 3 and that, for each of these graphs, we indicate two 4-total-colorings. Each coloring is a frontier-coloring (that is, it satisfies the frontier condition for all frontier-candidates). Observe that, for each indecomposable graph G_ℓ, $\ell = 1, \ldots, 6$, there are two colorings $\pi_{\ell,a}$ and $\pi_{\ell,b}$ such that, for each frontier candidate $\{u_i, u_j\}$ of G_ℓ, vertex u_i is a reference vertex of $(\pi_{\ell,a}, \{u_i, u_j\})$ if and only if vertex u_j is a reference vertex of $(\pi_{\ell,b}, \{u_i, u_j\})$. So, the conditions of Theorem 1 are satisfied.

Using Latin Squares to Color Split Graphs [1]

Sheila Morais de Almeida [a] Célia Picinin de Mello [a]
Aurora Morgana [b,*]

[a] *Institute of Computing, University of Campinas, Brazil*
[b] *Department of Mathematics, University of Rome "La Sapienza", Italy*

Key words: edge-coloring, latin square, split graphs, classification problem.

1. Introduction

In this note, G denotes a simple, undirected, finite, and connected graph with vertex set $V(G)$ and edge set $E(G)$. For any v in $V(G)$, the *degree* of v, denoted $d_G(v)$, is the number of vertices adjacent to v in G. A *clique* is a set of pairwise adjacent vertices of G. A *maximal clique* is a clique that is not properly contained in any other clique. The size of a maximum clique of G is denoted by $\omega(G)$. A *stable set* is a set of pairwise non adjacent vertices of G. An *edge-coloring* of G is an assignment of one color to each edge of G such that no adjacent edges have the same color. The *chromatic index*, χ', is the minimum number of colors for which G has an edge-coloring. A known theorem by Vizing [11] states that, for a simple graph, the chromatic index is at most $\Delta(G)+1$, where $\Delta(G)$ is the maximum vertex degree. Graphs whose chromatic index equals $\Delta(G)$ are *Class* 1; graphs whose chromatic index equals $\Delta(G)+1$ are *Class* 2. Despite the restriction imposed by Vizing, it is NP-complete to determine, in general, if a graph is Class 1 [6]. There are not many graph classes for which the problem is known to be polynomial; see [5, 7, 9] for examples. The complexity of the problem is open for very structured classes of graphs such as cographs, proper interval graphs and split graphs.

A graph G satisfying the inequality $|E(G)| > \Delta(G)\left\lfloor \frac{|V(G)|}{2} \right\rfloor$, is said to be an *overfull* graph. A graph G is *subgraph-overfull* when it has an overfull subgraph H with $\Delta(H) = \Delta(G)$ [4]. When the overfull subgraph H can be chosen to be a *neighbourhood*, we say that G is *neighbourhood-overfull* [3]. Overfull, subgraph-overfull, and neighbourhood-overfull graphs are in Class 2.

* Corresponding author.
[1] This research was partially supported by CAPES and CNPq (482521/2007-4 and 300934/2006-8).

A *split graph* is a graph whose vertex set admits a partition into a stable set and a clique. Split graphs is a well-studied class of graphs for which most combinatorial problems are solved [2, 8, 10]. It has been shown that every odd maximum degree split graph is Class 1 [1] and that every subgraph-overfull split graph is in fact neighbourhood-overfull [3]. It has been conjectured that every Class 2 split graph is neighbourhood-overfull [3]. The validity of this conjecture implies that the edge-coloring problem for split graphs is in P. The goal of this paper is to investigate this conjecture by giving another positive evidence for its validity. We describe a new subset of split graphs with even maximum degree that is Class 1. Using latin squares, we construct a polynomial edge-coloring for these graphs.

2. A split graph subset that is not neighbourhood-overfull

A *color diagram* $C = (C_1, \ldots, C_k)$ is a sequence of color arrays, where each color array $C_i = [c_{i,1}, \ldots, c_{i,d_i}]$, $1 \leq i \leq k$, consists of distinct colors. A color diagram C is *monotonic* if the color $c_{i,j}$ occurs at most $d_i - j$ times in $C_1, \ldots, C_{i-1}$ for all $1 \leq i \leq k$ and $1 \leq j \leq d_i$.

A monotonic color diagram can be used to provide an edge-coloring for a bipartite graph. Let B be a bipartite graph with bipartition $\{U, V\}$. If $C = (C_{u_1}, \ldots, C_{u_k})$ is a monotonic color diagram where, for each vertex $u_i \in U$, C_{u_i} is a set of $d_B(u_i)$ distinct colors, then B has an edge-coloring that uses the colors of C_{u_i} to color the edges incident to u_i, $u_i \in U$ [1]. We use this result in our study of edge-coloring of split graphs.

A $k \times k$-matrix with entries from $\{0, \ldots, k-1\}$ is called *latin square of order* k if every element of $\{0, \ldots, k-1\}$ appears in each row and column exactly once. A latin square of order k, $M = [m_{i,j}]$, is *commutative* if $m_{i,j} = m_{j,i}$, for $0 \leq i \leq j \leq k-1$ and it is idempotent if $m_{i,i} = i$, for $0 \leq i \leq k-1$.

From now on, G is a split graph with a partition $\{Q, S\}$, where Q is a maximal clique and S is a stable set. Note that Q is also a maximum clique. To every split graph G we shall associate the bipartite graph B obtained from G by removing all edges of the subgraph of G induced by Q. Let $d(Q)$ be the maximum degree of vertices of Q in the bipartite graph B, i.e., $d(Q) = \max_{v \in Q} d_B(v)$. Then $\Delta(G) = \omega(G) - 1 + d(Q)$.

In [1], Chen, Fu and Ko, use an odd order idempotent commutative latin square to show that an odd maximum degree split graph is Class 1. It is known that there is an idempotent commutative latin square of order n if, only if, n is odd. Hence, the technique presented in [1] could not be directly applied on split graphs with even maximum degree.

In order to provide an edge-coloring with $\Delta(G)$ colors for some split graphs when $\Delta(G)$ is even, we consider a matrix, $M = [m_{i,j}]$, $0 \leq i, j \leq \Delta(G) - 2$, where $m_{i,j} = (i + j) \pmod{(\Delta(G) - 1)}$. From now on, the entries of a matrix are called

colors. The matrix M is a commutative latin square of order $\Delta(G)-1$, so we need a new color. We replace some entries of M with the new color, as described in Algorithm ColorDiagrams. Then, we consider a vertex v in S with degree at least $\frac{\omega(G)}{2}$ and we label the vertices of Q as $u_1, u_2, \ldots, u_{\omega(G)}$ such that u_i is adjacent to v, $0 \leq i \leq d_G(v)$. Then, we use the color $a_{i,i}$ of the submatrix $A = [a_{i,j}]$ formed by the first $\omega(G)$ rows and columns of M to color the edge (v, u_i), $0 \leq i \leq d_G(v)$. After, we use the color $a_{i,j}$ to color the edge (u_i, u_j) of the subgraph of G induced by Q, $0 \leq i, j \leq \omega(G)$. Now, it remains to color $d(Q)-1$ edges of B and we use the monotonic color diagram C produced by our algorithm to color these edges. The constraints of Theorem 1 are given by this strategy. The algorithm used in our approach follows.

Algorithm ColorDiagrams($\Delta(G)$, $\omega(G)$, $d_G(v)$)
Construct a $(\Delta(G)-1) \times (\Delta(G)-1)$-matrix M where
$m_{i,j} = (i+j) \pmod{\Delta(G)-1}$.
Construct a sequence $C = (C_0, \ldots, C_{\omega(G)-1})$, where
$C_i = [m_{i,\omega(G)}, \ldots, m_{i,\Delta(G)-2}]$, $0 \leq i < \omega(G)$.
Add $m_{i,i}$ as the first element of C_i, $d_G(v) \leq i < \omega(G)$.
Add $\Delta(G)-1$ as the first element of C_i, $0 \leq i < \omega(G)$.
Construct a matrix $A_{\omega(G),\omega(G)}$, where $a_{i,j} \leftarrow m_{i,j}$, $0 \leq i, j < \omega(G)$;
$l \leftarrow 0$; $l' \leftarrow \omega(G)-1$; $x \leftarrow -1$; $c \leftarrow \omega(G)+x$;
If c is odd, then $count \leftarrow \lfloor \frac{\Delta(G)-\omega(G)-x-1}{2} \rfloor$, else $count \leftarrow \lfloor \frac{\Delta(G)-\omega(G)-x-2}{2} \rfloor$;
While ($l < l'$) and ($c < \Delta(G)-2$) do
  Replace the color c from $a_{l,l'}$ and $a_{l',l}$ of A with $\Delta(G)-1$;
  Replace the color $\Delta(G)-1$ of C_l and $C_{l'}$ with c;
  $l \leftarrow l+1$; $l' \leftarrow l'-1$; $count \leftarrow count-1$;
  if $count = 0$, then
    $x \leftarrow x+1$; $c \leftarrow \omega(G)+x$; $l \leftarrow l+1$;
    if c is odd, then $count \leftarrow \lfloor \frac{\Delta(G)-\omega(G)-x-1}{2} \rfloor$;
    if c is even, then $count \leftarrow \lfloor \frac{\Delta(G)-\omega(G)-x-2}{2} \rfloor$;
Return(A, C).

The following results are used in the proof of Theorem 1.

Lemma 1 *The matrix A returned by Algorithm ColorDiagrams is commutative, its elements are from $\{0, ..., \Delta(G)-1\}$, and it has pairwise distinct elements in each line and column. Moreover, if $\Delta(G)$ is even, the elements of the main diagonal of A are pairwise distinct.*

Lemma 2 *Let G be a split graph with even $\Delta(G)$. If G has a vertex v in S with $d_G(v) \geq \frac{\omega(G)}{2}$ and $(d(Q))^2 \geq 2\omega(G)+1$, then the sequence C returned by the Algorithm ColorDiagrams is a monotonic color diagram.*

Theorem 1 *Let G be a split graph with even $\Delta(G)$. If G has a vertex v in S with degree at least $\frac{\omega(G)}{2}$ and $(d(Q))^2 \geq 2\omega(G)+1$, then G is Class 1.*

A split graph satisfying the conditions of Theorem 1 is not neighbourhood-overfull. So, our result gives a positive evidence to the conjecture that for any split graph neighbourhood-overfullness is equivalent to being Class 2.

References

[1] B-L. Chen, H-L. Fu, and M. T. Ko. Total chromatic number and chromatic index of split graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 17:137–146, 1995.

[2] D. G. Corneil and Y. Perl. Clustering and domination in perfect graphs. *Discrete Applied Mathematics*, 9:27–39, 1984.

[3] C. M. H. de Figueiredo, J. Meidanis, and C. P. de Mello. Local conditions for edge-colouring. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 32:79–91, 2000.

[4] A. J. W. Hilton. Two conjectures on edge-colouring. *Discrete Mathematics*, 74:61–64, 1989.

[5] D. G. Hoffman and C. A. Rodger. The chromatic index of complete multipartite graphs. *Journal of Graph Theory*, 16:159–163, 1992.

[6] I. Holyer. The NP-completeness of edge-coloring. *SIAM Journal of Computing*, 10:718–720, 1981.

[7] D. König. Über Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre. *Mathematische Annalen*, 77:453–465, 1916.

[8] C. Ortiz, N. Maculan and J. L. Szwarcfiter. Characterizing and edge-colouring split-indifference graphs. *Discrete Applied Mathematics*, 82:209–217, 1998.

[9] M. Plantholt. The chromatic index of graphs with a spanning star. *Journal of Graph Theory*, 5:45–53, 1981.

[10] G. F. Royle. Counting set covers and split graphs. *Journal of Integer Sequences*, 3:1–5, 2000.

[11] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Metody Diskret. Analiz.*, 3:25–30, 1964.

On total chromatic number of direct product graphs

Katja Prnaver [a,*] Blaz Zmazek [b]

[a] *IMFM, Jadranska 19, Ljubljana, Slovenia*

[b] *University of Maribor, FME, Smetanova 17, 2000 Maribor, Slovenia*

Key words: total chromatic number, total coloring, direct product, tensor product

1. Introduction

All the graphs in this paper are considered to be undirected, finite and contain no loops or multiple edges. The *direct product* $G \times H$ of graphs G and H is a graph with $V(G \times H) = V(G) \times V(H)$ and $E(G \times H) = \{\{(a,x),(b,y)\}|\{a,b\} \in E(G) \text{and} \{x,y\} \in E(H)\}$. This product is also known as Kronecker product, tensor product, categorical product and graph conjunction.
A *vertex coloring* is an assignment of labels or colors to each vertex of a graph G such that no edge connects two identically colored vertices. The minimum number of colors required to color the graph G is called the *chromatic number*, denoted by $\chi(G)$. An *edge coloring* of a graph G is a coloring of the edges of G such that adjacent edges receive different colors. The minimium number of colors required for such a coloring is called *edge chromatic number* and denoted by $\chi'(G)$.
For $S \subset V(G) \bigcup E(G)$, a *partial total coloring* of G, is a mapping $\varphi : S \to C$ such that, for each adjacent or incident elements $x, y \in S$, we have $\varphi(x) \neq \varphi(y)$. If $S = V(G) \bigcup E(G)$, then φ is a *total coloring*. The least integer k, for which $|C| = k$ and φ is total coloring, is called the *total chromatic number* of G and is denoted by $\chi''(G)$ or sometimes also by $\chi_T(G)$.Clearly, $\chi''(G) \geq \Delta(G) + 1$. The *Total Coloring Conjecture (TCC)*, posed independently by Behzad[11] and Vizing[2], states that every simple graph G has $\chi''(G) \leq \Delta(G) + 2$. If $\chi''(G) = \Delta(G) + 1$, then G is a type 1 graph; if $\chi''(G) = \Delta(G) + 2$, then G is a type 2 graph.
The TCC has been confirmed for cartesian product of graphs G and H, if the TCC holds for the graphs G and H by Zmazek, Zerovnik [7] and for the powers of cycles C_k^k by Campos, Mello [5].Here we confirm the TCC for direct product of a path, P_n, and a graph G, where G is type 1 graph. We further investigate the total chromatic number of direct product of a path and an arbitrary cycle.

* Corresponding author.

2. Total coloring of $P_n \times P_m$

We start by investigating a product of two paths, P_n and P_m, $n, m \in \mathbb{N}$.
Theorem 1 $\chi''(P_n \times P_m) = 5$.

PROOF. Clearly, $\chi'(P_n) = 2$. Now color the edges of P_n with colors 1 and 2, and color the vertexes of P_m with colors $\{1, 2, ..., m\}$ in natural order. Define the function φ as following:

$$\varphi : S \to C''$$
$$\varphi((g,h)) = 2 \cdot C(h) + 3(\mathrm{mod}5)$$
$$\varphi((g,h),(g',h')) = C'(g,g') + 2 \cdot C(h)(\mathrm{mod}5); h < h'$$

where $S = V(G) \bigcup E(G), C'' = \{0,1,2,3,4\}$ and C and C' correspond to coloring of vertexes and edges respectively as described above.
Take an arbitrary vertex, say (g,h). The neighbours of this vertex are $(g-1, h-1), (g+1, h-1), (g-1, h+1), (g+1, h+1)$. As the vertex color depends only on the second coordinate of the vertex, showing that color assigned to two vertexes with consecutive second coordinates is different is sufficient. Take (g,h) and $(g+1, h+1)$:

$$\begin{aligned}\varphi(g,h) - \varphi(g+1,h+1) &= 2 \cdot C(h) - 2 - 2 \cdot C(h+1) + 2(\mathrm{mod}5) \\ &= 2 \cdot C(h) - 2(\cdot C(h) + 1) = -2(\mathrm{mod}5) \neq 0.\end{aligned}$$

Now observe the edges surrounding the vertex (g,h). The colors of these edges are:

- $\varphi((g,h)(g-1,h-1)) = C'(g,g-1) + 2 \cdot C(h-1)$
- $\varphi((g,h)(g-1,h+1)) = C'(g,g-1) + 2 \cdot C(h)$
- $\varphi((g,h)(g+1,h-1)) = C'(g,g+1) + 2 \cdot C(h-1)$
- $\varphi((g,h)(g+1,h+1)) = C'(g,g+1) + 2 \cdot C(h)$

Since $C'(g,g+1) - 1 = C'(g,g-1) = 1$ (without loss of generality) and $C(h-1) + 1 = C(h) = x$, the values are $2x, 2x+1, 2x+4, 2x+2$ respectively. Observe that all the edges have different color. Since $\varphi(g,h) = 2x + 3(\mathrm{mod}5)$ it is also proven that the color of the vertex differs from the colors of adjacent edges. □

The proof also provides an algorithm for total coloring of such a graph. It can be understood in simpler way. If the coloring of edges as described above is used, all the vertexes in same G-fibre have edges of same colors adjacent. As the maximum degree of a vertex in a G-fibre is 4, there is one color from C'' set not used and can be used for color of the vertexes in the fibre.

3. Total coloring of $G \times P_n$

The idea of coloring of $P_n \times P_m$, where one of the paths had no impact on the coloring of the vertexes and edges in the direct product, can be further developed to coloring of a direct product of path and arbitrary graph G.

Theorem 2 $\chi''(G \times P_n) = \Delta(G \times P_n) + 1$, *if* $\chi'(G) = \Delta(G)$.

PROOF. Proof of this theorem is done is similar way as in proof of theorem 1, by introducing function φ as:

$$\varphi : S \to C''$$
$$\varphi((g,h)) = \chi'(G) \cdot (C(h) + 1)(\mathrm{mod}\Theta)$$
$$\varphi((g,h),(g',h')) = C'(g,g') + \chi'(G) \cdot C(h)(\mathrm{mod}\Theta); h < h'$$

where $\Theta = (\Delta(G) \cdot 2 + 1)$ and $C'' = \{0, 1, 2, ..., \Theta - 1\}$.
Again, the colors of adjacent vertexes, edges adjacent to arbitrary vertex and the end vertexes of such edges much be verified to have different colors assigned.
The details of this proof will be given in full paper. □

Again, each of the possible $\Delta(G)\Delta(P_n)$ edges in G-fibre gets different color from set C'' assigned and the remaining one can be used to color the vertex.

Remark 1 *The function used in the proof will also produce total coloring of a graph* $G \times P_n$*, if the division is done by* $\Theta = \Delta(G) \cdot 2 + 3$ *and* $\chi'(G) = \Delta(G) + 1$. *However, such coloring will use* $\Delta(G \times P_n) + 3$ *colors which does not match the conjecture. Better colorings exist in this case .*

4. Total coloring of $C_m \times P_n$

Coloring of $C_{2k} \times P_n$ is an immediate corollary of Theorem 2, however we will prove that the theorem holds for all the cycles.

Theorem 3 $\chi'(C_m \times P_n) = 5$

PROOF. The theorem only needs to be proven for case where $m = 2k+1$. We will prove this by observing that the direct product $C_{2k+1} \times P_2$ produces an even cycle C_{2k+2}. There exists total coloring of such an even cycle with 4 colors [10]: since the sum of vertexes and edges in such a cycle is $4(k+1)$, we can color it by sequentially exchanging the colors $\{1, 2, 3, 4\}$. Such an cycle is also bipartite, so the vertexes in each of partitions will be colored in one color. These partitions correspond to two G-fibres if the cycle is observed as product of $C_{2k+1} \times P_2$. Without loss of generality, let the colors of vertexes in the colored $(2k + 1)$-cycle be $\{2, 4\}$ and the edges of colors $\{1, 3\}$.
The graph $C_m \times P_n$ is constructed of n copies of such a construction. If the first copy is colored as described above, the colors of edges, i.e. 1 and 3 cannot be used

for edge colors in next copy. One partition of next copy's graph will be the partition of first copy, so one color for the vertexes will already be defined. Now use of of the colors $\{1, 3\}$ for the other partition vertexes and use the colors 5 and the color of the other partition from first copy as colors for edges in this next copy.
Same procedure of assigning colors can be applied to all the next copies until we color all the copies of $C_m \times P_2$. □

References

[1] M. Behzad. Graphs and their chromatic numbers, Ph.D. Thesis, Michigan State University, 1965.

[2] V.G. Vizing. On an estimate of the chromatic class of a p-graph, Metody Diskret. Anal. 3, 25Ű30 (1964) (in Russian).

[3] H.P.Yap. Total colorings of graphs, in: Lecture Notes in Mathematics, vol. 1623, Springer, Berlin, p. 6. (1996)

[4] B. Zmazek, J. Zerovnik. Behzad-Vizing conjecture and Cartesian product graphs, Appl. Math. Lett. 15, No.6, 781-784 (2002)

[5] C.N. Campos, C.P. de Mello, A result on the total colouring of powers of cycles, Discrete Appl. Math. 155, No. 5, 585-597 (2007)

Routing

Room B, Session 5

Wednesday 14, 11:00-12:30

Bidirectional A^* on Time-dependent Graphs [1]

Giacomo Nannicini [a,b,*] Daniel Delling [c] Leo Liberti [b]
Dominik Schultes [c]

[a] *Mediamobile, 10 rue d'Oradour sur Glane, 75015 Paris*
[b] *LIX, École Polytechnique, 91128 Palaiseau, France*
[c] *Universität Karlsruhe (TH), 76128 Karlsruhe, Germany*

Key words: Shortest paths, Time-dependent graph, Goal-directed search

1. Introduction

We consider the TIME-DEPENDENT SHORTEST PATH PROBLEM (TDSPP): given a directed graph $G = (V, A)$, a source node $s \in V$, a destination node $t \in V$, an interval of time instants T, a departure time $\tau_0 \in T$ and a time-dependent arc weight function $c : A \times T \to \mathbb{R}_+$, find a path $p = (s = v_1, \ldots, v_k = t)$ in G such that its *time-dependent cost* $\gamma_{\tau_0}(p)$, defined recursively as follows:

$$\gamma_{\tau_0}(v_1, v_2) = c(v_1, v_2, \tau_0) \tag{1}$$
$$\gamma_{\tau_0}(v_1, \ldots, v_i) = \gamma_{\tau_0}(v_1, \ldots, v_{i-1}) + c(v_{i-1}, v_i, \tau_0 + \gamma_{\tau_0}(v_1, \ldots, v_{i-1})) \tag{2}$$

for all $2 \leq i \leq k$, is minimum. We also consider a function $\lambda : A \to \mathbb{R}_+$ such that $\forall (u, v) \in A, \tau \in T \quad (\lambda(u, v) \leq c(u, v, \tau))$. In other words, $\lambda(u, v)$ is a lower bound on the travelling time of arc (u, v) for all time instants in T. In practice, this can easily be computed, given an arc length and the maximum allowed speed on that arc. We naturally extend λ to be defined on paths, i.e. $\lambda(p) = \sum_{(v_i, v_j) \in p} \lambda(v_i, v_j)$. In this paper, we propose a novel algorithm for the TDSPP based on a bidirectional A^* algorithm. Since the arrival time is not known in advance (so c cannot be evaluated on the arcs adjacent to the destination node), our backward search occurs on the graph weighted by the lower bounding function λ. This is used for bounding the set of nodes that will be explored by the forward search. We assume that the graph has the FIFO property.

* Corresponding author.
[1] Partially supported by the Future and Emerging Technologies Unit of EC (IST priority – 6th FP), under contract no. FP6-02123502 (project ARRIVAL), and by DFG grant SA 933/1-3.

2. A^* with Landmarks

A^* is an algorithm for goal-directed search, similar to Dijkstra's algorithm, but which adds a potential function π to the priority key of each node in the queue. One way to compute the potential function, instead of using Euclidean distances, is to use the concept of *landmarks*. Landmarks have first been proposed in [2]; they are a preprocessing technique which is based on the triangular inequality. The basic principle is as follows: suppose we have selected a set $L \subset V$ of landmarks, and we have precomputed distances $d(v,\ell), d(\ell,v)\, \forall v \in V, \ell \in L$; the following triangle inequalities hold: $d(u,t) + d(t,\ell) \geq d(u,\ell)$ and $d(\ell,u) + d(u,t) \geq d(\ell,t)$. Therefore $\pi_t(u) = \max_{\ell \in L}\{d(u,\ell) - d(t,\ell), d(\ell,t) - d(\ell,u)\}$ is a lower bound for the distance $d(u,t)$, and it can be used as a potential function which preserves optimal paths. On a static graph (i.e. non time-dependent), bidirectional search can be implemented, using some care in modifying the potential function so that it is consistent for the forward and backward search (see [3]). Bidirectional A^* with the potential function described above is called ALT. It is straightforward to note that, if arc costs can only increase with respect to their original value, the potential function associated with landmarks is still a valid lower bound, even on a time-dependent graph. Unidirectional ALT in a time-dependent scenario has been tested in [1].

3. Bidirectional Search on Time-Dependent Graphs

Our algorithm is based on restricting the scope of a time-dependent A^* search from the source using a set of nodes defined by a time-*independent* A^* search from the destination, i.e. the backward search is a reverse search in G_λ, which corresponds to the graph G weighted by the lower bounding function λ.

Given a graph $G = (V, A)$ and source and destination vertices $s, t \in V$, the algorithm for computing the shortest time-dependent cost path p^* works in three phases.

(1) A bidirectional A^* search occurs on G, where the forward search is run on the graph weighted by c with the path cost defined by (1)-(2), and the backward search is run on the graph weighted by the lower bounding function λ. All nodes settled by the backward search are included in a set M. Phase 1 terminates as soon as the two search scopes meet.

(2) Suppose that $v \in V$ is the first vertex in the intersection of the heaps of the forward and backward search; then the time dependent cost $\mu = \gamma_{\tau_0}(p_v)$ of the path p_v going from s to t passing through v is an upper bound to $\gamma_{\tau_0}(p^*)$. In the second phase, both search scopes are allowed to proceed until the backward search queue only contains nodes whose associated key exceeds μ, with the additional constraint that the backward search cannot explore nodes already settled by the forward search. In other words: let β be the key of the minimum element of the backward search queue; phase 2 terminates as soon as $\beta > \mu$.

Again, all nodes settled by the backward search are included in M.

(3) Only the forward search continues, with the additional constraint that only nodes in M can be explored. The forward search terminates when t is settled.

We have the following theorems.

Theorem 1 *The algorithm in Sect. 3 computes the shortest time-dependent path from s to t for a given departure time τ_0.*

Theorem 2 *Let p^* be the shortest path from s to t. If the condition to switch to phase 3 is $\mu < K\beta$ for a fixed parameter K, then the algorithm in Sect. 3 computes a path p from s to t such that $\gamma_{\tau_0}(p) \leq K\gamma_{\tau_0}(p^*)$ for a given departure time τ_0.*

4. Experiments

of our time-dependent ALT algorithm. Our implementation is written in C++ using solely the STL. As priority queue we use a binary heap. Our tests were executed on one core of an AMD Opteron 2218 running SUSE Linux 10.1. The machine is clocked at 2.6 GHz, has 16 GB of RAM and 2 x 1 MB of L2 cache. The program was compiled with GCC 4.1, using optimization level 3. We use 16 maxcover landmarks [2], computed on the input graph using the lower bounding function λ to weight edges. When performing random s-t queries, the source s, target t, and the starting time τ_0 are picked uniformly at random and results are based on 10 000 queries. We tested our algorithm on the road network of Western Europe provided by PTV AG for scientific use, which has approximately 18 million vertices and 42.6 million arcs. A travelling time in uncongested traffic situation was assigned to each arc using that arc's category (13 categories) to determine the travel speed. We generated time-dependent costs using a random generator based on statistical real-world data.

Table 4 reports the results of our bidirectional ALT variant on time-dependent networks for different approximation values K using the European road network as input. For comparison, we also report the results on the same road network for the time-dependent versions of Dijkstra and unidirectional ALT. As the performed ALT-queries compute approximated results instead of optimal solutions, we record three different statistics to characterize the solution quality: error rate, average relative error, maximum relative error. By *error rate* we denote the percentage of computed suboptimal paths over the total number of queries. By *relative error* on a particular query we denote the relative percentage increase of the approximated solution over the optimum, computed as $\omega/\omega^* - 1$, where ω is the cost of the approximated solution computed by our algorithm and ω^* is the cost of the optimum computed by Dijkstra's algorithm. We report *average* and *maximum* values of this quantity over the set of all queries. We also report the number of nodes settled at the *end* of each phase of our algorithm, denoting them with the labels phase 1, phase 2 and phase 3.

As expected, we observe a clear trade-off between the quality of the computed so-

			ERROR			QUERY			
				relative		# settled nodes			time
input	method	K	rate	avg	max	phase 1	phase 2	phase 3	[ms]
EUR	Dijkstra	-	0.0%	0.000%	0.00%	-	-	8 908 300	6 325.8
	uni-ALT	-	0.0%	0.000%	0.00%	-	-	2 192 010	1 775.8
	ALT	1.00	0.0%	0.000%	0.00%	125 068	2 784 540	3 117 160	3 399.3
		1.05	4.0%	0.029%	4.93%	125 068	1 333 220	1 671 630	1 703.6
		1.10	18.7%	0.203%	8.10%	125 068	549 916	719 769	665.1
		1.13	30.5%	0.366%	12.63%	125 068	340 787	447 681	385.5
		1.15	36.4%	0.467%	13.00%	125 068	265 328	348 325	287.3
		1.20	44.7%	0.652%	18.19%	125 068	183 899	241 241	185.3
		1.50	48.8%	0.844%	25.70%	125 068	130 144	172 157	121.9
		2.00	48.9%	0.886%	48.86%	125 068	125 071	165 650	115.7

lution and query performance. If we are willing to accept an approximation factor of $K = 2.0$, queries are on average 55 times faster than Dijkstra's algorithm, but almost 50% of the computed paths will be suboptimal and, although the average relative error is still small, in the worst case the approximated solution has a cost which is 50% larger than the optimal value. The reason for this poor solution quality is that, for such high approximation values, phase 2 is very short. As a consequence, nodes in the middle of the shortest path are not explored by our approach, and the meeting point of the two search scopes is far from being the optimal one. However, by decreasing the value of the approximation constant K we are able to obtain solutions that are very close to the optimum, and performance is significantly better than for unidirectional ALT or Dijkstra. In our experiments, it seems as if the best trade-off between quality and performance is achieved with an approximation value of $K = 1.15$, which yields average query times smaller than 300 ms with a maximum recorded relative error of 13%. By decreasing K to values < 1.05 it does not pay off to use the bidirectional variant any more, as the unidirectional variant of ALT is faster and is always correct.

References

[1] D. Delling and D. Wagner. Landmark-based routing in dynamic graphs. In C. Demetrescu, editor, *WEA 2007*, volume 4525 of *LNCS*, New York, 2007. Springer.

[2] A. Goldberg and C. Harrelson. Computing the shortest path: A^* meets graph theory. In *SODA 2005*. SIAM, 2005.

[3] A. Goldberg, H. Kaplan, and R. Werneck. Reach for A^*: Efficient point-to-point shortest path algorithms. In C. Demetrescu, R. Sedgewick, and R. Tamassia, editors, *ALENEX 2005*. SIAM, 2005.

A Multi-start Heuristic Algorithm for the Generalized Traveling Salesman Problem

Valentina Cacchiani [a] Albert Einstein Fernandez Muritiba [a]
Marcos Negreiros [b] Paolo Toth [a,*]

[a] *DEIS, University of Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy*

[b] *Departamento de Estatística e Computação, Universidade Estadual do Ceará , Av. Paranjana, 1700 Campus do Itaperi, Fortaleza, CE, Brasil*

Key words: Generalized Traveling Salesman, Heuristic Algorithms, Decomposition Approach, Improvement Procedure

1. Abstract

We study the Generalized Traveling Salesman Problem (*GTSP*), which is a variant of the well-known Traveling Salesman Problem (*TSP*). We are given a graph $G = (V, E)$, where V is the set of nodes, and E the set of edges, each with an associated cost. The set of nodes V is partitioned into m clusters $V_1, \ldots, V_m$ with $V_1 \cup \ldots \cup V_m = V$ and $V_i \cap V_j = \emptyset$ if $i \neq j$. *GTSP* is to find an elementary cycle visiting at least one node for each cluster, and minimizing the sum of the costs of the traveled edges. We focus on the so-called *Equality GTSP* (*E-GTSP*), in which the cycle has to visit exactly one node for each cluster.

The *GTSP* is a generalization of the *TSP*: we obtain the traditional *TSP* in the particular case where all the clusters are composed by just one node. Thus the *GTSP* is NP-Hard.

Most of the literature focuses on heuristic approaches for solving the problem, due to its high complexity. However, in [1] Fischetti et al. present a Branch and Cut algorithm to solve the problem to optimality. The best state-of-art heuristic algorithms for the *GTSP* are the following:

1. the Generalized Initialization, Insertion and Improvement algorithm by Renaud and Boctor ([4]);
2. the Nearest Neighbor approach by Noon ([2]);
3. the Reinforcing Ant Colony System by Pintea et al. ([3]);

* Corresponding author.

4. the heuristic algorithms by Fischetti et al. ([1]), which are computed at the root node of the branch-decision tree.

We propose a multi-start heuristic algorithm, which iteratively starts with a randomly chosen set of nodes and applies a decomposition approach to the problem combined with improvement procedures. In a decomposition algorithm, the problem is subdivided into two subproblems. According to the classification introduced by Renaud and Boctor ([4]), there are two ways of decomposing the problem. One possibility is to select the nodes to be visited, and then construct a cycle that visits the selected nodes; another possibility is to determine the order in which the clusters are visited, and then construct the optimal cycle.

Our method combines these two approaches, alternating the two ways of decomposing the problem and introducing also some randomness in order to explore a greater solution space.

In particular, our approach considers a first phase to determine the visiting order of the clusters and a second phase to find the minimum cost cycle. The visiting order of the clusters is obtained as follows: we randomly choose, with uniform probability, one node in each cluster and then compute a TSP feasible solution by using the Farthest Insertion approach, followed by a 2-opt improvement procedure. Once the order of the clusters is fixed, the second phase starts: the Bellman-Ford algorithm is applied. It computes, in polynomial time, the shortest cycle which visits exactly one node in each cluster. This phase gives a new set of nodes, which can be different from the one obtained at the end of the first phase. Thus, we apply a 2-opt improvement procedure to the new sequence, allowing a change in the order of the clusters. If the order of the clusters is changed, we apply again the Bellman-Ford algorithm in an iterative way. Otherwise the current solution cannot be further improved, so we start again with a set of nodes, randomly chosen with uniform probability, one from each cluster. However, we also apply a probabilistic step: with probability p each node in the chosen set is substituted by the node corresponding to the same cluster in the best solution found so far. Our approach iteratively repeats these steps until a stop condition is reached (e.g., time limit).

The algorithm was tested on a set of benchmark instances obtained by a clustering procedure introduced by Fischetti et al. ([1]) applied to instances from the TSPLIB library. These instances are generally used to test the efficiency of the algorithms for the *GTSP*.

The results obtained by our approach are compared with the optimal solutions obtained by a Branch-and-Cut algorithm presented in [1]. Moreover, we compare our results with those obtained by the best state-of-art heuristic algorithms. The results show that our approach is competitive with the other heuristic algorithms, finding the optimal solution for the 83.8% of the instances in less then 10 seconds, and for the 100% of the instances in short computing times.

References

[1] M. Fischetti, J.J. Salazar-Gonzalez and P. Toth, *A branch-and-cut algorithm for the symmetric generalized traveling salesman problem*, Operations Research 45, 378–394, 1997.

[2] C. E. Noon, *The generalized traveling salesman problem*, Ph.D. Dissertation, University of Michigan, 1988.

[3] C.M. Pintea, P.C. Pop and C. Chira *The Generalized Traveling Salesman Problem Solved with Ant Algorithms*, Journal of Universal Computer Science 13, 1065–1075, 2007.

[4] J. Renaud and F.F. Boctor, *An efficient composite heuristic for the symmetric generalized traveling salesman problem*, European Journal of Operational Research 108, 571–584, 1998.

The Uncapacitated Swapping Problem

Shoshana Anily [a] Aharona Pfeffer [a,*]

[a] *Faculty of Management Tel-Aviv University*

Key words: swapping problem, transshipment problems

1. The Swapping Problem

The Swapping Problem (SP) was first introduced by Anily and Hassin (1992) on general graphs. In the problem that they defined, each vertex of a given graph may initially contain an object of a known type, and it may be associated with a desired object of a different type. The initial and final arrangements are assumed to be balanced, namely that the total number of objects of each type is equal in both arrangements. A single vehicle of unit capacity is available for shipping objects among the vertices. The SP is to compute a shortest route such that the vehicle can accomplish the rearrangement of the objects while following this route.

The original SP has many variations. For example, we can define the problem on different graph structures, such as a line, a circle, a tree, and a general graph; and also consider various capacities of the vehicle: one unit, some finite capacity k, or infinite capacity, i.e., the *uncapacitated SP*.

In this research we focus on the uncapacitated SP on a line and on a circle, and we present polynomial-time exact algorithms for each. For the linear track we present an $O(nlogn)$-time exact algorithm, and for the circular track, an $O(n^2logn)$-time exact algorithm, where n is the number of vertices on the graph.

Tables 1-3 summarize the known complexity results for the SP, for the unit capacity, finite capacity, and infinite capacity cases. The tables also present the results for the known *Stacker Crane Problem* and the *Dial-a-Ride Problem*. These two problems can be seen as special cases of the SP, where there is only one unit of each object type, thus each product has a specific destination. With respect to the unpacacitated

* Corresponding author.

Table 1
Results for the Unit Capacity Case ($k = 1$). [from Anily et.al. (2006)]

	Swapping Problem			**Stacker Crane Problem**		
Graph Structure	Preemptive	Non-Preemptive	Mixed	Preemptive	Non-Preemptive	Mixed
Line	Polynomial[3]	Polynomial[3]	Polynomial[3]	Polynomial	Polynomial	Polynomial
Circle		-[16]		Polynomial[6]	Polynomial[6]	Polynomial[16]
Tree	NP-hard[4]	NP-hard[9,10]	NP-hard	Polynomial	NP-hard	NP-hard
General	NP-hard[2]	NP-hard[2]	NP-hard[2]	NP-hard	NP-hard[11]	NP-hard

Table 2
Results for the Finite Capacity Case ($k > 1$)

	Swapping Problem			**Dial-a-Ride Problem**		
Graph Structure	Preemptive	Non-Preemptive	Mixed	Preemptive	Non-Preemptive	Mixed
Line		NP-hard	NP-hard		NP-hard[16]	NP-hard
Circle		NP-hard	NP-hard		NP-hard	NP-hard
Tree	NP-hard	NP-hard	NP-hard	NP-hard[8]	NP-hard[8]	NP-hard
General	NP-hard	NP-hard	NP-hard	NP-hard[4]	NP-hard[8]	NP-hard

Table 3
Results for the Uncapacitated SP

Graph Structure	**Swapping Problem**	**Dial-a-Ride Problem**
Line	Polynomial[15]	Polynomial
Circle	Polynomial*	Polynomial
Tree		
General	NP-hard	NP-hard[14,17]

* This work.

case (Table 3) it can be seen that the SP on a general graph is NP-hard, as it generalizes the Traveling Salesman Problem. The complexity of the uncapacitated SP on a tree is still an open problem.

1.1 The linear track

For the linear track case we develop an algorithm, which allows to specify general initial and terminal positions for the vehicle. Next we list the three key elements of the algorithm:

- We use the *minimally balanced partition* defined by Anily et.al. (1999) to determine the line-segments which the vehicle traverses loaded.
- We prove that every optimal tour must contain one of two specific basic routes.
- We find the minimal total length to be added to the basic route in order to make it a feasible solution.

We also present a simplified algorithm, of complexity $O(n)$, for the case that the initial and terminal positions of the vehicle are at the endpoints.

In the circular track case we first prove that in the optimal solution the vehicle either covers all the circumference, or it leaves one interval uncovered. For the second case, for each interval the problem reduces to the SP on a line. For the first case, we perform a transformation of the given circle to a set of linear tracks (with more than n stations each). On these linear tracks we can solve the problem using the SP on a line.

References

[1] S. Anily and J. Bramel (1999), “Approximation for the Capacitated Traveling Salesman Problem with Pickups and Deliveries". *Naval Research Logistics* 46: 654-670.

[2] S. Anily and R. Hassin (1992), “The Swapping Problem". *Networks* 22: 419-433.

[3] S. Anily, M. Gendreau and G. Laporte (1999), “The Swapping Problem on Line". *SIAM Journal on Computing* 29: 327-335.

[4] S. Anily, M. Gendreau and G. Laporte (2006), “The Preemptive Swapping Problem on a Tree". Working Paper, Recanati Graduate School of Business, Tel-Aviv University.

[5] S. Anily and G. Mosheiov (1994), “The Traveling Salesman Problem with Delivery and Backhauls". *Operations Research Letters* 16: 11-18.

[6] M.J. Attalah and S.R. Kosaraju (1988), “Efficient Solutions to Some Transportation Problems with Applications to Minimizing Robot Arm Travel". *SIAM Journal on Computing* 17: 849-869.

[7] P. Chalasani and R. Motwani (1999), “Approximating Capacitated Routing and Delivery Problems". *SIAM Journal on Computing* 28: 2133-2149.

[8] M. Charikar and B. Raghavachari (1998), “The Finite Capacity Dial-a-Ride Problem". *Proceedings of the 39th Annual IEEE Symposium on the Foundations of Computer Science* 458-467.

[9] G.N. Frederickson and D.J. Guan (1992), “Preemptive Ensemble Motion Planning on a Tree". *SIAM Journal on Computing* 21: 1130-1152.

[10] G.N. Frederickson and D.J. Guan (1993), “Nonpreemptive Ensemble Motion Planning on a Tree". *Journal of Algorithms* 15: 29-60.

[11] G.N. Frederickson, M.S. Hecht and C.E. Kim (1978), “Approximation Algorithms for some Routing Problems". *SIAM Journal on Computing* 7: 178-193.

[12] D.J. Guan (1998), “Routing a Vehicle of Capacity Greater Than One". *Discrete Applied Mathematics* 81: 45-57.

[13] S.O. Krumke, J. Rambau and S. Weider (2000), “An Approximation Algorithm for the Non-Preemptive Capacitated Dial-a-Ride Problem". Konrad-Zuse-Zentrum-fur-Informationstechnik-Berlin, ZIB-Report 00-53.

[14] M. Kubo and H. Kasugai (1990), Heuristic Algorithms for the Single Vehicle Dial-a-Ride Problem. *Journal of the Operations Research Society of Japan* 33; 354-365.

[15] A. Pfeffer (2004), Uncapacitated Swapping Problems on a Line. M.Sc. Thesis (in Hebrew), Recanati Graduate School of Business, Tel-Aviv University.

[16] A. Pfeffer (2007), “The Swapping Problems on Graphs". Research proposal, Recanati Graduate School of Business, Tel-Aviv University.

[17] H. Psaraftis (1983), Analysis of an $O(n^2)$ Heuristic for the Single Vehicle Many-to-Many Euclidean Dial-a-Ride Problem. *Transportation Research* 17B: 133-145.

Graph theory (III)

Room A, Session 6

Thursday 15, 9:00-10:30

Edge fault-diameter of Cartesian graph bundles

Iztok Banič [a,c], Rija Erveš [b,*], Janez Žerovnik [a,c]

[a] *FME, University of Maribor, Smetanova 17, Maribor 2000, Slovenia.*

[b] *FCE, University of Maribor, Smetanova 17, Maribor 2000, Slovenia.*

[c] *Institute of Mathematics, Physics and Mechanics, Jadranska 19, Ljubljana 1000, Slovenia.*

Key words: Cartesian graph bundles, Cartesian graph products, edge-connectivity, edge fault-diameter, interconnection network

1. Introduction

In the design of large interconnection networks several factors have to be taken into account. A usual constraint is that each processor can be connected to a limited number of other processors and the delays in communication must not be too long. Extensively studied network topologies in this context include graph products and bundles. 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 [6, 9] and multiplicative circulant graphs [15]. Other graph products, sometimes under different names, have been studied as interesting communication network topologies [5, 13, 15].

Furthermore, an interconnection network should be fault-tolerant. Since nodes or links of a network do not always work, if some nodes or links are faulty, some information may not be transmitted by some of these nodes, links. Therefore the (edge) fault-diameter has been determined for many important networks recently [7, 8, 12, 16]. The concept of fault-diameter of Cartesian product graphs was first described in [11], but the upper bound was wrong, as shown by Xu, Xu and Hou who corrected the mistake [16]. An upper bound for the fault-diameter of Cartesian graph products and bundles was given in [2, 3]. Also an upper bound for the edge

* Corresponding author.

fault-diameter of Cartesian graph products was given in [4]. Graph bundles were first studied in [14].

We generalize the result of [4] for two factors to Cartesian graph bundles. As a k-edge connected graph remains connected if up to $k-1$ edges are missing, we study the diameter of a graph with any permitted number of edges deleted. We show that the edge-connectivity of Cartesian graph bundle G with fibre F over the base graph B, is at least $k_F + k_B$, and we give an upper bound for the edge fault-diameter of Cartesian graph bundles in terms of edge fault-diameters of the fibre and the base graph. We also show that the bounds are tight. For details, see [1].

2. Preliminaries

Definition 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 $F \square K_2$.*

More precisely, the mapping $p : G \to B$ maps graph elements of G to graph elements of B, i.e. $p : V(G) \cup E(G) \to V(B) \cup E(B)$. In particular, here we also assume that the vertices of G are mapped to vertices of B and the edges of G are mapped either to vertices or to edges of B. We say an edge $e \in E(G)$ is *degenerate* if $p(e)$ is a vertex. Otherwise we call it *nondegenerate*. The mapping p will also be called the *projection* (of the bundle G to its base B). Note that each edge $e = uv \in E(B)$ naturally induces an isomorphism $\varphi_e : p^{-1}(\{u\}) \to p^{-1}(\{v\})$ between two fibres. It may be interesting to note that while it is well-known that a graph can have only one representation as a product (up to isomorphism and up to the order of factors) [10], there may be many different graph bundle representations of the same graph [17]. Here we assume that the bundle representation is given.

Example 1 *It is less known that graph bundles also appear as computer topologies. A well known example is the twisted torus on Figure 1. Cartesian graph bundle with fibre C_4 over base C_4 is the ILIAC IV architecture.*

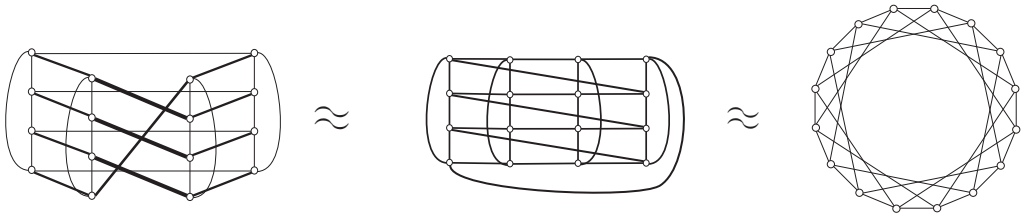


Fig. 1. Twisted torus: Cartesian graph bundle with fibre C_4 over base C_4.

Definition 2 *The* edge-connectivity *of a graph G, $\lambda(G)$, is the minimum cardinality over all edge-separating sets in G. A graph G is said to be k-*edge connected*, if $\lambda(G) \geq k$.*

Definition 3 *Let G be a k-edge connected graph and $0 \leq a < k$. Then we define*

the a-edge fault-diameter *of* G *as*

$$\bar{\mathcal{D}}_a(G) = \max\{\mathrm{d}(G \setminus X) \mid X \subseteq E(G), |X| = a\}.$$

Note that $\bar{\mathcal{D}}_a(G)$ is the largest diameter among subgraphs of G with a edges deleted, hence $\bar{\mathcal{D}}_0(G)$ is just the diameter of G, $\mathcal{D}(G)$. For $a \geq k$, the edge fault-diameter of k-edge connected graph does not exist. In other words, $\bar{\mathcal{D}}_a(G) = \infty$ as some of the graphs are not edge-connected.

3. Edge Fault-diameter of Cartesian graph bundles

Theorem 1 *Let* F *and* B *be* k_F*-edge connected and* k_B*-edge connected graphs respectively, and* G *a Cartesian graph bundle with fibre* F *over the base graph* B*. Let* $\lambda(G)$ *be the edge-connectivity of* G*. Then* $\lambda(G) \geq k_F + k_B$*.*

Theorem 2 *Let* F *and* B *be* k_F*-edge connected and* k_B*-edge connected graphs respectively,* $0 \leq a < k_F$*,* $0 \leq b < k_B$*, and* G *a Cartesian bundle with fibre* F *over the base graph* B*. Then*

$$\bar{\mathcal{D}}_{a+b+1}(G) \leq \bar{\mathcal{D}}_a(F) + \bar{\mathcal{D}}_b(B) + 1.$$

Next example shows that the bound in Theorem 2 is tight.

Example 2 *Let* $G = P_2 \square P_2$*.* G *is a graph bundle with fiber* $F = P_2$ *over the base graph* $B = P_2$*. Then for* $a = b = 0$ *we have*

$$\bar{\mathcal{D}}_{a+b+1}(G) = 3,$$

$$\bar{\mathcal{D}}_b(B) + \bar{\mathcal{D}}_a(F) + 1 = 1 + 1 + 1 = 3.$$

References

[1] I. Banič, R. Erveš, J. Žerovnik, Edge fault-diameter of Cartesian graph bundles, submitted.

[2] I. Banič, J. Žerovnik, Fault-diameter of Cartesian graph bundles, Inform. Process. Lett. 100 (2006) 47–51.

[3] I. Banič, J. Žerovnik, Fault-diameter of Cartesian product of graphs, Adv. in Appl. Math. 40 (2008) 98–106.

[4] I. Banič, J. Žerovnik, Edge fault-diameter of Cartesian product of graphs, Lecture Notes in Comput. Sci. 4474 (2007) 234-245.

[5] J.-C. Bermond, F. Comellas, D. Hsu, Distributed loop computer networks: a survey, J. Parallel Distrib. Comput. 24 (1995) 2–10.

[6] P. Cull, S. M. Larson, On generalized twisted cubes, Inform. Process. Lett. 55 (1995) 53–55.

[7] D. Z. Du, D. F. Hsu, Y. D. Lyuu, On the diameter vulnerability of kautz digraphs, Discrete Math. 151 (2000) 81–85.

[8] K. Day, A. Al-Ayyoub, Minimal fault diameter for highly resilient product networks, IEEE Trans. Parallel Distrib. Syst. 11 (2000) 926–930.

[9] K. Efe, A variation on the hypercube with lower diameter, IEEE Trans. Comput. 40 (1991) 1312–1316.

[10] S. Klavžar, W. Imrich, *Products Graphs, Structure and Recognition*, Wiley, New York, 2000.

[11] M. Krishnamoorthy, B. Krishnamurty, Fault diameter of interconnection networks, Comput. Math. Appl. 13 (1987) 577–582.

[12] 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.

[13] X. Munoz, Asymptotically optimal (δ, d', s)-digraphs, Ars Combin. 49 (1998) 97–111.

[14] T. Pisanski, J. Shawe-Taylor, J. Vrabec, Edge-colorability of graph bundles, J. Comb. Theory Ser. B 35 (1983) 12–19.

[15] I. Stojmenović, Multiplicative circulant networks: Topological properties and communication algorithms, Discrete Appl. Math. 77 (1997) 281–305.

[16] M. Xu, J.-M. Xu, X.-M. Hou, Fault diameter of Cartesian product graphs, Inform. Process. Lett. 93 (2005) 245–248.

[17] B. Zmazek, J. Žerovnik, Algorithm for recognizing Cartesian graph bundles, Discrete Appl. Math. 120 (2002) 275–302.

Decomposing trees with large diameter

Romain Ravaux *

Laboratoire PRISM, Université de Versailles-Saint Quentin en Yvelines, 45 Av des Etats Unis, 78035 Versailles, France

Key words: decomposition, trees, diameter

1. Introduction

We call partition of the integer n a sequence $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_p)$ such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$ and $\sum_{i=1}^{p} \lambda_i = n$. The spectrum of λ is defined by $sp(\lambda) = \{\lambda_1, \lambda_2, \ldots, \lambda_p\}$. Consider a n-vertex graph $G = (V, E)$ and let $\lambda = (\lambda_1, \ldots, \lambda_p)$ be a partition of n. A decomposition of G for λ, called a λ-decomposition, is a partition $\{V_1, \ldots, V_p\}$ of V such that for all $1 \leq i \leq p$, we have $|V_i| = \lambda_i$, and the subgraph of G induced by any subset V_i is connected. Such a partition $V_1, \ldots, V_p$ of V is called a (G, λ)-partition. The graph G is said decomposable if and only if for all partition λ of n the graph G is decomposable for λ.

Respectively in 1976 and 1977, Györi [1] and Lovász [2] have shown that any n-vertex k-connected graph G is decomposable for all partitions $\lambda = (\lambda_1, \ldots, \lambda_k)$ of n which contain k integers. However their proofs do not yield any polynomial-time algorithm. Differents results have been done for particulars cases. For a state of the art see [3]. In [4] it has been shown a polynomial algorithm for deciding if a tripode (three chains linked to one vertex of degree 3) is decomposable.

In this paper we focus on trees with a large diameter. We note $D(T)$ the diameter of the tree T. We show that for all partition $\lambda = (\lambda_1, \ldots, \lambda_p)$ of n with $|sp(\lambda)| \geq \alpha$, any n-vertex tree T with diameter $D(T) = n - \alpha$ is λ-decomposable. This structural result provides an algorithm to decide if a n-vertex tree T with diameter $D(T) = n - \alpha$ is decomposable with time complexity $n^{O(\alpha)}$.

2. Structural results

Proposition 1 below is based on Lemma 1 whose proof is not given in this abstract.

* Corresponding author.

Lemma 1 *Let I be a set of natural integers such that $I \subseteq \{1, \ldots, n-1\}$ and $|I| = \alpha - 1$. For all partition $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_p)$ of n such that $|sp(\lambda)| \geq \alpha$, there exists a permutation $\pi_1, \ldots, \pi_p$ of λ such that for all $1 \leq i \leq p$ we have $\sum_{j=1}^{i} \pi_j \notin I$.*

Proposition 1 *Consider a n-vertex tree $T = (V, E)$ with diameter $D(T) = n - \alpha$. The tree T is decomposable for all partitions $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_p)$ of n with $|sp(\lambda)| \geq \alpha$.*

Proof. We first compute a chain of $n - \alpha + 1$ vertices $\langle x_0, x_1, \ldots, x_{n-\alpha} \rangle$ with time complexity $n^{O(1)}$. The graph $F = (V_F, E_F)$ such that $V_F = V$ and $E_F = E - \{[x_0, x_1], [x_1, x_2], \ldots, [x_{n-\alpha-1}, x_{n-\alpha}]\}$ is a forest composed of $n - \alpha + 1$ trees each one containing one vertex of the chain $\langle x_0, x_1, \ldots, x_{n-\alpha} \rangle$. For all $0 \leq i \leq n - \alpha$, we note A_i the set of vertices of the tree containing the vertex x_i. We intend to show that for all partition $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_p)$ of n with $|sp(\lambda)| \geq \alpha$, there exists a ($T$,$\lambda$)-partition $V_1, V_2, \ldots, V_p$ of $V(G)$ with the following property : for all $0 \leq i \leq n - \alpha$, there exists $j \in \{1, \ldots, p\}$ such that $A_i \subseteq V_j$. Then, for all $0 \leq i \leq n - \alpha$ the vertices of the tree induced by A_i are included in one of the sets of the (T,λ)-partition $V_1, V_2, \ldots, V_p$. Such a (T,λ)-partition is called a (T,λ)-clean partition (see Figure 1).

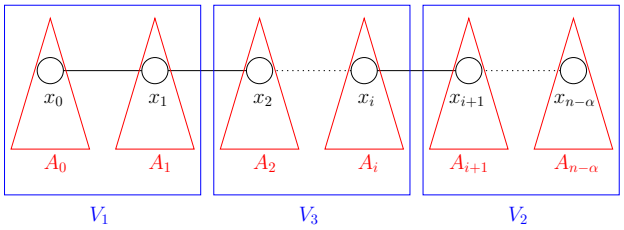


Fig. 1. A tree T with diameter $D(T) = n - \alpha$ and a (T,p)-clean partition V_1, V_2, V_3

Consider the set of natural integers $P = \bigcup_{i=0}^{n-\alpha} \{\sum_{j=0}^{i} |A_j|\}$ and the set of natural integers $I = \{1, \ldots, n\} \setminus P$. We call P the set of *possible integers* and I the set of *forbidden integers*. By definition, P and I give a partition of $\{1, \ldots, n\}$, with $n \in P$, $|P| = n - \alpha + 1$ and $|I| = \alpha - 1$. By Lemma 1, there exists a permutation $\pi_1, \ldots, \pi_p$ of λ such that for all $1 \leq i \leq p$ we have $\sum_{j=1}^{i} \pi_j \in P$. This permutation yields a (T,λ)-clean partition of V.□

Notice that the Proposition 1 is tight. Indeed, consider an integer α. Let $n = \sum_{i=1}^{\alpha-1} 2i$. Consider the partition $\lambda = (2, 4, 6, \ldots, 2\alpha - 2)$ of n. We have $|sp(\lambda)| = \alpha - 1$. The n-vertex tree $T = (V, E)$ with diameter $D(T) = n - \alpha$ in Figure 2 is not decomposable for the partition λ, since for all $i \in \{1, \ldots, \alpha - 1\}$, if the vertex $x_0 \in V_i$, of size $2i$, then one of the vertices $\{f_1, \ldots, f_i\}$ will be isolated and there is no part of size 1 in λ.

3. Algorithmic results

By Proposition 1 any tree $T = (V, E)$ with diameter $D(T) = n - \alpha$ is decomposable for all partition λ with $|sp(\lambda)| \geq \alpha$. To decide if T is decomposable, one must

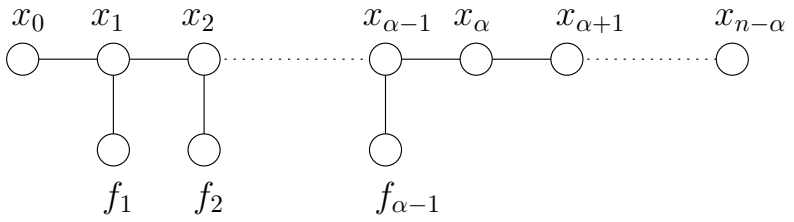


Fig. 2. A tree T with diameter $n - \alpha$

check wether it is decomposable by all other partitions. In this section, we present an algorithm doing so in time $n^{O(\alpha)}$. First we consider the case of a single partition.
Proposition 2 *Consider a n-vertex tree $T = (V, E)$ with diameter $D(T) = n - \alpha$. Let $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_p)$ be a partition of n . Deciding if the tree T is decomposable for the partition λ can be done with a time complexity $n^{O(\alpha)}$.*

Proof. If $|sp(\lambda)| \geq \alpha$, then by Proposition 1, we know that T is decomposable for λ. Suppose now that $|sp(\lambda)| \leq \alpha-1$. We note $u_1, u_2, \ldots, u_{\alpha-1}$ the $\alpha-1$ vertices of T that do not belong to the chain $\langle x_0, x_1, \ldots, x_{n-\alpha}\rangle$. We intend to generate all the (T,λ)-partitions. We start with the p empty sets $V_1, \ldots, V_p$ of the (T,λ)-partition. The first step of the process consists in generating all the possibles sets V_i which will contain at least one of the vertices $u_1, \ldots, u_{\alpha-1}$. First we construct a set containing the vertex u_1. We note V_{i_1} this set. The size of the set V_{i_1} can be equal to at most $\alpha-1$ values. For each value, we note c this size of V_{i_1}. We choose the other vertices of $\{u_2, \ldots, u_{\alpha-1}\}$ which will belong to V_{i_1}. There exists $2^{\alpha-1}$ possible subsets of $\{u_2, \ldots, u_{\alpha-1}\}$. For each subset S of $\{u_2, \ldots, u_{\alpha-1}\}$, we add the vertices of S in V_{i_1}. Then $V_{i_1} = \{u_1\} \cup S$. Consider $g = \min\{i \in \{0, \ldots, n-\alpha\} : A_i \cap V_{i_1} \neq \emptyset\}$. Let $d = \max\{i \in \{0, \ldots, n-\alpha\} : A_i \cap V_{i_1} \neq \emptyset\}$. Adding the vertices $x_g, x_{g+1}, \ldots, x_d$ in the subset V_{i_1} is a necessary condition (but not sufficient) for the subgraph of T induced by V_{i_1} to be connected. A this step, we must verify if the sub-graph of T induced by V_{i_1} is connected. This can be done in time $O(n)$. If it is the case and if we have $|V_{i_1}| \leq c$ then we add $|V_{i_1}| - c$ vertices in V_{i_1} from the chains $\langle x_0, \ldots, x_g\rangle$ and $\langle x_d, \ldots, x_{n-\alpha}\rangle$, by preserving the connexity. There are at most $O(n)$ possibilities. The number of such possible sets containing the vertice u_1 is thus $O(\alpha 2^{\alpha-1} n)$.

Given such a set containing the vertice u_1, we remove the vertices choosen (set V_{i_1}) from the tree T. We obtain a forest F. From the forest F, we search all the possibles sets V_i containing at least one of the remaining vertices u_i. Thus the number of possible sets V_i containing at least one of the vertices $u_1, \ldots, u_{\alpha-1}$ is $O(\alpha^\alpha 2^{\alpha^2} n^\alpha)$.

At this stage, We have to generate all the possible sets V_i containing none of the vertices $u_1, \ldots, u_{\alpha-1}$. Let p' the remaining partition. The partition p' contains the integers of p which have not been used during the first step of the process. Let S be the set of vertices which belong to the sets V_i containing at least one of the vertices $u_1, \ldots, u_{\alpha-1}$. Let R be the subgraph of T induced by $V - S$. The subgraph R is a forest with at most α connected components, each one being a chain. Thus deciding if R is decomposable for the partition p' is equivalent to solving a α-subset sum problem, with $\alpha \leq n$ and n coded in unary. By dynamic programing we can decide if R is decomposable for the partition p' in time $n^{O(\alpha)}$. Thus we can decide

if T is decomposable for the partition λ in time $n^{O(\alpha)}$.□

Theorem 1 *Consider a n-vertex tree $T = (V, E)$ with diameter $D(T) = n - \alpha$. Deciding if the tree T is decomposable can be done with time complexity $n^{O(\alpha)}$.*

Proof. By Proposition 1, we know that the tree T is decomposable for all partition λ of n with $|sp(\lambda)| \geq \alpha$. Thus it remains to study the partitions λ of n with $|sp(\lambda)| \leq \alpha - 1$. Consider an integer n and an integer α. The number of partitions λ of n with $|sp(\lambda)| \leq \alpha - 1$ is $O(\alpha n^{2\alpha})$ and we can generate them with a time complexity $O(\alpha n^{2\alpha+1})$. By Proposition 2 deciding if the tree T is decomposable can be done with time complexity $n^{O(\alpha)}$. □

References

[1] E. GYÖRI. On division of graphs to connected subgraphs. *In Colloquia Mathematica Societatis János Bolyai,* 1976

[2] L. LOVÁSZ. A homology theory for spanning trees of a graph. *Acta Mathematica Academiae Scientiarum Hungaricae,* 1977

[3] D. BARTH and H. FOURNIER. A degree bound on decomposable trees. *Discrete mathematics,* 2006

[4] D. BARTH and O. BAUDON and J. PUECH. Decomposable trees: a polynomial algorithm for tripodes. *Discrete applied mathematics,* 2002

On the Bi-enhancement of Chordal-bipartite Probe Graphs

Elad Cohen [a] Martin Charles Golumbic [a] Marina Lipshteyn [a,*]
Michal Stern [a,b]

[a] *Caesarea Rothschild Institute, and Department of Computer Science, University of Haifa, Haifa, Israel*

[b] *Academic College of Tel-Aviv-Jaffa, Israel*

Abstract

Lately, a lot of research has been done on $\mathscr{C}$-probe graphs. In this paper we focus on chordal-bipartite probe graphs. We prove a structural result that if B is a bipartite graph with no chordless cycle of length strictly greater than 6, then B is chordal-bipartite probe if and only if a certain "enhanced" graph B^* is a chordal-bipartite graph. This theorem is analogous to a result on interval probe graphs in [13] and to one on chordal probe graphs in [8].

Key words: Probe, bipartite, chordal.

1. Introduction

Let $G = (V, E)$ be a finite, undirected, simple graph. We say that vertex u "sees" vertex v if $uv \in E$. A set $X \subseteq V$ is a *stable* set in G if for all $u, v \in X, (u, v) \notin E$, i.e., no vertex in X sees another vertex in X. A graph G is a *bipartite* graph if its vertices can be partitioned into two disjoint stable sets $V = X \cup Y$. We will refer to this as the "black/white" bipartition of the vertices.

A sequence $(v_1, \ldots, v_n)$ of distinct vertices is a *path* in G if $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq n-1$. A closed path $(v_1, \ldots, v_n, v_1)$ is called a *cycle* if in addition $(v_n, v_1) \in E$. A *chord* of a cycle $(v_1, \ldots, v_n, v_1)$ is an edge between two vertices of the cycle that is not an edge of the cycle. A *chordless* cycle C_n is a cycle which contains no chords and has n vertices and n edges.

A graph G is a *chordal graph* if it contains no induced chordless cycle C_n, for $n \geq 4$. A graph G is a *chordal-bipartite* graph if it is a bipartite graph and it contains

* Corresponding author.

no induced chordless cycle C_n, for $n \geq 6$. (Note that just as a chordal graph may contain triangles, a chordal-bipartite graph may contain chordless 4-cycles.)

Let $\mathscr{C}$ be a graph class. A graph G is called a *$\mathscr{C}$-probe* graph if its vertex set can be partitioned into two subsets, P (probes) and N (non-probes), where N is a stable set, and one can add to G a set of edges between pairs of non-probes such that the resulting graph is in $\mathscr{C}$. More formally, G is $\mathscr{C}$-probe if there exists a completion $E' \subseteq \{(u,v) \mid u,v \in N,\ u \neq v\}$ such that $G' = (V, E \cup E')$ is in $\mathscr{C}$. Recognizing whether or not G is a *$\mathscr{C}$-probe* graph is known as the *non-partitioned* probe problem.

In the *partitioned* version of the probe problem, the partition of V into probes and non-probes is given and fixed. The *partitioned $\mathscr{C}$*-probe problem is a special case of the $\mathscr{C}$-sandwich problem [6].

Interval probe graphs were first introduced by Zhang [13] and studied further in [7, 10–12]. Chordal probe graphs were investigated in [8] and a characterization and recognition algorithm was given in [2] for both the partitioned and non-partitioned versions. Other probe classes are to be found in [3].

In this paper, we focus on the partitioned version of chordal-bipartite probe graphs. In particular, we prove a structural result that if B is a bipartite graph with no chordless cycle of length strictly greater than 6, then B is chordal-bipartite probe if and only if a certain "enhanced" graph B^* is a chordal-bipartite graph. This theorem is analogous to a result on interval probe graphs in [13] and to one on chordal probe graphs in [8]. We believe it may also shed light on the more general case. We conclude the paper with open questions.

2. Motivation: chordal probe graphs

Let $G = (V, E)$ be a graph whose vertices have been partitioned into a set P of probes and a stable set N of non-probes. The following was proved in [8]:
Lemma 1 *If G is a chordal probe graph with respect to the partition $P \cup N$, then probes and non-probes must alternate on every chordless cycle.*

In the specific case of a chordless 4-cycle with edges ab, bc, cd, da, this means that either $\{a, c\}$ are probes and $\{b, d\}$ are non-probes, or vice versa. Moreover, suppose that $\{a, c\}$ are probes, then any possible chordal probe completion of G would be forced to contain the addition of an edge bd, which Zhang [13] called an *enhanced edge*. The *enhanced graph* $G^* = (P \cup N, E^*)$ is defined to be the graph obtained from G by adding all enhanced edges from all chordless 4-cycles.
Theorem 1 *Let G be a graph containing no induced chordless cycles C_k for $k > 4$. If G has a probe/non-probe partition in which probes and non-probes alternate on every chordless 4-cycle, then the enhanced graph G^* is a chordal completion of G.*

Lemma 1 together with Theorem 1 prove the next corollary.

Corollary 1 *If G is a C_k-free graphs for $k > 4$, then G is chordal probe if and only if G^* is chordal.*

This also gave an alternate proof of a result of Zhang [13].
Corollary 2 *If G is an interval probe graph, then G^* is chordal.*

3. Chordal-bipartite probe graphs

We now turn our attention to the probe problem for chordal-bipartite graphs. Our goal will be to obtain a result in the same spirit as Cor. 1 in the case of a bipartite graph that has no chordless cycles of size strictly greater than 6.

Let $B = (P, N, E)$ be a bipartite graph whose vertex set partitioned into P (probes) and N (non-probes), where N is always assumed to be a stable set. Note that in the case of a bipartite graph B, the "black/white" bipartition of the vertices and the "probe/non-probe" bipartition of the vertices are generally different! Moreover, if B is connected, then the completion edges between non-probes will have one endpoint white and the other black, in order to maintain the bipartite property.

We begin by stating some necessary properties due to Berry, et al. [1]. The reader may wish to reconstruct a proof.
Lemma 2 *If B is a chordal-bipartite probe graph with respect to the partition $P \cup N$, then on every chordless cycle of length ≥ 6 in B the following must hold: (1) every probe sees at most one other probe, (2) there is at least one edge of the cycle whose endpoints are probes.*

Lemma 2 implies, in particular, that (1) on a chordless cycle there is no consecutive triple of probes, and that (2) there must be at least two pairs of consecutive probes, due to the parity of a bipartite graph. Moreover,
Remark 1 *In the specific case of a chordless 6-cycle C_6, there are exactly 2 non-probes, one white and one black, opposite each other, and any possible chordal-bipartite probe completion of B must contain the added edge joining them.*

We will call this forced edge a *bi-enhanced edge*. Thus, with respect to a given probe/non-probe partition, we define the *bi-enhanced graph* B^* to be the graph obtained from B by adding all bi-enhanced edges from all chordless 6-cycles.

We are now ready to state our main result.
Theorem 2 *Let B be a bipartite graph that contains no chordless cycle C_k for $k > 6$. If B has a probe/non-probe partition in which probes and non-probes satisfy the property in Remark 1 on every chordless 6-cycle, then B^* is a chordal-bipartite completion of B.*

Similar to the case of chordal probe graphs, Lemma 2 together with Theorem 2 immediately prove the next corollary.

Corollary 3 *If B is a C_k-free bipartite graph for $k > 6$, then B is chordal-bipartite probe if and only if B^* is chordal-bipartite.*

An overview of the proof of the Theorem 2 can be found in the appendix.

4. Open questions

In Section 3, we prove that the properties of Lemma 2 characterize chordal-bipartite probe graphs in the case where the given graph contains no chordless cycles of size greater than 6. Do these properties characterize chordal-bipartite probe graphs in the general case, or are there further conditions needed?

On the algorithmic side, given a probe/non-probe partition of a bipartite graph B, how do we most efficiently find the bi-enhanced edges, that is, build B^*? What is the complexity of recognition of chordal-bipartite probe graphs, even in the case where there are no chordless cycles of size greater than 6?

References

[1] A. Berry, E. Cohen, M.C. Golumbic, M. Lipshteyn, N. Pinet, A. Sigayret and M. Stern, France-Israel Binational Science Collaboration Project, unpublished technical report, 2007.

[2] A. Berry, M.C. Golumbic and M. Lipshteyn, Cycle-bicolorable graphs and triangulating chordal probe graphs, *SIAM Journal on Discrete Mathematics* 21:573-591, 2007.

[3] M.-S. Chang, T. Kloks, D. Kratsch, J. Liu and S.-L. Peng, On the recognition of probe graphs of some self-complementary classes of perfect graphs, LNCS, 3595:808-817, 2005.

[4] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, *Academic Press*, New York, 1980. Second edition: *Annals of Discrete Math.* 57, Elsevier, Amsterdam, 2004.

[5] M.C. Golumbic and C.F. Goss, Perfect elimination and chordal-bipartite graphs, *Journal of Graph Theory*, 2:155-163, 1978.

[6] M.C. Golumbic, H. Kaplan and R. Shamir, Graph sandwich problems, *Journal of Algorithms*, 19:449-473, 1995.

[7] M.C. Golumbic and M. Lipshteyn, On the hierarchy of interval, probe and tolerance graphs, Proc. 32th Southeastern Conf. on Combinatorics, Graph Theory and Computing, *Congressus Numerantium* 153, Utilitas Math., Winnipeg, Canada, 2001, pp. 97-106.

[8] M.C. Golumbic and M. Lipshteyn, Chordal probe graphs, *Discrete Applied Mathematics*, 143:221-237, 2004.

[9] M.C. Golumbic and A.N. Trenk, *Tolerance Graphs*, Cambridge University Press, 2004.

[10] J.L. Johnson and J.P. Spinrad, A polynomial time recognition algorithm for probe interval graphs, *Proc. SODA'01*,(12th Annual ACM-SIAM Symposium on Discrete Algorithms) 477-486, 2001.

[11] R.M. McConnell and J.P. Spinrad, Construction of probe interval models, *Proceedings of SODA'02*,(13th Annual ACM-SIAM Symposium on Discrete Algorithms) 866-875, 2002.

[12] F.R. McMorris, C. Wang and P. Zhang, On probe interval graphs, *Discrete Applied Mathematics*, 88:315-324, 1998.

[13] P. Zhang, Probe interval graphs and their application to physical mapping of DNA, *Manuscript, 1994.*

Column generation

Room B, Session 6

Thursday 15, 9:00-10:30

Column Generation Algorithms for the Capacitated m-Ring-Star Problem [1]

Edna A. Hoshino [a] Cid C. de Souza [b,*]

[a] *University of Mato Grosso do Sul, Department of Computing and Statistics, Campo Grande MS, Brazil*

[b] *University of Campinas, Institute of Computing, Campinas SP, Brazil*

Key words: ring star problem, integer programming, column generation

---

In the capacitated m-ring-star problem (CmRSP) a set of customers has to be visited by m vehicles initially located at a central depot. Each vehicle performs a route or *ring* that starts and ends at the depot and is characterized by an ordered set of customers and *connection points*. These connection points are selected among a set of predefined sites called the *Steiner* points. Besides, there is also a *star* associated to each vehicle. The *star* of vehicle t is a set of pairs of the form (u, v) where u is a customer and v is a customer or *Steiner* point belonging to the *ring* of t. In the latter situation, we say that u is *connected* to v. Customers in the *ring-star*(i.e., *ring* or *star*) of t are said to be covered by t and their quantity is limited by the capacity of the vehicle which is assumed to be the same for the entire fleet. Now, a solution for the CmRSP can be viewed as a set of m *ring-star*s covering all customers. Routing costs incur for every pair of consecutive sites in a *ring*, while connection costs incur for every connection defined by a *star*. The cost of a solution is then given by the sum of all routing costs plus all the connection costs induced by its m *ring-star*s. The CmRSP asks for a solution with minimum cost and can be easily shown to be $\mathcal{NP}$-hard since it generalizes the *Traveling Salesman Problem* (TSP).

The CmRSP was introduced by Baldacci et al. [1] who describe an application in the design of large fiber optics networks. The authors proposed a *branch-and-cut* (BC) algorithm for the problem and reported experiments where instances of moderate size were solved in reasonable time. To the best of our knowledge, this is the only exact algorithm available for the CmRSP. On the heuristic side, Mauttone et al. [2] proposed an algorithm combining GRASP (Greedy Randomized Adaptive Search Procedure) and Tabu Search which obtained good solutions for

* Corresponding author.

[1] First author supported by a scholarship from Capes/PICDT (Brazilian Ministry of Education). Second and corresponding author supported by grants 301732/2007-8, 478470/2006-1, 472504/2007-0 and 473726/2007-6 from *Conselho Nacional de Desenvolvimento Científico e Tecnológico* and grant 03/09925-5 from *Fundação de Amparo à Pesquisa do Estado de São Paulo*.

the same instances tested in [1]. The literature also contains results for the *Ring-Star Problem* (RSP) and its variations [3, 4]. The RSP can be viewed as restricted case of the CmRSP where a single uncapacitated vehicle is available. On the other hand, the *Vehicle Routing-Allocation Problem* (VRAP) presented by Beasley and Nascimento in [5] is a generalization of the CmRSP where customers may remain unattended, though this situation is penalized in the objective function. Special attention has been paid to instances of the VRAP where only one vehicle is at hand, the so-called *Single Vehicle Routing-Allocation Problem* [6].

In this work we propose an integer programming formulation for the CmRSP based on a *set covering* model and develop a *branch-and-price* (BP) algorithm to solve it exactly. We decided to investigate the adequacy of column generation to the CmRSP, encouraged by the success of this approach for the classical Capacitated Vehicle Problem (CVRP), that can be interpreted as a special case of the CmRSP. In particular, the CmRSP is suited to cope with CVRP applications where customers can be served indirectly by displacing themselves to a site covered by one of the m routes. The best results reported in the literature concerning the exact solution of the CVRP were obtained by a robust *branch-and-cut-price* (BCP) algorithm proposed in [7]. BCP algorithms embed cutting planes and column generation in a standard branch-and-bound procedure for solving Integer Programming (IP) problems. One of the key ingredients of the BCP algorithm described in [7] refers to the relaxation to the pricing (column generation) problem. The pricing problem arising in the set covering model for the CmRSP asks for a single capacitated ring-star with minimum reduced cost. Through polynomial reductions involving a variant of TSP with Profits [8], called Profitable Tour Problem [9], one can easily show that this pricing problem is $\mathcal{NP}$-hard. Thus, different ways to relax the pricing problem to a more tractable one are investigated here.

Our approach to relax the pricing problem is similar to that of using q-routes for the CVRP [7]. Basically, a q-route is a relaxation of a route that admits repetition of vertices. In a similar fashion, we adopt a relaxed pricing problem that searches for a *relaxed ring-star*, i.e., a *ring-star* where vertices are allowed to be repeated in the *ring* and/or in the *star*. Unfortunately, the dual bound obtained by linear relaxation becomes weak with this relaxed pricing. It can be improved by avoiding the occurrence of k*-cycles*, cycles with length less or equal to k, inside the *rings*. The prohibition of k*-cycles* in paths and its use in the relaxation of discrete optimization problems is not a novelty. Examples where this idea was applied can be found in [10] for the TSP and in [11] for the *Resource Constrained Shortest Path Problem*. The elimination of k*-cycles* in q-routes was used to solve the *Vehicle Routing Problem* in [12]. The success of this idea relies on the fact that, for small values of k, the elimination of k-cycles can be done without changing the complexity of the algorithm that computes the paths or q-routes. To eliminate k-cycles from *relaxed ring-stars*, we used the *label setting algorithm* (LSA) [13] following the idea of Irnich and Villeneuve [11] to solve the *Non-elementary Shortest Path Problem with Resource Constraints and k-cycle Elimination* [11, 14]. By prohibiting k-cycles we avoid some vertex repetitions in the *ring*, but not in the *star*. To get rid of some repetitions in the *ring-star* we can represent all vertices of the *ring-star* in a string

in some predefined order, and forbid the occurrence of k-cycles in the string. As this structure is not actually a k-cycle, we call it a k-stream. We observed that the prohibition of k-streams was important not only to augment dual bounds, but also to improve the overall performance of the *branch-and-price* algorithm. When we forbid 3-streams instead of 3-cycles, the number of instances solved at optimality raised of almost 100%. Moreover, the running time was reduced, on average, in 40%. This was made possible through a clever implementation of the *label setting algorithm* to identify *useless* paths. *Useless* paths are *ring-stars* that are dominated with respect to cost and can be discarded during the execution of the LSA. Differently from the routine presented in [11], our implementation is based on a deterministic finite automaton, which reduced in 60% the time spent to solve the pricing problem.

Conclusions. We proposed a *set covering* IP model for CmRSP and *branch-and-price* algorithms to solve this model. The key point in developing such algorithms is how to solve the pricing problem, which is $\mathcal{NP}$-hard. To obtain a faster code, we relaxed the pricing problem in different ways and, through experimentation, we investigated which of these variants of the algorithm leads to a better performance. We end up with a *branch-and-price* code for the CmRSP, called BP3sA, which we proved to be competitive with a *branch-and-cut* algorithm proposed earlier. This was achieved through a careful implementation of the algorithm which solves the relaxed pricing problem via a deterministic finite automaton. We also noticed that the BP3sA and BC do not dominate each other and some instances are better suited for one or the other algorithm. This suggests that the next step in this research should be the development of a *branch-and-cut-and-price* algorithm combining the two techniques.

References

[1] Baldacci, R., Dell'Amico, M., Salazar, J.: The capacitated m-ring star problem. Operations Research **55** (2007) 1147–1162

[2] Mauttone, A., Nesmachnow, S., Olivera, A., Robledo, F.: A hybrid metaheuristic algorithm to solve the capacitated m-ring star problem. In: International Network Optimization Conference. (2007)

[3] Labbé, M., Laporte, G., Martín, I.R., González, J.S.: The ring-star problem: Polyhedral analysis and exact algorithm. Networks **43** (2004) 117–189

[4] Dias, T., de Souza Filho, G., Macambira, E., Cabral, L., Fampa, M.: An efficient heuristic for the ring star problem. In: Experimental Algorithms WEA 2006. Volume 4007 of Lecture Notes in Computer Science., Springer (2006) 24– 35

[5] Beasley, J., Nascimento, E.: The vehicle routing-allocation problem: A unifying framework. TOP: An Official Journal of the Spanish Society of Statistics and Operations Research **4**(1) (June 1996) 65–86

[6] Vogt, L., Poojari, C., Beasley, J.E.: A tabu search algorithm for the single vehicle routing allocation problem. Journal of the Operational Research Society (2007) 467–480

[7] Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M.P., Reis, M., Uchoa, E., Werneck, R.: Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical Programming **106**(3) (July 2006) 491–511

[8] Feillet, D., Dejax, P., Gendreau, M.: Traveling salesman problems with profits. Transportation Science **39**(2) (2005) 188–205

[9] Dell'Amico, M., Maffioli, F., Varbrand, P.: On prize-collecting tours and the asymmetric travelling salesman problem. International Transactions in Operational Research **2**(3) (1995) 297–308

[10] Houck, D., Picard, J., Queyranne, M., Vemuganti, R.: The travelling salesman problem as a constrained shortest path problem: theory and computational experience. Opsearch **17** (1980) 93–109

[11] Irnich, S., Villeneuve, D.: The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. Informs Journal on Computing **18**(3) (2006) 391–406

[12] Christofides, N., Mingozzi, A., Toth, P.: Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. Mathematical Programming **20** (1981) 255–282

[13] Ahuja, R., Magnanti, T., Orlin, J.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall (1993)

[14] Irnich, S., Desaulniers, G.: Shortest path problems with resource constraints. In: Column Generation. Springer (2005) 33–65

Column Generation for the Minimum Hyperplane Clustering Problem

Edoardo Amaldi [a] Alberto Ceselli [b] Kanika Dhyani [a,*]

[a] *Dipartimento di Elettronica e Informazione, Politecnico di Milano*

[b] *Dipartimento di Tecnologie dell'Informazione, Universita' degli Studi di Milano*

Key words: Hyperplane clustering, column generation, infeasible linear systems

1. Introduction

Clustering is a fundamental optimization problem and is extensively studied due to the wide range of applications including data mining and mathematical biology. Classical clustering methods [10] partition a given set of points in $\mathbb{R}^n$ into k clusters so as to minimize some dissimilarity measure between the data points and the "center" of the cluster they have been assigned to.

Since in a number of applications the main issue is to partition the points according to their co-planarity rather than to their distribution w.r.t. their cluster centers, there has recently been a growing interest for clustering w.r.t. subspaces [13]. In this work we address the interesting case in which all the subspaces are hyperplanes.

There are two main variants of the Hyperplane Clustering Problem (HCP) depending on whether the number k of hyperplanes is known a priori (k-HCP) or whether it has to be minimized subject to some maximum error tolerance (MIN HCP).

Although there is some related work in computer science [7], control [8] and computational geometry [2, 9, 12] on k-HCP, the only discrete optimization-based approach for the MIN HCP that we are aware of was proposed in [4] for piecewise linear model fitting with applications to line segment detection in digital images and time series modeling.

The MIN HCP is defined as follows: Given m points $\{\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_m\}$ in $\mathbb{R}^n$ and a maximum allowed tolerance $\epsilon > 0$, determine a minimum number k of hyperplanes $\mathcal{H}_j = \{\vec{a} \mid \vec{a} \in \mathbb{R}^n, \vec{a}\vec{x}_j = b_j\}, j = 1, \ldots, k$, such that each point lies within $\pm\epsilon$ deviation from the hyperplane it is assigned to, where the hyperplane parameters $\{(\vec{x}_j, \vec{b}_j)\}_{j=1,\ldots,k}$ take real values.

* Corresponding author.

If we view each data point $\vec{a}_i, 1 \leq i \leq m$, as the i-th row of a matrix $\vec{A} \in \mathbb{R}^{m \times n}$, then the hyperplane parameters must simultaneously satisfy $\vec{a}_i \vec{x} \leq b + \epsilon$ and $\vec{a}_i \vec{x} \geq b - \epsilon$, for all $1 \leq i \leq m$. When more than one hyperslab of width $\frac{\epsilon}{\|\vec{x}\|}$ is needed to cover all the points the resulting linear system containing m pairs of complementary inequalities is clearly infeasible. From now on the subset of points that can be assigned to a same hyperplane within ϵ is called an *h-cluster* if the (sub)system composed by the above complementary inequalities with the same ϵ is feasible.

Note that MIN HCP is NP-hard since it is already NP-complete to decide whether a set of points in the plane ($n = 2$, $k = 2$) can be covered exactly by k lines ($\epsilon = 0$) unless P=NP [11].

2. Column generation approach

In this work we make a first step towards an exact algorithm for solving the MIN HCP by developing a column generation approach. We propose a set covering formulation for the master problem ($\mathcal{MP}$). Given the set $\mathcal{S}$ of all possible h-clusters (or feasible subsystems), for each $s \in \mathcal{S}$ we consider a binary variable y_s such that $y_s = 1$ iff the corresponding h-cluster s appears in the solution. Then ($\mathcal{MP}$) in which we minimize the number of hyperplanes can be formulated as follows:

$$\begin{aligned} \min \ & \sum_{s \in \mathcal{S}} y_s \\ \text{s.t.} \ & \\ & \sum_{s \in \mathcal{S}} D_{is} y_s \geq 1 \quad \forall i = 1, \ldots, p \\ & y_s \in \{0, 1\} \quad \forall s \in \mathcal{S}, \end{aligned} \tag{1}$$

where $D_{is} = 1$ if subsystem (h-cluster) s contains row (point) i and $D_{is} = 0$ otherwise.

Since the number of feasible subsystems can be exponential w.r.t. the number of rows for a given set of inequalities, we work with a representative small subset $S' \subset S$. Generating an appropriate $\mathcal{S}'$ is a critical issue that will be discussed later.

By solving a linear relaxation of $\mathcal{MP}$ (with just the nonnegativity constraints on the y_s) for a given $\mathcal{S}'$ we obtain a feasible solution to the relaxed problem and a dual value w_i for each constraint in $\mathcal{MP}$. The set $\mathcal{S}'$, is expanded only if the reduced cost of a feasible subsystem s' given by:

$$c_{s'w} = 1 - \sum_{i=1}^{p} D_{is'} w_i \tag{2}$$

that is to be added to it, is negative. Negative reduced cost columns are obtained by solving the pricing subproblem $\mathcal{PP}$:

$$\min\ (1 - \sum_{i=1}^{p} w_i D_i) = 1 - \max \sum_{i=1}^{p} w_i D_i \tag{3}$$

s.t.

$$-\epsilon - M(1 - D_i) \leq A\vec{x} - b \qquad \forall i = 1, \ldots, p \tag{4}$$

$$A\vec{x} - b \leq \epsilon + M(1 - D_i) \qquad \forall i = 1, \ldots, p \tag{5}$$

$$\| \vec{x} \|_2 = 1 \tag{6}$$

$$D_i \in \{0, 1\} \qquad \forall i = 1, \ldots, p,$$

where the M is a large enough constant so that when $D_i = 0$ constraints (4)-(5) are inactive. $D_i = 1$ forces the point i to be assigned to hyperplane iff they lie within $\pm\epsilon$ deviation from it. Due to constraint (6), which is needed to avoid the trivial solution $\vec{x} = \vec{0}$, the pricing subproblem is mixed integer nonlinear program (MINLP) with a nonconvex constraint.

Note that the objective function (3) with constraints (4)-(5) amounts to a weighted version of the MAX FS [5], in which given an infeasible system we look for a feasible subsystem with the largest total weight. Since MAX FS is NP-hard to solve even approximately and we have the additional nonconvex constraint (6), the $\mathcal{PP}$ is particularly challenging and we need a heuristic that can be repeatedly applied to produce feasible subsystems (columns) on the fly.

2.1 Solving the pricing subproblems

To generate an initial set of columns $\mathcal{S}'$, we have extended the randomized thermal relaxation (RTR) for MAX FS proposed in [3], so as to deal with weighted pairs of complementary inequalities and the normalization (6). See the full paper for a description.

Since the computational load of our extended RTR can be substantial for large-sized systems, we propose and investigate alternative pricing subproblems where trivial solutions are avoided in different ways (with different norms).

l_∞-norm pricing subproblem: Given that $\| \vec{x} \|_\infty \leq \| \vec{x} \|_2$, substituting (6) with $\| \vec{x} \|_\infty = 1$ we obtain an upper bound for the optimal value of $\mathcal{PP}$. By using the l_∞ norm we get a MINLP which can be linearized by reformulating $\| \vec{x} \|_\infty = 1$ with the help of binary variables $\vec{u} \in \{0, 1\}^n$ as $\vec{x} \geq \vec{1} - 2(\vec{1} - \vec{u}), \vec{u}^T\vec{1} = 1$, where $\vec{1}$ is an n-dimensional vector of ones.

Relaxed Pricing Subproblem: Here we optimize over the region sandwiched by putting the largest sized cuboid in a unit spheroid. The resulting formulation is a MILP with the additional binary variables $\vec{u}, \vec{v} \in \{0, 1\}^n$ and the added constraints

$$\vec{x} \geq (1 + \frac{1}{\sqrt{m}})\vec{u} - \vec{1}, \qquad \vec{x} \leq (1 - \frac{1}{\sqrt{m}})\vec{v} + \vec{1}, \qquad \vec{u}^T\vec{1} + \vec{v}^T\vec{1} \geq 1.$$

3. Computational results

We implemented the column generation approach within the SCIP [1] framework, using the MINLP solver BONMIN [6] for solving the 2-norm pricing subproblem and CPLEX for the other MILP pricing subproblems.

Our algorithms were tested on realistic random instances as well as real world data. The results provided by the column generation (CG) approach with the three alternative methods for pricing were compared with those found in the literature and those obtained by a greedy procedure (where feasible subsystems are extracted iteratively till no rows are left).

For small-sized instances (up to $m = 150$), CG tends to provide better solutions than the greedy in shorter or comparable CPU time. For larger-sized instances ($m \geq 500$), we obtain better solutions but at the cost of increased CPU time as a larger number of feasible subsystems (columns) needs to be generated to achieve a small duality gap. Detailed results will be provided in the complete article.

As far as alternative pricing methods are concerned, the 2-norm $\mathcal{PP}$ requires on average a smaller computing time than the relaxed and l_∞-norm ones. We are currently experimenting speedup strategies where we alternate between different pricing subproblems. Future work includes devising branching rule to reach optimality as well as developing a heuristic to refine the solution provided by the CG in which points consistent with more than one h-cluster are possibly reassigned.

References

[1] T. Achterberg, *SCIP -a framework to integrate constraint and mixed integer programming*, 2004, [Online] http://www.zib.de/Publications/abstracts/ZR-04-19, pp. 4–19.

[2] P. K. Agarwal and C. M. Procopiuc, *Approximation algorithms for projective clustering*, Journal of Algorithms **46** (2003), 115ÂŰ–139.

[3] E. Amaldi, P. Belotti, and R. Hauser, *Randomized relaxation methods for themaximum feasible subsystem problem*, Proceedings, IPCO XI, Berlin 2005. Lecture Notes in Computer Science **3509** (2005), 249–264.

[4] E. Amaldi and M. Mattavelli, *The MIN PFS problem and piecewise linear model estimation*, Discrete Appl. Math. **118** (2002), 115–143.

[5] E. Amaldi, M. E. Pfetsch, and L. E. Trotter Jr., *On the maximum feasible subsystem problem, IISs and IIS-hypergraphs*, Math. Programming A **95** (2003), 533–554.

[6] P. Bonami and J. Lee, *BONMIN users' manual*, [Online] http://projects.coin-or.org/Bonmin.

[7] P. S. Bradley and O. L. Mangasarian, *k-plane clustering*, J. of Global Optimization **16** (2000), no. 1, 23–32.

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

[9] Sariel Har-Peled and Kasturi Varadarajan, *Projective clustering in high dimensions using core-sets*, SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry (New York, NY, USA), ACM, 2002, pp. 312–318.

[10] A.K. Jain, M. N. Murty, and P.J. Flynn, *Data cluster: A review*, ACM Computing Surveys **31** (1999), no. 3, 264–323.

[11] N. Megiddo and A. Tamir, *On the complexity of locating linear facilities in the plane*, Operations Research Letters **1** (1982), 194–197.

[12] N. Mishra, R. Motwani, and S. Vassilvitskii, *Sublinear projective clustering with outliers*, 15th Annual Fall Workshop on Computational Geometry and Visualization (2005).

[13] L. Parsons, E. Haque, and H. Liu, *Subspace clustering for high dimensional data: a review*, SIGKDD Explor. Newsl **6** (2004), no. 1, 90–105.

Two-stage Column Generation in Container Terminal Management

Ilaria Vacca [a,*] Michel Bierlaire [a] Matteo Salani [a]

[a] *Transp-OR, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland*

Key words: berth allocation, qc assignment, dantzig-wolfe decomposition

1. Introduction

In this paper, we consider the Tactical Berth Allocation Problem (TBAP) with Quay Cranes Assignment as presented in [4].
The authors take into account two decision problems arising in container terminals, which are usually solved hierarchically by the terminal planners: the Quay Crane Assignment Problem (QCAP), which assigns to incoming ships a certain *QC profile* (number of quay cranes per working shift), and the Berth Allocation Problem (BAP), which consists of assigning and scheduling incoming ships to berthing positions. QCAP and BAP are strictly correlated, because assigned QC profiles affect ship handling times, which impact on the berth allocation. These two problems are integrated in TBAP and the resulting combined problem is solved at the tactical level. For more details on container terminal operations, we refer the reader to [7], [6] and [8].
In [4] a compact formulation of the problem is presented i.e. a formulation with a polynomial number of variables and constraints. Given $n = |N|$ ships with time windows on the arrival time at the terminal, $m = |M|$ berths with time windows on availability, a planning time horizon discretized in $|H|$ time steps, a set P_i of feasible QC profiles defined for every ship $i \in N$, and the maximum number of quay cranes available in the terminal, the aim is to find a feasible assignment of ships to berths, a feasible scheduling and a feasible QC profile assignment for every ship, in order to maximize the total value of selected profiles.
In this work, we propose a Dantzig-Wolfe reformulation of the compact model presented in [4], whose linear relaxation exhibits a poor lower bound. Our reformulation results in a combination of feasible berth and QC assignments, called *berth sequences*. The resulting model has an exponential number of variables and is solved via Column Generation (cf. [3]), which yields to better dual bounds but

* Corresponding author.

requires the optimal solution of the pricing problem.
We propose a network model to reduce the pricing subproblem to an Elementary Shortest Path Problem with Resource Constrains (ESPPRC), inspired by [1]. The main difficulty of this formulation is that the number of feasible QC profiles which can be assigned to ships is huge, although polynomially bounded, and the solution of the pricing problem on the resulting network might be impractical.
The set of feasible QC profiles is proposed by practitioners; the main intuition of our approach is that only a restricted subset of QC profiles contributes to the optimal solution. Therefore, we propose a two stage column generation algorithm which, at the first stage, builds berth sequences over a subset of QC profiles and, at the second stage, adds improving QC profiles to the network model.
Reduced cost arguments adapted from [2] are used to identify promising and useless profiles. More specifically, we investigate reduced cost variable elimination, a promising technique to reduce the number of variables in any integer linear program (cf. [5]). In particular, a non-negative integer variable can be eliminated when its reduced cost, with respect to a feasible dual solution, exceeds the duality gap.

2. Extensive Formulation

We use the concept of berth sequence, which represents a sequentially ordered (sub)set of ships in a berth with an assigned QC profile.
Let Ω^k be the set of all feasible sequences r for berth $k \in M$, including the empty sequence, which means that berth k is not used.
Let z_r^k be the decision variable which is 1 if sequence $r \in \Omega^k$ is used by berth k and 0 otherwise. The extensive TBAP formulation is the following:

$$\max \quad \sum_{k \in M} \sum_{r \in \Omega^k} v_r^k z_r^k \tag{1}$$

$$s.t. \quad \sum_{k \in M} \sum_{r \in \Omega^k} y_{ir}^k z_r^k = 1 \qquad \forall i \in N, \tag{2}$$

$$\sum_{k \in M} \sum_{r \in \Omega^k} q_r^{hk} z_r^k \leq Q^h \qquad \forall h \in H, \tag{3}$$

$$\sum_{r \in \Omega^k} z_r^k = 1 \qquad \forall k \in M, \tag{4}$$

$$z_r^k \in \{0,1\} \qquad \forall r \in \Omega^k, \forall k \in M. \tag{5}$$

where: v_r^k is the value of sequence $r \in \Omega^k$; y_{ir}^k is 1 if ship i in sequence $r \in \Omega^k$ and 0 otherwise; q_r^{hk} is the number of QCs used by sequence $r \in \Omega^k$ at time step h; Q^h is the number of QCs available at time step h.
The value v_r^k of a sequence $r \in \Omega^k$ is given by the sum of the values of the profiles assigned to ships served by the sequence:

$$v_r^k = \sum_{i \in N} \sum_{p \in P_i} v_i^p \lambda_{ir}^{pk} \tag{6}$$

where v_i^p is the value of profile $p \in P_i$ and λ_{ir}^{pk} is a parameter which is 1 if profile $p \in P_i$ is in sequence $r \in \Omega^k$ and 0 otherwise.
Equations (2) are ship-cover constraints, (3) are QCs capacity constraints and (4) are convexity constraints. The objective function (1) maximizes the total value of chosen sequences.

Identical restrictions on subsets. We now assume that feasible subsets Ω^k have identical requirements (i.e. berths have identical availability's time windows): consequently, $\Omega^k = \Omega \ \forall k$.
Let z_r be the decision variable which is 1 if sequence $r \in \Omega$ is chosen and 0 otherwise. The extensive formulation is now defined as follows:

$$\max \quad \sum_{r\in\Omega} v_r z_r \tag{7}$$

$$s.t. \quad \sum_{r\in\Omega} y_{ir} z_r = 1 \qquad \forall i \in N, \tag{8}$$

$$\sum_{r\in\Omega} q_r^h z_r \leq Q^h \qquad \forall h \in H, \tag{9}$$

$$\sum_{r\in\Omega} z_r = m, \tag{10}$$

$$z_r \in \{0,1\} \qquad \forall r \in \Omega. \tag{11}$$

Parameters and constraints are the same as in (1)-(5), except for the index k which has disappeared.
Note that the integer linear program defined by (7)-(11) chooses only sequences, without assignment to berths. However, as berths are assumed to be identical, this assignment can be done post-optimization, arbitrarily, without loss of generality.

3. Two-stage Column Generation

The linear relaxation of (7)-(11), called Master Problem (MP), has a huge number of columns (variables), as it is defined on the space of all feasible sequences Ω. We define the Restricted Master Problem (RMP) on a subset $\Omega' \subset \Omega$ of columns and we solve it by column generation.
Let $[\pi, \mu, \pi_0]$ be an optimal dual solution to an RMP, where $\pi \in \mathbb{R}^n$ is the dual vector associated to ship-cover constraints (8), $\mu \geq 0$ is the dual vector associated to capacity constraints (9) and $\pi_0 \in \mathbb{R}$ is the dual variable associated to the aggregated convexity constraint (10).
The reduced cost of a sequence r is:

$$\tilde{v}_r = v_r - \sum_{i\in N} \pi_i y_{ir} - \sum_{h\in H} \mu_h q_r^h - \pi_0 \tag{12}$$

where π_i represents the dual price of serving ship i in sequence r and μ_h represents the dual price of using an additional quay crane at time step h.

The *pricing problem* (or *subproblem*):

$$\max_{r\in\Omega\setminus\Omega'}\{\tilde{v}_r\} = \max_{r\in\Omega\setminus\Omega'}\{v_r - \sum_{i\in N}\pi_i y_{ir} - \sum_{h\in H}\mu_h q_r^h\} - \pi_0 \tag{13}$$

identifies a column r^* with maximum reduced cost. If $\tilde{v}_{r^*} > 0$, we have identified a new column to enter the basis; if $\tilde{v}_{r^*} \leq 0$, we have proven that the current solution of RMP is also optimal for MP.

In the pricing problem, the decisions which have to be made are: (i) whether ship i is in the sequence or not (represented by variable y_{ir}); (ii) whether profile p is used by ship i or not (represented by variable λ_{ir}^p which is implicitly involved in the pricing problem through $v_r = \sum_{i\in N}\sum_{p\in P_i} v_i^p \lambda_{ir}^p$); (iii) the order of ships in the sequence (implicitly represented by variable q_r^h).
By defining a network $G(\bar{N}, A)$, which has one node for every ship $i \in N$, for every profile $p \in P_i$ and for every time step $h \in H$, and whose arcs have transit time equals to the length of the profile, we can reduce the pricing problem to an Elementary Shortest Path Problem with Resource Constraints (ESPPRC). The size of this network grows polynomially with the number of ships, working shifts and QC profiles; however, in the worst case (unbounded time windows) the network is complete and has $\sim |\bar{N}|^2$ arcs. Consequently, since ESPPRC is a NP-Hard combinatorial problem, its solution on such a big network is impractical.
We therefore propose to build the network by considering a smaller subset $P_i' \subset P_i$ of QC profiles for every $i \in N$, and to solve the MP over this restricted subset. We remark that this operation prevent us to find a valid dual bound at the end of the column generation algorithm. In order to fix this, we add an artificial QC profile for each ship i, which has the highest profile value $\tilde{v}_p$, the shortest duration $\tilde{t}_p$ and the smallest QC utilization among all feasible QC profiles for ship i in $P_i \setminus P_i'$. The solution of the pricing with this additional profile is now super-optimal and thus we are able to compute a valid dual bound (UB) for the MP.
Thanks to this bound and to a feasible primal solution (LB), we can compute a duality gap $g = UB - LB$. We can now eliminate all variables λ_{ir}^p with a reduced cost strictly smaller than $-g$. Since not all λ variables have already been generated (indeed none of the $\lambda \in P_i \setminus P_i'$ is), we adapt the method recently proposed by [2] to compute a valid bound on their reduced cost. Remarkably, any QC profile in P_i could be eliminated from the pricing subproblem with this approach. When a profile in P_i' is eliminated, all the corresponding profiles are eliminated from the master as well. Among the remaining QC profiles, we select the subset of profiles with strictly positive reduced cost and we iterate the entire process. In this sense the column generation process has two stages: firstly, berth sequences are built considering a subset of QC profiles and, subsequently, promising QC profiles are added to the model.

References

[1] Desaulniers G., Desrosiers J., Ioachim I., Solomon M.M., Soumis F. & Villeneuve D. (1998) *A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems*, Fleet Management and Logistics, Kluwer Academic Publishers, 57-93.

[2] Irnich S., Desaulniers G., Desrosiers J. & Hadjar A. (2007) *Path Reduced Costs for Eliminating Arcs*, HEC Montréal, Les Cahiers du GERAD, G-2007-79.

[3] Lübbecke M.E. & Desrosiers J. (2005) *Selected Topics in Column Generation*, Operations Research **53**, 1007-1023.

[4] Moccia L., Giallombardo G., Salani M. & Vacca I. (2008) *The Tactical Berth Allocation Problem with Quay Cranes Assignment and Transshipment-related Quadratic Yard Costs*, Working paper.

[5] Nemhauser G.L. & Wolsey L.A. (1988) *Integer Programming and Combinatorial Optimization*, Wiley, New York.

[6] Stahlbock R. & Voss S. (2007) *Operations research at container terminals: a literature update*, OR Spectrum **30**, 1-52.

[7] Steenken D., Voss S. & Stahlbock R. (2004) *Container terminal operation and operations research - a classification and literature review*, OR Spectrum **26**, 3-49.

[8] Vacca I., Bierlaire M. & Salani M. (2007) *Optimization at Container Terminals: Status, Trends and Perspectives*, Technical report TRANSP-OR 071204, Transport and Mobility Laboratory, EPFL.

Mathematical programming

Room A, Session 7

Thursday 15, 11:00-12:30

A Compact Representation for Chordal Graphs

L. Markenzon [a,1,*] and P. R. C. Pereira [b]

[a] *Núcleo de Computação Eletrônica*
Universidade Federal do Rio de Janeiro, RJ, Brazil
[b] *Instituto Militar de Engenharia, RJ, Brazil*

Key words: Chordal graphs, Representation, Clique-trees

1. Introduction

In this paper, we address the problem of developing simple and efficient algorithms for problems on chordal graphs. Many applications deal with very large graphs; in this case an adequate representation affects not only the performance of the algorithm but also makes the implementation easier.

We present a compact representation for chordal graphs. Besides saving computer storage significantly, the representation provides additional information of the structural properties of the graph that leads to more efficient algorithms for solving problems for the family.

2. Representing Chordal Graphs

A graph G is said to be *chordal* when every cycle of length 4 or more has a chord (an edge joining two non-consecutive vertices of the cycle). Basic concepts and properties of chordal graphs can be found in Golumbic [3] and Blair and Peyton [1]. All graphs are supposed to be connected.

In [1], Blair and Payton stated that a total ordering of the maximal cliques of the graph, say $Q_1, Q_2, \ldots, Q_\ell$, has the *running intersection property* (*RIP*) if for each clique Q_j, $2 \leq j \leq \ell$, there exists a clique Q_i, $1 \leq i \leq j-1$, such that $Q_j \cap (Q_1 \cup Q_2 \cup \ldots \cup Q_{j-1}) \subset Q_i$.

For any *RIP* ordering of the maximal cliques, a tree T_{rip} can be constructed making each clique Q_j adjacent to a *parent* clique Q_i identified by the expression above. Observe that any *RIP* ordering numbers each parent before any of its children.

* Corresponding author.
[1] Supported by grants 306893/2006-1 and 473603/2007-1, CNPq, Brazil.

Moreover, the set of trees T_{rip} equals the set of clique-trees of G. A clique-tree can be a representation of a chordal graph.

These results allow us to propose a new representation for chordal graphs. Let $G = (V, E)$ be a chordal graph, and $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_\ell\}$ the set of maximal cliques of G, with a *RIP* ordering. The *compact representation* of G is the sequence of pairs

$$\mathcal{C}R(G) = [(P_j, S_j)], \ell \geq j \geq 1,$$

such that $S_1 = \emptyset$, $S_j = Q_j \cap (Q_1 \cup Q_2 \cup \ldots \cup Q_{j-1})$ and $P_j = Q_j - S_j$.

The most important advantage of the compact representation when compared with the clique-tree representation is that several structural properties of the graph can be deduced from it. Given a chordal graph G represented by $\mathcal{C}R(G) = [(P_\ell, S_\ell), \ldots, (P_2, S_2), (P_1, S_1)]$, it can be proved that:

- The ℓ maximal cliques of G are $Q_1 = P_1 \cup S_1$, ..., $Q_\ell = P_\ell \cup S_\ell$.
- The sequence $[P_\ell, P_{\ell-1}, \ldots, P_1]$ is a perfect elimination ordering (*peo*) of G.
- S_2, S_3, ..., S_ℓ are the minimal vertex separators of G.
- The set $V' = V - (S_2 \cup S_3 \cup \ldots \cup S_\ell)$ is the set of simplicial vertices of G.
- There is an unique clique-tree $T = (V_T, E_T)$ of G such that the edges are the pairs (Q_j, Q_i), for $\ell \geq j \geq 2$, and i is obtained by

$$i = \max\{t \mid S_j \cap P_t \neq \emptyset\}.$$

It is interesting to observe that a chordal graph can have several compact representations. As an example, let G be a chordal graph with $V = \{a, b, c, d, e, f, g, h\}$ and $E = \{(a, b), (a, c), (a, d), (a, e), (a, f), (b, c), (b, d), (b, e), (b, f), (c, d), (e, f), (f, g), (f, h), (g, h)\}$. Two correct compact representations of G are

$\mathcal{C}R_1(G) = [(\{g, h\}, \{f\}), (\{e, f\}, \{a, b\}), (\{a, b, c, d\}, \emptyset)]$ and

$\mathcal{C}R_2(G) = [(\{c, d\}, \{a, b\}), (\{a, b, e\}, \{f\}), (\{f, g, h\}, \emptyset)]$.

Another important advantage is the easy implementation of queries. We can consider, for instance, the *adjacency query*, that tests whether two vertices u and v are adjacent. If u and v belong to the same P_i then the vertices are adjacent. Otherwise, let $u \in P_i$ and $v \in P_j$, being $i > j$. The vertices are adjacent if and only if $v \in S_i$.

3. Generating and Analyzing the Compact Representation

Based on the properties stated in Section 2, it is easy to develop a simple and efficient algorithm to obtain the compact representation.

Blair and Peyton [1] show that a maximal cardinality search (*MCS*) detect the maximal cliques of the graph searching them one after the other; we can prove the same to a *lex-BFS*, with similar arguments. Based on this theoretical framework, we present, in [5], a simple algorithm that determines the maximal cliques, the

minimal vertex separators and the clique-tree. In order to determine the minimal vertex separators of a chordal graph, it is sufficient to establish the intersection of each pair of maximal cliques adjacent in the clique-tree. The determination of the maximal cliques of the graph can be accomplished exploring the *peo* of the graph. The algorithm presented in [3] analyzes the vertices as they appear in the *peo*. However, the same result can be obtained performing an inductive construction of the graph; in this way we are able to build, at the same time, the clique-tree and, consequently, to determine the minimal vertex separators of G. As this algorithm provide every partial result needed to establish the representation, it is possible to modify it to obtain, also in linear time, $\mathcal{C}R(G)$. Other algorithms that determine the minimal vertex separators can be found in the literature [2, 4].

We can evaluate the storage needs of the representation. Being $\sigma = \sum_{i=2,\ell} |S_i|$, the representation takes $n + \sigma$ memory positions, against $n + 2m$ of the traditional representation by adjacency sets. Notice that, since a minimal vertex separator is always a monotone adjacency set, $\sigma \leq m$. Even when the edges are not represented twice, the compact representation is better.

Since σ is not directly related to the number of edges of the graph, it is interesting to establish the actual benefit of the compact representation. It is not difficult to see that good cases happen when there are few maximal cliques in the graph and, for each Q_i, $|S_i| < |P_i|$ – this corresponds to a smaller σ. The best case is the complete graph, since its traditional representation with $n + n(n-1)$ positions is reduced to n positions. Bad cases happen when the chordal graph has a large number of maximal cliques, for instance a k-tree.

However, it is possible to establish a more precise evaluation. In order to do that, we must study the spent and saved memory positions for each maximal clique, on each representation. This study is presented in the extended paper, along with examples of the behavior of the compact representation for some known problems of chordal graphs.

References

[1] Blair, J. R. S., Peyton, B., An Introduction to Chordal Graphs and Clique Trees, In Graph Theory and Sparse Matrix Computation, IMA 56, pp. 1–29, 1993.

[2] Chandran, L.S., Grandoni, F., A Linear Time Algorithm to List the Minimal Separators of Chordal Graphs, *Discrete Mathematics*, 306, pp. 351–358, 2006.

[3] Golumbic, M.C., *Algorithmic Graph Theory and Perfect Graphs*, 2^{nd} edition, Academic Press, New York, 2004.

[4] Kumar, P.S., Madhavan, C.E.V., Minimal Vertex Separators of Chordal Graphs, *Discrete Applied Mathematics*, 89, pp. 155–168, 1998.

[5] Markenzon, L., Pereira, P.R.C., Algorithms for Chordal Graphs: the Determination of Minimal Vertex Separators and the Recognition of Planarity, Tech. Rep. RT 134 / SE-8, Instituto Militar de Engenharia, 2007.

Constrained Decompositions of Integer Matrices and their Applications to Intensity Modulated Radiation Therapy [2,3]

Céline Engelbeen [a,*,1] and Samuel Fiorini [a]

[a] *Department of Mathematics, Université Libre de Bruxelles, CP 216, Boulevard du Triomphe, 1050 Brussels, Belgium.*

Key words: Decomposition of integer matrices, consecutive ones property, multileaf collimator, radiation therapy.

When a cancer is diagnosed, physicians can prescribe radiation therapy sessions whose the aim is to destroy the tumor(s) by exposing it to radiations while preserving the healthy organs and tissues located in the radiation field. Nowadays, to deliver radiations, most hospitals use a multileaf collimator. The arm of the collimator can fully circle around the patient and stop at certain angles. These angles give several possible radiation angles in such a way that it is possible to place the tumor at the epicenter of the radiations and the organs located in the radiation field are different for each angle (so they receive a smaller dose than the tumor(s)).

To establish a radiation therapy plan three steps are needed:

(1) Determining the different radiation angles.
(2) Computing an intensity function for each angle.
(3) Modulating the radiation to obtain the required intensities for each angle.

Every function computed in step (2) is encoded as an integer matrix I of size $M \times N$ whose entries correspond to elementary parts of the radiation beam (called *bixels*); the value of each entry i_{mn} of the matrix gives the required dose for the corresponding bixel.

Here we assume that the two first steps are completed. So, we receive an integer

* Corresponding author.

[1] Céline Engelbeen is a research fellow of the "Fonds pour la Formation à la Recherche dans l'Industrie et dans l'Agriculture" (FRIA).

[2] Extended abstract. A full version has been submitted for publication in Networks and is available at `http://homepages.ulb.ac.be/~sfiorini/papers/decomp_IMRT.pdf`

[3] This research is supported by an "Actions de Recherche Concertées" (ARC) projet of the "Communauté française de Belgique".

matrix I for each angle, and we have to modulate the radiation since the component of the multileaf collimator that sends radiation is a linear accelerator which can only send a uniform radiation. Actually we have to decompose the matrix I as a linear combination of binary matrices where the ones mean that radiations do get through and the zeroes that radiations do not get through. The zeroes in the binary matrices used in the decomposition are generated by the metallic leaves of the collimator which can block the radiations and are placed between the radiation source and the patient. The machine has a left leaf and a right leaf for each row of the matrices we consider. Thus each row of any binary matrix used in the decomposition consists of a certain number of consecutive zeroes (corresponding to the bixels blocked by the left leaf), then a certain number of consecutive ones and finally a certain number of consecutive zeroes (corresponding to the bixels blocked by the right leaf). Hence, the matrices used in the decomposition of I have to satisfy the consecutive ones property.

Formally, the problem of modulating the radiation can be stated as follows: Given a matrix I of size $M \times N$ with non-negative integer entries, we seek a decomposition of I as a weighted sum of binary matrices having the consecutive ones property. Moreover, the coefficients of the decomposition are restricted to be non-negative integers. This is due to the fact that the linear accelerator can not send a smaller dose than 1 cGray, which corresponds to have a one in the decomposition. Our objective is to minimize the total sum of the coefficients. This correponds to minimizing the total *beam-on time*, which is highly desirable for medical reasons. We call this problem the *minimum beam-on time* problem (BOT).

Efficient methods to solve this problem were found by many authors [1–3, 5]. We now briefly describe one solution approach. We first define the difference matrix $\Delta = (\delta_{mn})$ of size $M \times (N+1)$ by $\delta_{mn} := i_{mn} - i_{m,n-1}$, where $i_{m0} = i_{m,N+1} := 0$. If $\delta_{mn} > 0$ we know that, for at least δ_{mn} time units, the radiation has to pass through bixel (m, n) and not through bixel $(m, n-1)$. To achieve this the left leaf in the m-th row has to be placed in position n for at least δ_{mn} time units. So, the positive entries of the matrix Δ give a lower bound on the time during which the left leaves have to be in a certain position. Similarly, the negative entries of Δ give a lower bound on the time during which the right leaves have to be in a certain position.

Let now $\Delta^+ = (\delta^+_{mn})$ and $\Delta^- = (\delta^-_{mn})$ be the matrices of size $M \times (N+1)$ defined by $\delta^+_{mn} := \max\{0, \Delta_{mn}\}$ and $\delta^-_{mn} := \max\{0, -\Delta_{mn}\}$. These matrices describe the *unavoidable* leaf positions. The maximum row sum of Δ^+ (or Δ^-) gives a lower bound on the optimal beam-on-time. It turns out that a decomposition of I with beam-on-time equal to this lower bound (and hence an optimal decomposition) can be found by matching the first left leaf position in Δ^+ with the first right leaf position in Δ^-, the second with the second, ... and so on. Each row is processed independently from the others. This algorithm is called the *standard decomposition algorithm*. It has complexity $O(MN + KM)$ where K is the number of matrices output by the algorithm [4].

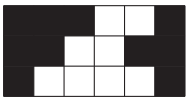
[4] Note that $K = O(MN)$ so the algorithm runs in $O(MN + M^2N)$.

$\Delta^+ = (5 \quad 0 \quad 0 \quad 2 \quad 0),$ $\Delta^- = (0 \quad 4 \quad 0 \quad 0 \quad 3)$

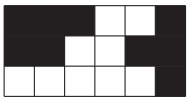
We investigate variants of BOT with additional constraints on the matrices used in the decomposition. Constraints appearing in the application include the interleaf motion and interleaf distance constraint. The former constraint says that the end of a left leaf cannot be located at the right of the end of a right leaf of an adjacent row (in other words leaves in adjacent rows cannot "overlap"). The BOT under this constraint was previously studied by, e.g., Baatar, Hamacher, Ehrgott and Woeginger [3].

Fig. 1. The interleaf motion constraint forbids such leaf configurations.

The interleaf distance constraint is new and says that the distance between the ends of two left (or two right) leaves can not be bigger than a certain constant c.



ok



bad

Fig. 2. The interleaf distance constraint for a constant $c = 2$.

The problems are formulated, after an idea of Baatar *et al.* [3], by introducing a $M \times (N + 1)$ non-negative integer matrix W describing the *extra* leaf positions. The decomposition is implicitly described by the pair of matrices $\Delta^+ + W$ and $\Delta^- + W$ (via the standard decomposition algorithm).

Baatar *et al.* [3] propose a complicated algorithm to solve the resulting IP with complexity $O(M^2N + KM)$ (they only consider the interleaf motion constraint). Here, for both constraints, we prove that finding an optimal decomposition amounts to finding a maximum value potential in an auxiliary network with integer arc lengths. This leads to cleaner, more efficient algorithms. We give a $O(MN + KM)$ algorithm to solve the problem under the interleaf distance constraint. We also give a $O(Mlog(M)N + KM)$ algorithm for the problem under the interleaf motion constraint and hence improve on the $O(M^2N + KM)$ algorithm of Baatar *et al.* Moreover we can show the problem can still be solved in $O(Mlog(M)N + KM)$ time when both constraints are considered simultaneously. We believe that our algorithms are asymptotically optimal.

Acknowledgments

We thank Pierre Scalliet and Stefaan Vynckier from the radiotherapy service of the *Cliniques universitaires Saint-Luc* (Brussels) and Stephane Simon from the *Institut Bordet* (Brussels) for several discussions on the subject of the article.

References

[1] R.K. Ahuja, H.W. Hamacher, A Network Flow Algorithm to Minimize Beam-On Time for Unconstrained Multileaf Collimator Problems in Cancer Radiation Therapy, Networks 2005.

[2] R. Alfredo, C. Siochi, Minimizing static intensity modulation delivery time using an intensity solid paradigm, INT. J. Radiation Oncology Biol. Phys., Vol 43, No. 3, pp. 671-680, 1999

[3] D. Baatar, H.W. Hamacher, M. Ehrgott, G.J. Woeginger, Decomposition of Integer Matrices and Multileaf Collimator Sequencing, Discrete Applied Mathematics. 152: 6-34, 2005.

[4] N. Boland, H.W. Hamacher, F. Lenzen, Minimizing Beam-On Time in Cancer Radiation Treatment Using Multileaf Collimators, Networks, vol 43(4), 226-240 2004.

[5] P. Xia, L. Verhey, Intensity modulation with static MLC fields, (abstr.) Med Phys, 1997

A Note on LP Relaxations for the 1D Cutting Stock Problem with Setup Costs [1]

Alessandro Aloisio [a] Claudio Arbib [a] Fabrizio Marinelli [b,*]

[a] *Università degli Studi di L'Aquila, Dipartimento di Informatica, via Vetoio, I-67010 Coppito, L'Aquila, Italia*

[b] *Università Politecnica delle Marche, Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione via Brecce Bianche, I-60131 Ancona, Italy*

Abstract

We discuss two formulations of the Pattern Minimization Problem: [*Van*] introduced by Vanderbeck, and [*GG*] obtained adding setup variables to the cutting stock formulation of Gilmore-Gomory. Let $z_V^{LP}(\mathbf{u})$ ($z_{GG}^{LP}(\mathbf{u})$) be the bound given by the linear relaxation of the former (the latter) under a given vector $\mathbf{u} = (u_k)$ of parameters. We show that $z_{GG}^{LP}(\mathbf{u}) \geq z_V^{LP}(\mathbf{u})$, and provide a case where the inequality holds strict.

Key words: Cutting Stock, Integer Decomposition, Linear Relaxations

1. The Problem

Let I be a set of one-dimensional part types to be produced by cutting identical stock items of given length w. Let $w_i < w$ (let d_i) denote the length (the demand) of part type $i \in I$. In the 1-dimensional Cutting Stock Problem (1-CSP) one wants to produce d_i parts for each $i \in I$ minimizing the number of used stock items. A solution to the 1-CSP gives a set of cutting patterns | each describing a distinct way of packing part types of I into a single stock item | and the number of times each pattern is replicated (activation level). In the Pattern Minimization Problem (1-PMP), see [5], a solution that uses a minimum number of cutting patterns is searched among all those with a minimum number z^* of stock items, where z^* is given by a preliminary solution of the relevant 1-CSP.

* Corresponding author.

[1] This work has been partially financed by the EU Commission, within the Cooperative Research Project SCOOP (contract n. 032998), coordinated by Università Politecnica delle Marche, Ancona, Italy.

In general, the 1-PMP is considerably hard [1]. A non-linear integer formulation [Kan] can easily be derived from the 1-CSP compact assignment model [4]. Here, x_{ij} is the number of items of part type i obtained from a stock item cut according to pattern j, z_j is the number of times pattern j is used, and y_j is a 0-1 variable indicating whether pattern j is used at all, or not:

$$[\textit{Kan}] \qquad \min \sum_{j=1}^{z^*} y_j \tag{1}$$

$$\sum_{j=1}^{z^*} z_j x_{ij} = d_i \qquad i \in I \tag{2}$$

$$\sum_{i \in I} w_i x_{ij} \leq w y_j \qquad j = 1 \ldots z^* \tag{3}$$

$$\sum_{j=1}^{z^*} z_j \leq z^* \tag{4}$$

$$z_j \leq z^* y_j \qquad j = 1 \ldots z^* \tag{5}$$

$$z_j \geq 0 \qquad j = 1 \ldots z^* \tag{6}$$

$$x_{ij} \geq 0 \qquad i \in I, j = 1 \ldots z^* \tag{7}$$

$$0 \leq y_j \leq 1 \qquad j = 1 \ldots z^* \tag{8}$$

$$x_{ij}, z_j, y_j \text{ integer} \qquad i \in I, j = 1 \ldots z^* \tag{9}$$

A lower bound to [Kan] is given by the optimal value of an associated bin-packing problem, obtained by setting $d_i = 1, \forall i$, or by a lower bound to this value [3]. These bounds are however too weak for an effective use within a branch-and-bound algorithm.

2. Reformulations and lower bounds by linear relaxation

Reformulating [Kan] by discretization, see [6], gives tighter bounds to the 1-PMP. Indeed, different master formulations can be drawn from [Kan], depending on the set of dualized constraints. In [5], the author describes a 1-PMP master formulation [*Van*] obtained by dualizing (1) and (3), or equivalently, from discretization of the polyedron defined by (2) and (4)-(8).

An alternative master formulation [*GG*], very close to that of Gilmore and Gomory for the 1-CSP [2] with the addition of fixed setup costs, derives from discretizing the polyedron defined by (2), (6) and (7).

Both formulations [Van] and [GG] involve variables associated to cutting patterns. Since the set K of all the feasible cutting patterns grows exponentially in the number of part types, column generation procedures are required to solve the linear relaxations [Van^{LP}] and [GG^{LP}] of respectively [Van] and [GG]. In fact, such column generation algorithms are very similar to each other since both formula-

tions lead to the same pricing problem. Therefore the quality of the bounds provided by the linear relaxations $[Van^{LP}]$ and $[GG^{LP}]$ remains a crucial aspect of the design of an effective exact algorithm.

In [5] the Author discards formulation $[GG]$, and develops a branch-and-price algorithm for $[Van]$: he argues in fact that $[Van^{LP}]$ results from the dualization of fewer constraints of $[Kan]$ than $[GG^{LP}]$, and therefore, by Lagrangian theory, the former must be stronger than the latter.

We notice however that one gets $[Van]$ and $[GG]$ by mere dualization of $[Kan]$ *only if* the upper bound u_k to the activation level of the k-th cutting pattern, $k \in K$, is set to the trivial value z^*. But both $[Van]$ and $[GG]$ use specific upper bounds u_k instead of the trivial value z^*, and indeed their properties strongly depend on those u_k. The main purpose of this note is then to bring the attention on the crucial role of the u_k's and to show that, in order to find a good and practical exact algorithm, formulation $[GG]$ should not in principle be discarded. In fact, let $z_V^{LP}(\mathbf{u})$ and $z_{GG}^{LP}(\mathbf{u})$ denote the optimal value of programs $[Van^{LP}]$ and $[GG^{LP}]$, for $\mathbf{u} = (u_k)_{k \in K}$. Then

Proposition 1 $z_V^{LP}(\mathbf{u}) \leq z_{GG}^{LP}(\mathbf{u})$*, and the inequality holds strict for some vector* $\mathbf{u} \in \mathbb{R}^{|K|}$ *of upper bounds to the activation level of cutting patterns.*

References

[1] Dyckhoff, H., G. Scheithauer, J. Terno. 1997. Cutting and Packing (C&P), in: M. DellŠAmico, F. Maffioli, S. Martello (Eds.), *Annotated Bibliographies in Combinatorial Optimization*, Wiley (Chichester, 1997) 393-413.

[2] Gilmore, P.C., R.E. Gomory. 1963. A Linear Programming Approach to the Cutting Stock Problem – Part II, *Operations Research* **11** 863-888.

[3] Haouari, M., A. Gharbi. 2005. Fast lifting procedures for the bin packing problem, *Discrete Optimization* **2** 201-218.

[4] Kantorovich, L.V. 1960. Mathematical models of organising and planning production, *Management Science*, **6** 366-422.

[5] Vanderbeck, F. 2000. Exact Algorithm for Minimising the Number of Setups in the One-Dimensional Cutting Stock Problem, *Operations Research* **48** 915-926.

[6] Vanderbeck, F., M.W.P. Savelsbergh. 2006. A generic view of Dantzig-Wolfe decomposition in mixed integer programming, *Operations Research Letters* **34** 296-306.

Location

Room B, Session 7

Thursday 15, 11:00-12:30

Locating Median Paths on Connected Outerplanar Graphs

Isabella Lari [a] Federica Ricca [a] Andrea Scozzari [b,*]
Ronald I. Becker [c]

[a] *Università di Roma "La Sapienza", Dip. Statistica, Probabilità e Statistiche Applicate*

[b] *Università di Roma "La Sapienza", Dip. Matematica per le Decisioni Economiche, Finanziarie ed Assicurative*

[c] *University of Cape Town, Dept. of Mathematics and Applied Mathematics*

Key words: Path location, path median, biconnected outerplanar graphs.

1. Extended Abstract

During the last two decades, there has been a growing interest in the developing of location models on networks, with particular attention to the location of extensive facilities, such as paths or trees. Almost in all cases, the criteria used are the *minsum* criterion, according to which the sum of the distances from all the vertices of the network to the facility is minimized, and the *minimax* criterion, that is, the distance from the facility to the farthest vertex in the network is minimized. The present paper investigates the problem of locating a path-shaped facility with the minsum criterion without restrictions on the length of the path on *outerplanar* graphs. Examples of such a problem include the location of pipelines, evacuation routes, mass transit routes or routing a highway through a road network, and public transit lines. An optimal path for this problem is also referred to as a *median path* and we will refer to the problem under study as the Median Path Problem (MPP).
In the literature, the median path problem was widely studied when there is a restriction on the length of the path. In fact, on general networks this problem is NP-complete [3, 5, 11]. In particular, in [5] it is shown that it is NP-Complete on planar graphs with vertex degree less than or equal to 5, while [3] provides the same result on rectangular grid graphs. In [11] it is shown that the median path problem with length at most equal to a given constant, is NP-hard on outerplanar graphs, but in [7] it is actually shown that the same problem is NP-hard even on the class of cactus graphs. Nevertheless, [11] provides a pseudo-polynomial time algorithm for the solution of MPP with restricted length on series-parallel graphs. Moreover,

* Corresponding author.

if the graph is unweighted, the above algorithm has an overall time complexity of $O(n^7 log\, n)$, where n is the number of vertices of the graph.
For MPP with restricted length on general networks different directions were investigated: in [7] a metaheuristic approach was presented, while [2] suggests a branch-and-cut algorithm. Finally, a number of papers have investigated the problem of locating median paths with restricted length on trees and efficient polynomial time algorithms were provided (see, e.g., [1, 4, 5, 8, 10]).
Since also MPP without length restrictions is NP-hard on general networks [5], this problem was mainly studied on trees, as well [9, 10, 12]. To the best of our knowledge, MPP without length restrictions has not been studied yet on networks with cycles.
In this paper we study this problem on the class of outerplanar graphs. Notice that, if we consider a biconnected outerplanar graph G, the solution is trivial, since a median path without restrictions on the length is simply given by the path passing through all the vertices on the outercycle of G. On the other hand, this is not true if we consider the case of finding a median path between two fixed end vertices in a biconnected outerplanar graph. Actually, this will turn out to be a special case of our more general problem for which we will provide an algorithm, linear in the number of vertices of the graph. In this paper we consider the more general class of *connected* outerplanar graphs (or, simply, outerplanar graphs). In particular, we focus our attention on the case in which equal weights are assigned to the edges of G, while nonnegative weights are associated to the vertices of G.

Given an outerplanar graph $G = (V, E)$, with $|V| = n$, it can be suitably decomposed into blocks and bridges [6] and it can be represented by a tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$, where $V_{\mathcal{T}}$ is the set of blocks and bridges of G. There is an edge between two vertices of $\mathcal{T}$ if they share a cut vertex in G. We call $\mathcal{T}$ the *representation tree* of G. In our algorithm we root $\mathcal{T}$ at any block or bridge H, and we denote the resulting rooted tree by $\mathcal{T}_H$. In order to provide an algorithm for the MPP in G, we need a preprocessing phase for computing some quantities. We exploit the structure of the representation tree of G both in the preprocessing and in the algorithm.

In the preprocessing phase, we visit the tree $\mathcal{T}_H$ bottom up, level-by-level, and compute some quantities associated both to its edges and its vertices. Unlike the preprocessing usually implemented for median path location problems on trees, here the particular structure of each vertex of $\mathcal{T}_H$ (i.e., a block of G) requires a more complex procedure to compute the necessary quantities.

After the preprocessing, the algorithm works by rooting the corresponding representation tree $\mathcal{T}$ at any block or bridge H and visits $\mathcal{T}_H$ top down by a breadth-first-search. For each vertex $u \in V$ the algorithm computes the minimum sum of the distances of a path from a vertex in H to vertex u. Among all the paths found so far, the algorithm selects the one with the minimum sum of the distances, $P(H)$. This procedure is repeated by rooting $\mathcal{T}$ at every possible block or bridge. Among all the paths $P(H)$, the optimal path P^* for MPP corresponds to the one with the minimum sum of the distances.

Let k be the number of blocks and bridges in G. We prove that, for a given H,

the preprocessing phase applied to $\mathcal{T}_H$ runs in $O(n)$ time, implying that the time complexity of the preprocessing is $O(kn)$.
We also prove that, for a given H, $P(H)$ can be found in $O(n)$ time. Hence, the overall time complexity for solving MPP is $O(kn)$.

In addition, as a byproduct of our main algorithm, we provide a linear time procedure to find a median path without restrictions on its length between two fixed vertices in a biconnected outerplanar graph.

References

[1] S. Alstrup, P.W. Lauridsen, P. Sommerlund, and M. Thorup, Finding cores of limited length. Techincal Report, The IT University of Copenhagen (2001).

[2] P. Avella, M. Boccia, A. Sforza, I. Vasil'ev, A branch-and-cut algorithm for the median-path problem. Computational Optimization and Applications, 32 (2005), 215-230.

[3] R.I. Becker, I. Lari, A. Scozzari, G. Storchi, The location of median paths on grid graphs. Annals of Operations Research, 150 (2007), 65-78.

[4] R.I. Becker, Y.I. Chiang, I. Lari, A. Scozzari, G. Storchi, Finding the ℓ-core of a tree. Discrete Applied Mathematics, 118 (2002), 25-42.

[5] S.L. Hakimi, E.F. Schmeichel, M. Labbé, On locating path- or tree- shaped facilities on networks. Networks, 23 (1993), 543-555.

[6] F. Harary. Graph Theory. Reading, MA: Addison-Wesley, 1994.

[7] I. Lari, F. Ricca, A. Scozzari, Comparing different metaheuristic approaches for the median path problem with bounded length. European Journal of Operational Research, to appear (doi:10.1016/j.ejor.2007.07.001).

[8] E. Minieka. The optimal location of a path or tree in a tree network. Networks, 15 (1985), 309-321.

[9] C.A. Morgan, J.P. Slater, A linear algorithm for a core of a tree. Journal of Algorithms, 1 (1980), 247-258.

[10] S. Peng, A.B. Stephens, Y. Yesha, Algorithms for a core and a k-tree core of a tree. Journal of Algorithms, 15 (1993). 143-159.

[11] M.B. Richey, Optimal location of a path or tree on a network with cycles. Networks, 20 (1990), 391-407.

[12] P.J. Slater, Locating central paths in a graph. Transportation Science, 16 (1982), 1-18.

Dynamic Programming for Optimization of Capacitor Allocation in Power Distribution Networks

José Federico Vizcaino González [a] Christiano Lyra [a,*]

[a] *School of Electrical and Computer Engineering, University of Campinas (UNICAMP), Av. Albert Einstein 400, 13083-852 Campinas, São Paulo, Brazil*

Key words: power distribution networks, electric power distribution, loss reduction, capacitor allocation, capacitor placement and sizing, dynamic programming.

1. Introduction

This paper could be named “The Return of Dynamic Programming for Optimization of Capacitor Allocation in Power Distribution Networks". Indeed, it revisits some forty years old ideas that seem to have been forgotten in the area of capacitor sizing and placement in power distribution networks.

As energy travels from generation plants to customers, electrical resistance in transmission and distribution lines causes dissipation of energy. These *technical losses* are large fees on electric power systems; typical figures for losses in the literature amount to around 7% of total energy production, 2% in transmission and 5% in distribution [1]. Loss reduction can be seen as a hidden source of energy.

Capacitor allocation is a cost-effective way to decrease losses in power distribution networks. They can provide local reactive power that reduces technical losses by avoiding part of the reactive energy flows in power lines. The optimal capacitor allocation problem searches to find out the best places and sizes for capacitors in distribution networks. The objective function for the problem reflects a compromise between cost of capacitors and energy savings along a specified payback period.

Techniques to unveil the best alternatives for capacitor allocation in distribution networks have been developed for more than 50 years. During the sixties mathematical programming techniques were blooming; among them, dynamic programming. A lot of research effort was applied to model problems in such a way that it could be addressed as a dynamic programming problem—for which an optimal solution could be easily found. Capacitor allocation in power distribution networks did not stay aside from the tide. With some simplifying assumptions, Duran [2]

* Corresponding author.

proposed an approach based on dynamic programming to find optimal solutions to the problem.

Most electric distribution networks operate with a radial configuration. Using a graph terminology, a distribution *feeder* is a tree rooted at a distribution substation that provides a unique path from the substation to each load point. The main simplifying assumption adopted by Duran [2] was that these trees do not have lateral branches. Under this assumption, he showed that the capacitor allocation problem could be represented as a one-dimensional dynamic programming problem. When the lateral branches are considered, his formulation seems to require more dimensions, one for each lateral branch, what precludes the application of the technique for real scale distribution networks.

In the following decades after the contribution of Duran [2] many approaches based on heuristics were proposed to solve the capacitor allocation problem in radial distribution networks. To name a few, Baran and Wu proposed a heuristic method guided by the solution of a mixed integer programming problem [3, 4]; Gallego et al. adopted tabu search [5]; Mendes et al. proposed a hybrid genetic algorithm [6].

This paper revisits the ideas of Duran [2]. It shows that with further examination their simplifying assumptions are not necessary; dynamic programming can be applied to find optimal solutions for capacitor allocation in real scale distribution networks. The chief concept is to project the multidimensional dynamic programming formulation into an equivalent one-dimensional representation. Ideas and data structures borrowed from network flows optimization algorithms allow implementation of the new approach.

2. Problem Discussion

Technical losses in a section k of a power distribution line can be computed as the product of the square of electric currents flowing in the line by the equivalent resistance of the section. Currents can be decomposed into *in-phase components* (i_{Pk}) and *quadrature components* (i_{Qk}). The in-phase component is associated with the *active power* P_k, also named “useful power", because it is the power used to produce work, light and heat; the quadrature component is associated with the *reactive power* Q_k, a consequence of the electric physics that flows back and forth in power lines without being actually “consumed".

Figure 1 shows a simplified diagram of a distribution feeder. The in-phase component can be computed as $i_{Pk} = \frac{P_k}{V_k}$, where V_k is the voltage at the first upstream node—for a graph representation of a distribution network please refer to [7]. The quadrature component can be computed as $i_{Qk} = \frac{Q_k}{V_k}$. Losses in the section k are computed as $l_k = r_k \frac{(P_k)^2+(Q_k)^2}{(V_k)^2}$, where r_k is the resistance of section k.

Losses in power lines can be reduced by installing capacitors (C_i) of adequate size

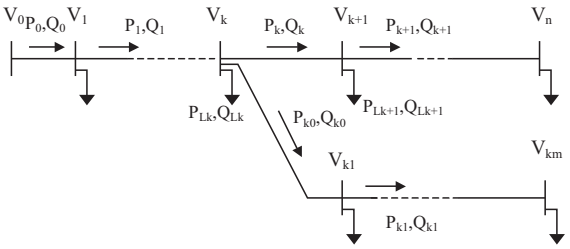


Fig. 1. Schematic Diagram of a Power Distribution Feeder.

at some nodes of the distribution network. These capacitors provide local reactive power Q_{Ci} and, consequently, reactive currents i_{Ci}, in opposite direction of the quadrature components. Thus, $l_k = r_k \frac{(P_k)^2+(Q_k-Q_{Ci})^2}{(V_k)^2}$ when a capacitor C_i is installed at node k, downstream from section k.

The objective of the capacitor allocation problem can be expressed as

$$Min_{s\in S_C}[\sum_{i\in S_C} f(C_i) + \alpha_{et}\sum_{t\in T}\tau_t \sum_{k\in N}\sum_{j\in A_k} r_{kj}\frac{(P_{kj})^2+(Q_{kj})^2}{(V_k)^2}] \qquad (1)$$

where S_C is the set of capacitors for possible allocation in the network, $f(C_i)$ is the cost of a capacitor C_i, α_{et} is the value of energy during interval t, τ_t is the duration of interval t, T is the set of time intervals, N is the set of nodes in the distribution network, A_k is the set of arcs kj with origin at node k, r_{kj} is the resistance of arc kj, P_{kj} is the total active power flow in arc kj and Q_{kj} is the total reactive power flow in arc kj. All solutions must satisfy power flow equations, electrical constraints and specific operational goals [6].

The dynamic programming approach (DP) proposed by Duran [2] associates *stages* to the nodes of the distribution network, *control variable* at a node k to the capacitive reactive power (Q_{Ci}) injected at the node and *state* to the total capacitive power flowing in the arc immediately upstream from a node k. If there is only one arc downstream from each node, it leads to a one-dimensional DP problem. If there are more than one arc downstream from the nodes, this DP approach needs additional dimensions, one for each lateral branch; in these cases, computing a solution for real feeders with many arcs downstream from their nodes is an impossible task (thousands of lateral branches is a common instance). That is why DP has been long forgotten as a technique to solve the capacitor allocation problem.

The paper proposes a concept that complements the original ideas of Duran [2]. Instead of increasing the dimension of state variables for each additional downstream branch, it proposes to design very simple secondary optimization problems that give the optimal array of capacitors for the whole set of branches downstream from a node. Adoption of this concept allows projecting the problem into a one-dimensional DP problem. In doing so, it allows to rescue DP as a technique to solve real capacitor allocation problems.

Remind that DP finds a global optimum to the problem, what gives it a special appeal compared to heuristic approaches.

Acknowledgment. The authors kindly acknowledge the support from the Brazilian National Council for Scientific and Technological Development (CNPq).

References

[1] Jennings B. Bunch and Richard D. Miller and James E. Wheeler (1982), “Distribution System Integrated Voltage and Reactive Power Control", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-101, pp. 284–289.

[2] H. Duran (1968), “Optimum Number, Location and Size of Shunt Capacitors in Radial Distribution Feeders: A Dynamic Programming Approach", *IEEE Transactions on Power Apparatus and Systems*, Vol. PAS-87, pp. 1769–1774.

[3] M. Baran and F. Wu (1989a), “Optimal Capacitor Placement on Radial Distribution Systems", *IEEE Transactions on Power Delevery*, Vol. 4, pp. 725–733.

[4] M. Baran and F. Wu (1989b), “Optimal Sizing of Capacitors Placed on a Radial Distribution System", *IEEE Transactions on Power Delevery*, Vol. 4, pp. 735–743.

[5] R. A. Gallego, A. J. Monticelli and R. Romero (2001), “Optimal Capacitor Placement in Radial Distribution Networks", *IEEE Transactions on Power Systems*, Vol.16, pp. 630–637.

[6] A. Mendes, P. Franca, C. Lyra, C. Pissarra and C. Cavellucci (2005), “Capacitor Placement in Large-Sized Radial Distribution Networks", *IEE Proceedings—Generation, Transmission and Distribution*, Vol. 152, pp. 496–502.

[7] C. Cavellucci and C. Lyra (1997), “Minimization of Energy Losses in Electric Power Distribution Systems by Intelligent Search Strategies", *International Transactions in Op. Research*, Vol.4, pp. 23–33.

Approximating (r, p)-Centroid on a Path

J. Spoerhase [a] H.-C. Wirth [a,*]

[a] *Department of Computer Science, University of Würzburg*

1. Introduction

We investigate a problem from the area of competitive location where two competing providers, the leader and the follower, sequentially place facilities into a market in order to maximize their revenue. It is assumed that all competitors provide the same type of good or service. Hence the user preference can be expressed solely in terms of distances to the locations of the servers. In our scenario the distances are induced by an underlying edge weighted graph.

The benefit of each competitor is measured by the size of his party, i.e., the total demand (or weight) of the users connecting to the competitor. The providers act in a non-cooperative way and only aim at maximizing their own benefit. Once the leader has chosen his position, the follower is able to determine an optimal location. Hence the follower's reaction is predictable, which the leader can take into account when he makes the initial decision.

2. Problem Definition

Consider an undirected graph $G = (V, E)$ with positive edge lengths $d\colon E \to \mathbb{N}^+$. An edge of the graph can be considered as an infinite set of *points*. A point x on edge $e = (u, v)$ is specified by the distance from one of the endpoints of e, and the remaining distance is derived from the invariant $d(u, x) + d(x, v) = d(e)$. Notice that the set of points of a graph includes the set of nodes. All points which are not nodes are called *inner points*. In the sequel we will use G (and e) both for denoting the graph (the edge) and for denoting all of its points, as the meaning will become clear from the context. In the sense of these considerations the edge length function d is extended to a distance function $d\colon G \times G \to \mathbb{N}_0^+$ defined on all pairs of points. Nonnegative node weights $w\colon V \to \mathbb{N}_0^+$ specify the demand of users who are always placed at nodes of the graph.

Let $X, Y \subset G$ be finite sets of nodes or points, specifying a server placement of the leader or follower player, respectively. The distance of a user u to a point set M is $d(u, M) := \min_{m \in M} d(u, m)$. A user u prefers the follower if $d(u, Y) < d(u, X)$.

* Corresponding author.

By $w(Y \prec X) := \sum\{\, w(u) \mid d(u,Y) < d(u,X) \,\}$ the total weight of the follower party is denoted.

Let $r, p \in \mathbb{N}$. Leader and follower are allowed to place p or r facilities, respectively, into the graph. Let $X_p \subset G$ be a set of $|X_p| = p$ points. Let

$$w_r(X_p) := \max_{\substack{Y_r \subset G \\ |Y_r|=r}} w(Y_r \prec X_p)$$

be the maximum influence any r-element follower placement can gain over a fixed leader placement X_p. An *absolute* (r, X_p)*-medianoid* of the graph is any set $Y_r \subset G$ of $|Y_r| = r$ points where $w(Y_r \prec X_p) = w_r(X_p)$ is attained. Let

$$w_{r,p} := \min_{\substack{X_p \subset G \\ |X_p|=p}} w_r(X_p)\,.$$

An *absolute* (r,p)*-centroid* of the graph is any set $X_p \subset G$ of $|X_p| = p$ points where $w_r(X_p) = w_{r,p}$ is attained. The notions *discrete* (r, X_p)*-medianoid* and *discrete* (r,p)*-centroid* are defined similarly, with the server sets restricted to nodes $X_p, Y_r \subseteq V$ rather than points.

Previous Results

The (r,p)-centroid and (r, X_p)-medianoid problems have been introduced in [1]. The discrete (r,p)-centroid on a path is solvable in polynomial time $O(pn^4)$ while the absolute (r,p)-centroid on a path and the discrete (r,p)-centroid on a spider are NP-hard [4].

3. Absolute (r,p)-centroid on a path

Let $P = (v_1, \ldots, v_n)$ the input path, $w\colon V \to \mathbb{N}$ the node weights, $d\colon E \to \mathbb{N}_+$ be the edge lengths.

For arbitrary real numbers $x, y \in \mathbb{R}$ we denote by $x \dot{-} y := \max\{x - y, 0\}$ the asymmetric difference. With each leader positioning X_p we associate a vector $(\delta_1(X_p), \ldots, \delta_{2p}(X_p))$ where $\delta_i(X_p) := w_i(X_p) - w_{i-1}(X_p)$ is the incremental gain of placing the ith follower. Hence $w_r(X_p) = \sum_{i=1}^{r} \delta_i(X_p)$. Since $\delta_1(X_p) \geq \ldots \geq \delta_{2p}(X_p)$, finding $w_{r,p}$ can be considered as an r-sum minimization problem [2]. The authors reduce in a general framework r-sum optimization problems to minisum problems. In our context we define z*-reduced weights* (for $z \in \mathbb{N}$)

$$w^{(z)}(X_p) := \sum_{i=1}^{2p} \Big((w_i(X_p) - w_{i-1}(X_p)) \dot{-}\, z \Big)$$

and establish the following result:

Lemma 1 (Reduction from r-sum to minisum) *There is a suitable $z \in \mathbb{N}$ such that each optimum under minisum function $w^{(z)}$ is also an optimum under r-sum function w_r.*

In order to evaluate an optimum of $w^{(z)}$ for a given parameter $z \in \mathbb{N}$ we use a dynamic programming approach. Assume the path is populated incrementally with leader servers from left to right. Let $R(\pi, W) =: R$ denote the rightmost feasible leader position of placing π servers such that $w^{(z)}$ restricted to the interval $[0, R]$ does not exceed W. Then, for any $\pi = 1, \ldots, p$ and $W = 0, \ldots, w(P)$,

$$R^{(z)}(\pi+1, W) = \max \left\{ x \;\middle|\; \begin{array}{ll} \exists (W_0, \tilde{W}) \colon W_0 + \tilde{W} = W & \text{and} \\ w^{(z)}\Big(R^{(z)}(\pi, W_0), x\Big) \leq W & \end{array} \right\}.$$

We can assume w.l.o.g. that there is always placed a leader at the rightmost node v_n. (Otherwise, add a dummy node with suitable weight to the path, and increment p by one.) Taking this into respect, the $w^{(z)}$-optimum weight can be derived from the outcome vector $R^{(z)}(p, \cdot)$ of the dynamic programming table by

$$\min_{X_p} w^{(z)}(X_p) = \min\{\, W \mid R^{(z)}(p, W) \geq d(v_1, v_n) \,\};$$

a corresponding leader server placement X_p can be derived from maintaining the positions during the dynamic program.

In order to compute the desired result it suffices to compute all $w^{(z)}$-optima for $z \in [0, w(P)]$ and output a placement X_p where $w_r(X_p)$ is minimal.

Lemma 2 (Solving r-sum problem) *The problem of determining $w_{r,p}$ and a corresponding leader placement X_p (i.e., where $w_r(X_p) = w_{r,p}$ is attained) can be solved in pseudo-polynomial running time $O(p \cdot w(P)^2 \cdot n^2)$.*

From this result we can derive a fully polynomial time approximation scheme (FPTAS) applying a standard scaling technique to the weights of the nodes [3].

Theorem 1 (Approximation) *There is a FPTAS for absolute (r, p)-centroid on a path.*

References

[1] S. L. Hakimi · *On locating new facilities in a competitive environment* · European Journal of Operations Research **12** (1983), 29–35.

[2] A. P. Punnen and Y. P. Aneja · *On k-sum optimization* · Operations Research Letters **18** (1996), 233–236.

[3] C. H. Papadimitriou and K. Steiglitz · *Combinatorial optimization* · Prentice Hall, 1982.

[4] J. Spoerhase and H.-C. Wirth · *(r, p)-centroid problems on paths and trees* · Tech. Report no. 441, University of Würzburg, Department of Computer Science, 2008, `http://www.informatik.uni-wuerzburg.de/forschung/technical_reports/`.

List of Corresponding Authors:

Name	email	page
Edoardo Amaldi	amaldi@elet.polimi.it	112
Evangelos Bampas	ebamp@cs.ntua.gr	35
Alberto Caprara	acaprara@deis.unibo.it	16
Daniele Catanzaro	dacatanz@ulb.ac.be	48
Roberto Cordone	cordone@dti.unimi.it	52
Maria L.A.G. Cremers	m.l.a.g.cremers@rug.nl	78
Paolo Detti	detti@dii.unisi.it	39
Kanika Dhyani	dhyani@elet.polimi.it	162
Ardeshir Dolati	dolati@shahed.ac.ir	11
Niklaus Eggenberg	niklaus.eggenberg@epfl.ch	72
Céline Engelbeen	cengelbe@ulb.ac.be	177
Rija Erveš	rija.erves@uni-mb.si	144
Nikolaos Fountoulakis	nikolaos@maths.bham.ac.uk	88
Bernhard Fuchs	fuchs@ibr.cs.tu-bs.de	26
Vassilis Giakoumakis	Vassilis.Giakoumakis@u-picardie.fr	94
Stefano Gualandi	gualandi@elet.polimi.it	32
Çiğdem Güler	gueler@mathematik.uni-kl.de	108
Leo Liberti	liberti@lix.polytechnique.fr	66
Marina Lipshteyn	marina.lipshteyn@Ethos-Networks.com	152
Christiano Lyra	chrlyra@densis.fee.unicamp.br	189
Raphael Machado	raphael@cos.ufrj.br	118
Gordana Manić	manic.gordana@gmail.com	62
Fabrizio Marinelli	marinelli@diiga.univpm.it	181
Lilian Markenzon	markenzon@nce.ufrj.br	174
Aurora Morgana	morgana@mat.uniromal.it	123
Giacomo Nannicini	giacomo.nannicini@v-trafic.com	132
Tim Nieberg	nieberg@or.uni-bonn.de	44
Hans-Christoph Wirth	wirth@informatik.uni-wuerzburg.de	193

Jacob Jan Paulus	j.j.paulus@ewi.utwente.nl	82
Aharona Pfeffer	pfeffer@post.tau.ac.il	139
Katja Prnaver	katja.prnaver@imfm.uni-lj.si	127
Romain Ravaux	romain.ravaux@prism.uvsq.fr	148
Cid C. de Souza	cid@ic.unicamp.br	104, 158
Christian Schulte	schulte@or.uni-bonn.de	22
Andrea Scozzari	andrea.scozzari@uniroma1.it	186
Rüdiger Stephan	stephan@math.tu-berlin.de	58
Dimitrios M. Thilikos	saket@ii.uib.no	2
Paolo Toth	paolo.toth@unibo.it	136
Ping-Ying Tsai	bytsai0808@gmail.com	7
Ilaria Vacca	ilaria.vacca@epfl.ch	167
Juan A. Rodríguez-Velázquez	juanalberto.rodriguez@urv.cat	98

List of Participants:

Name	email	page
Roberto Aringhieri	aringhieri@dti.unimi.it	48
Evangelos Bampas	ebamp@cs.ntua.gr	35
Andrea Bettinelli	andrea.bettinelli@unimi.it	
Maurizio Bruglieri	maurizio.bruglieri@polimi.it	
Valentina Cacchiani	valentina.cacchiani@unibo.it	136
Sonia Cafieri	cafieri@lix.polytechnique.fr	
Daniele Catanzaro	dacatanz@ulb.ac.be	48
Victor Cavalcante	victor.cavalcante@ic.unicamp.br	104
Alberto Ceselli	ceselli@dti.unimi.it	162
Roberto Cordone	cordone@dti.unimi.it	52
Maria L.A.G. Cremers	m.l.a.g.cremers@rug.nl	78
Paulo Renato Da Costa Pereira	prenato@ime.eb.br	174
Kanika Dhyani	dhyani@elet.polimi.it	162
Ardeshir Dolati	dolati@shahed.ac.ir	11
Niklaus Eggenberg	niklaus.eggenberg@epfl.ch	72
Céline Engelbeen	cengelbe@ulb.ac.be	177
Rija Erveš	rija.erves@uni-mb.si	144
Ulrich Faigle	faigle@zpr.uni-koeln.de	
Henning Fernau	fernau@uni-trier.de	98
Nikolaos Fountoulakis	nikolaos@maths.bham.ac.uk	88
Bernhard Fuchs	fuchs@ibr.cs.tu-bs.de	26
Stefano Gualandi	gualandi@elet.polimi.it	32
Çiğdem Güler	gueler@mathematik.uni-kl.de	108
Edna Ayako Hoshino	edna@ic.unicamp.br	158
Johann L. Hurink	j.l.hurink@utwente.nl	82
Walter Kern	kern@math.utwente.nl	
Leo Liberti	liberti@lix.polytechnique.fr	66
Marina Lipshteyn	marina.lipshteyn@Ethos-Networks.com	152
Christiano Lyra	chrlyra@densis.fee.unicamp.br	189

Raphael Machado	raphael@cos.ufrj.br	118
Francesco Maffioli	maffioli@elet.polimi.it	112
Gordana Manić	manic.gordana@gmail.com	62
Fabrizio Marinelli	marinelli@diiga.univpm.it	181
Sheila Morais de Almeida	sheila@ic.unicamp.br	123
Giacomo Nannicini	giacomo.nannicini@v-trafic.com	132
Tim Nieberg	nieberg@or.uni-bonn.de	44
Maddalena Nonato	nntmdl@unife.it	
Cheikh Brahim Ould El Mounir	elmounir@u-picardie.fr	94
Andrea Pacifici	andrea.pacifici@uniroma2.it	39
Jacob Jan Paulus	j.j.paulus@ewi.utwente.nl	82
Vagner Pedrotti	pedrotti@ic.unicamp.br	
Aharona Pfeffer	pfeffer@post.tau.ac.il	139
Katja Prnaver	katja.prnaver@imfm.uni-lj.si	127
Romain Ravaux	romain.ravaux@prism.uvsq.fr	148
Federica Ricca	federica.ricca@uniroma1.it	186
Giovanni Righini	righini@dti.unimi.it	
Matteo Salani	matteo.salani@epfl.ch	72, 167
Rainer Schrader	schrader@zpr.uni-koeln.de	
Christian Schulte	schulte@or.uni-bonn.de	22
Anna Schulze	schulze@zpr.uni.koeln.de	26
Andrea Scozzari	andrea.scozzari@uniroma1.it	186
Joachim Spoerhase	spoerhase@informatik.uni-wuerzburg.de	193
Rüdiger Stephan	stephan@math.tu-berlin.de	58
Dimitrios Thilikos	sedthilk@math.uoa.gr	2
Emiliano Traversi	emiliano.traversi2@unibo.it	16
Ping-Ying Tsai	bytsai0808@gmail.com	7
Ilaria Vacca	ilaria.vacca@epfl.ch	167
Xinhui Wang	xinhuiw@math.utwente.nl	7