CTW04 Workshop on Graphs and Combinatorial Optimization

Scientific Program

Villa Vigoni, Menaggio (CO), Italy

31st May to 2nd June 2004

Leo Liberti, Francesco Maffioli (Eds.)

Introduction

This volume collects the extended abstracts of the contributions that have been selected for presentation at the Workshop.

An additional special volume of Discrete Applied Mathematics (DAM) devoted to the CTW04 Workshop will be prepared. This volume will contain full-length versions of presentations given at the workshop, as well as other contributions related to the topics of the workshop. We hereby invite all participants to submit full-length papers before **30 September 2004**.

CTW04 stands for Cologne-Twente Workshop 2004. These workshops, on Graph Theory and Combinatorial Optimization, are usually organized bi-annually by either Cologne University or Twente University. As its importance grew over time, it was recognized that the frequency of the Workshops should be increased to one per year, and that more institutions should take the leading role in its organization. For the first time, a Cologne-Twente Workshop is organized in a place entirely different from either Cologne or Enschede: the CTW04 Workshop is organized by the Operations Research group of the Dipartimento di Elettronica e Informazione, Politecnico di Milano, and takes place at Villa Vigoni, Menaggio, Como, Italy, between 31st May and 2nd June 2004.

The local organizers thank the members of the program committee:

> E. Amaldi (Milan), H.J. Broersma (Enschede), U. Faigle (Cologne), J.L. Hurink (Enschede), F. Malucelli (Milan), R. Schrader (Cologne), R. Schultz (Duisburg), G.J. Woeginger (Enschede)

for their help in setting up such an attractive program.

We acknowledge the help of the Dipartimento di Elettronica e Informazione at the Politecnico di Milano. Special thanks also go to all the anonymous referees.

Leo Liberti
Francesco Maffioli

Milano, 21st April 2004

Table of Contents

Sessions

There are three plenary invited talks (M. Queyranne — Chair: U. Faigle; L. Stougie — Chair: F. Malucelli; H. Hamacher — Chair: F. Maffioli).

The contributed talks are scheduled into 16 sessions (6 of which are split into two parts). The order of the talks in each session follows the order of the following list. The name of the presenting author is typeset in boldface.

- Advances in Optimization I (31/5, 14:30, Sala Marcolini) – II (31/5, 17, Conference Hall). Chairs: S. Pickl, R. Schultz.
  - D. Lozovanu, **S. Pickl**, *Special dynamic programming technique for multiobjective discrete control and for dynamic games on graph-based networks*
  - L. Liberti, N. Maculan, **S. Kucherenko**, *The kissing number problem: new results from global optimization*
  - A. Chierici, **R. Cordone**, R. Maja, *The demand-dependent optimization of regular train timetables*
  - **W. Bein**, *Knowledge state algorithms and the 2-server problem*
  - **F. Tardella**, *Connections between continuous and combinatorial optimization problems through an extension of the fundamental theorem of Linear Programming*
  - A. Märkert, **R. Schultz**, *On deviation measures in stochastic integer programming*
- Algebraic Structures (2/6, 10:30, Sala Marcolini). Chair: R. Schrader.
  - **H. Gropp**, *More on orbital matrices*
  - **M.R. Emamy-K.**, *The cut number of the n-cube, boolean methods and a geometric connection to threshold logic*
  - F. Maffioli, **N. Zagaglia Salvi**, *A particular class of graphic matroids*
- Applications of Local Search (1/6, 16:30, Library). Chair: L. Liberti.
  - E. Amaldi, **L. Liberti**, F. Maffioli, N. Maculan, *Algorithms for finding minimum fundamental cycle bases in graphs*
  - **D. van Dyck**, V. Fack, *To be or not to be Yutsis*
- Approximability and Complexity (31/5, 14:30, Conference Hall). Chair: F. Maffioli.
  - N. Ahuja, **A. Baltz**, B. Doerr, A. Srivastav, *Coloring graphs with minimal edge load*
  - U. Faigle, B. Fuchs, **B. Wienand**, *Covering graphs by colored stable sets*
  - F.V. Fomin, **D.M. Thilikos**, *A* 3-*approximation for the pathwidth of Halin graphs*
  - T. Doi, **T. Fujito**, *A primal-dual method for approximating tree cover with two weights*
- Being Hamiltonian (31/5, 17, Sala Marcolini). Chair: V. Deineko.
  - **S. Zhang**, B. Chen, R. Yu, *Heavy cycles in k-connected weighted graphs*
  - **A. Asratian**, *A new local condition for a graph to be Hamiltonian*
- Circulant Graphs (31/5, 14:30, Library). Chair: D. Räbiger.
  - **A. Uejima**, H. Ito, *Subdivision of the hierarchy of H-colorable graph classes by circulant graphs*
  - **V.E. Brimkov**, *Clique, chromatic, and Lovász numbers of certain circulant graphs*

- Cliques (1/6, Sala Marcolini, 14:30). Chair: F. Tardella.
  - **M. Cajkova**, V. Fack, *Clique algorithms for classifying substructures in generalized quadrangles*
  - **L.M. Torres**, *On cliques associated to 3-set packing problems*
  - **V.M.F. Dias**, C.M.H. de Figuereido, J.L. Szwarcfiter, *On the generation of bicliques of a graph*
- Colouring Problems I-II (31/5, 15:30, Library). Chairs: J. Žerovnik
  - B. Zmazek, **J. Žerovnik**, *Behzad-Vizing conjecture and Cartesian product graphs*
  - **T. Faik**, *About the b-continuity of graphs*
  - **C. Àlvarez**, M. Serna, *The proper interval colored graph problem for caterpillar trees*
  - **A. Vietri**, *The complexity of arc-colorings for directed hypergraphs*
- Data Structures and Retrieval (2/6, 10:30, Library). Chair: S. Werth.
  - B. Doerr, N. Hebbinghaus, **S. Werth**, *An improved discrepancy approach to declustering*
  - K. Fukuda, **S. Picozzi**, *Lexico-smallest representations, duality and matching polyhedra*
  - N. Betzler, R. Niedermeier, **J. Uhlmann**, *Tree decompositions of graphs: saving memory in dynamic programming*
- Exact Algorithms I-II (1/6, 10, Library). Chair: G. Righini.
  - **G. Nicosia**, A. Pacifici, *Exact algorithms for a discrete metric labelling problem*
  - **T. Brueggemann**, W. Kern, *An improved local search algorithm for 3-SAT*
  - M. Joswig, **M.E. Pfetsch**, *Computing optimal discrete morse functions*
  - R. Aringhieri, R. Cordone, *The multicommodity multilevel bottleneck assignment problem* (presented by **G. Righini**)
- Flows and Networks I-II (1/6, 10, Sala Marcolini). Chair: M. Pınar.
  - **D. Gijswijt**, *On a packet scheduling problem for smart antennas and polyhedra defined by circular-ones matrices*
  - A. Altın, E. Amaldi, P. Belotti, **M. Ç. Pınar**, *Virtual private network design under traffic uncertainty*
  - **M.-C. Costa**, A. Billionnet, *Multiway cut and integer flow problems in trees*
  - **S.G. Kolliopoulos**, *Minimum-cost single-source 2-splittable flow*
- Graph Theory I-II (1/6, 10, Conference Hall). Chairs: B. Fuchs.
  - J.-L. Fouquet, **J.-M. Vanherpe**, *On* $(P_5, \bar{P}_5)$*-sparse graphs and other families*
  - **V. Giakoumakis**, S. Olariu, *The set of prime extensions of a graph: the finite and the infinite case*
  - **E. Prisner**, *k-Pseudosnakes in n-dimensional hypercubes*
  - **M. Aïder**, *Extended distance-hereditary graphs*
- Numbers and Graphs (1/6, 16:30, Sala Marcolini). Chair: L. Stougie.
  - **N. Hebbinghaus**, *Discrepancy of sums of arithmetic progressions*
  - **A.N.M. Salman**, H.J. Broersma, *The Ramsey numbers of paths versus kipases*
- Polynomial Cases of Problems in NP (1/6, 14:30, Library). Chair: M. Queyranne.
  - **F. Carrabs**, R. Cerulli, M. Gentili, G. Parlato, *Minimum feedback vertex set on diamonds*
  - **P. Detti**, C. Meloni, M. Pranzo, *Minimum dominating trail set for two-terminal series parallel graphs*
  - W.Y.C. Chen, **X. Li**, C. Wang, X. Zhang, *Linear time algorithms to the minimum all-ones problem for unicyclic and bicyclic graphs*

- POLYNOMIAL PROBLEMS (1/6, 16:30, Conference Hall). Chair: E. Amaldi.
  - **M. Wattenhofer**, R. Wattenhofer, *Fast and Simple Algorithms for Weighted Perfect Matching*
  - **S. Nikolopoulos**, L. Palios, *On the strongly connected and biconnected components of the complement of graphs*
- TSP AND ROUTING I (1/6, 14:30, Conference Hall) – II (2/6, 10:30, Conference Hall). Chairs: H. Hamacher, R. Cordone.
  - **V. Deineko**, *New exponential neighbourhood for polynomially solvable TSPs*
  - G. Righini, **M. Salani**, *Dynamic programming algorithms for the elementary shortest path problem with resource constraints*
  - **D. Räbiger**, *Semi-preemptive routing on a line*
  - R. Aringhieri, **M. Bruglieri**, F. Malucelli, M. Nonato, *An asymmetric vehicle routing problem arising in the collection and disposal of special waste*
  - D. Cantone, **S. Faro**, Two-Levels-Greedy*: a generalization of Dijkstra's shortest path algorithm*
  - **P. Belotti**, F. Malucelli, *Network design with grooming constraints*

Workshop Program

There are six plenary sessions (a 15 minutes welcoming session, three 1-hour plenary lectures, the 1-hour "open problems" session and the 30 minutes closing session). Contributed talks are scheduled in groups of three parallel sessions held in different seminar rooms as specified below. Wide-interest topics will be presented in the Conference Hall, whilst other topics will be discussed in the smaller rooms (Sala Marcolini, the Library room).

31st May, AM	9-9:30	9:30-10	10-10:30	10:30-11	**11-11:15**	**11:15-11:45**	**11:45-12:15**
Conference Hall				*Coffee*	Welcome	Plenary (M. Queyranne)	
Sala Marcolini							
Library							

31st May, PM	14:30-15	15-15:30	15:30-16	16-16:30	16:30-17	17-17:30	17:30-18
Conference Hall	Approximability and Complexity				*Coffee*	Advances in Optimization II	
Sala Marcolini	Advances in Optimization I					Being Hamiltonian	
Library	Circulant Graphs		Colouring Problems I			Colouring Problems II	

1st June, AM	**9-9:30**	**9:30-10**	10-10:30	10:30-11	11-11:30	11:30-12	12-12:30
Conference Hall	Plenary (L. Stougie)		Graph Theory I		*Coffee*	Graph Theory II	
Sala Marcolini			Flows and Networks I			Flows and Networks II	
Library			Exact Algorithms I			Exact Algorithms II	

1st June, PM	14:30-15	15-15:30	15:30-16	16-16:30	16:30-17	17-17:30	**17:30-18:30**
Conference Hall	TSP and Routing I			*Coffee*	Polynomial Problems		Open Problems
Sala Marcolini	Cliques				Numbers and Graphs		
Library	Polynomial Cases of Problems in NP				Applications of Local Search		

2nd June, AM	**9-9:30**	**9:30-10**	10-10:30	10:30-11	11-11:30	11:30-12	**12-12:30**
Conference Hall	Plenary (H. Hamacher)		*Coffee*	TSP and Routing II			Closing
Sala Marcolini				Algebraic Structures			
Library				Data Structures and Retrieval			

- Lunch will take place at 13:00.
- Dinner will take place at 19:30.
- On Tuesday, 1st June 2004, at 21:00, there will be a piano concert in the Conference Hall. The pianist Massimo Bianchi will play the following program:
  - · F. Chopin, *Andante spianato e Polacca brillante, op. 22*
  - · F. Liszt, *Scherzo e Marcia*
  - · Beethoven/Liszt, *Piano transcription of Symphony n. 7, op. 92*:
    Poco sostenuto-Vivace — Allegretto — Presto — Allegro con brio
- Lunch will also be served at 13:00 on Wednesday 2nd June after the closing session.

Invited Talks

Decomposition of consecutive-1 matrices and applications

Horst W. Hamacher [1]

University of Kaiserslautern, Germany

Any integer matrix A can be decomposed into a non-negative linear combination of $(0, 1)$ matrices with the consecutive-1 property (either in rows or in columns). In this talk we will review results on how to do this decomposition with various objectives to be minimized such as:

- sum of coefficients
- number of consecutive-1 matrices to be used

Moreover a sequencing problem of the consecutive-1 matrices is considered. As applications we discuss the modulation of radiation in cancer therapy and the design of stops in public transportation systems. The presentation will be based on joint work with Ahuja, Baatar, Boland, Lenzen, Liebers, Schöbel and Wagner.

[1] E-mail: `hamacher@mathematik.uni-kl.de`.

Submodular function minimization in $\mathbb{Z}^n$ and searching in Monge arrays

Maurice Queyranne [1]

Sauder School of Business, University of British Columbia, Vancouver, B.C., Canada
Laboratoire Leibniz-IMAG, Grenoble, France

We establish and use connections among the problem of searching in a multidimensional Monge array, minimizing a submodular function on a sublattice of the integer lattice $\mathbb{Z}^n$, and Boolean submodular function minimization. In particular, we obtain the first algorithm for minimizing a submodular function on a sublattice S of $\mathbb{Z}^n$ with time complexity polynomial in the length of a maximal chain of S, and we use it to construct the first polynomial algorithm (regarded as a "major improvement" by Aggarwal and Park) for the problem of searching in a multidimensional Monge array.

We also present some results on the extension of submodular and modular functions, correcting an extension proposed by Lovasz, and we show that every modular function on a sublattice of $\mathbb{Z}^n$ is separable.

Finally, we describe a penalization approach to the problem of minimizing submodular functions on a sublattice of $\mathbb{Z}^n$ or of the Boolean hypercube.

The presentation is based on joint work with Fabio Tardella.

[1] E-mail: `Maurice.Queyranne@sauder.ubc.ca`.

Polynomial solvability of Mader's edge-disjoint paths problem

Leen Stougie [1]

Department of Mathematics and Computer Science, TU Eindhoven, The Nederlands
CWI, Amsterdam, The Nederlands

In 1978 Mader proved a minmax relation between the number of vertex and edge disjoint paths in a graph between prespecified *terminal* vertices of the graph and the size of a cut induced by a partition of the graph. His theorem generalizes a fundamental theorem of Menger from 1927. Menger's theorem in itself generalized the famous König's theorem, which is at the basis of matching theory, and it also implies the max flow-min cut theorem for network flows.

Mader did not give an algorithm for finding the maximum number of edge or vertex disjoint terminal paths. It was Lovasz in 1980 who showed that this problem can be solved in polynomial time using a technical proof based on matroid matching theory. Recently, Judith Keijsper, Rudi Pendavingh and I gave an easy proof for the edge-disjoint version through an integer linear programming formulation of the problem. We show that Mader's result implies strong duality between an integer solution of both the primal and the the dual of the linear programming relaxation of this problem. The proof also shows that a weighted version of the problem is polynomially solvable, which was not known before.

[1] E-mail: `leen@win.tue.nl`.

Contributed Papers

Coloring Graphs with Minimal Edge Load

Nitin Ahuja[1], Andreas Baltz[2], Benjamin Doerr[3], Anand Srivastav[4]

Mathematisches Seminar, Bereich II, Christian-Albrechts-Universität zu Kiel, Christian-Albrechts-Platz 4, 24118 Kiel, Germany

Abstract

The *load* of a coloring $\varphi : V \to \{\text{red, blue}\}$ for a given graph $G = (V, E)$ is a pair $L_\varphi = (r_\varphi, b_\varphi)$, where r_φ is the number of edges with at least one red end-vertex and b_φ is the number of edges with at least one blue end-vertex. Our aim is to find a coloring φ such that $l_\varphi := \max\{r_\varphi, b_\varphi\}$ is minimized. We show that this problem is NP-complete. For trees, we give a polynomial time algorithm computing an optimal solution. This has load at most $m/2 + \Delta \log_2 n$, where m and n denote the number of edges and vertices respectively. For arbitrary graphs, a coloring with load at most $\frac{3}{4}m + O(\sqrt{\Delta m})$ can be found in deterministic polynomial time using a derandomized version of Azuma's martingale inequality. This bound cannot be improved in general: almost all graphs have to be colored with load at least $\frac{3}{4}m - \sqrt{3mn}$.

Key words: graph coloring, graph partitioning

1 Introduction

Let $G = (V, E)$ be a graph. For a *coloring* $\varphi : V \to \{\text{red, blue}\}$ we define the *load* of φ by $L_\varphi := (r_\varphi, b_\varphi)$, where r_φ counts the number of edges incident with at least one red vertex, and b_φ is the number of edges incident with at least one blue vertex. The aim of the *Minimum Load Coloring Problem (MLCP)* is to find a coloring φ such that $l_\varphi := \max\{r_\varphi, b_\varphi\}$ is minimized. To our knowledge there exists no prior literature on this particular problem. A generalization to hypergraphs was regarded by Ageev et al. [1]. They study scheduling aspects of an optical communication network with n nodes $V = \{v_1, \ldots, v_n\}$ and n hyperedges $E = \{E_1, \ldots, E_n\}$. Node v_i wants to send packets of data to a set $E_i \subseteq V$ of other nodes. Given a set $W = \{w_1, \ldots, w_k\}$ of k available wavelength, the aim is to find an assignment $\varphi : V \to W$ of wavelengths to nodes such that the maximum load (number of packets) on any wavelength is minimized.

[1] E-mail: `nia@numerik.uni-kiel.de`.
[2] E-mail: `aba@numerik.uni-kiel.de`.
[3] E-mail: `bed@numerik.uni-kiel.de`.
[4] E-mail: `asr@numerik.uni-kiel.de`.

Notation: Throughout the paper let n denote the number of vertices of the graph under consideration, m the number of edges and Δ the maximum vertex degree. We put $l(G)$ to be the minimum (hence optimal) load among all vertex colorings of G.

2 *NP*–Completeness

A reduction to MinBisection shows that MLCP is NP-complete.

Theorem 1 *MLCP $\in NPC$.*

Proof.[Sketch] Let $G = (V, E)$ be an instance of MinBisection. We construct an instance $G' = (V', E')$ of MLCP by adding two cliques $C_1 = (V_1, E_1)$, $C_2 = (V_2, E_2)$, $|V_1| = |V_2| = 2m + 2$, $E_1 := \mathcal{P}_2(V_1)$, $E_2 := \mathcal{P}_2(V_2)$, a set of edges $\bar{E} := \{\{v, v'\} \mid v \in V,\ v' \in V_1 \cup V_2\}$ connecting each $v \in V$ to each clique vertex, and a matching $M = (V_3, E_3)$ of size m. Moreover, we define another instance G'' of MLCP by adding one more free matching edge $e'' = \{v_1'', v_2''\}$ to G'.

We claim that optimal solutions to MLCP on G' and G'' determine a minimum bisection in G and vice versa. Since MinBisection is NP-complete (cf. Karpinski [4]), and, obviously, MLCP $\in NP$, this implies MLCP $\in NPC$. Let OPT_B be the size of a minimum bisection of G. It is enough to show that $l(G') = \begin{cases} \frac{|E'|}{2} + \frac{\mathrm{OPT}_B}{2} + n(m+1) & \text{if } \mathrm{OPT}_B \text{ is even} \\ \frac{|E'|}{2} + \frac{\mathrm{OPT}_B}{2} + n(m+1) + \frac{1}{2} & \text{otherwise,} \end{cases}$

and $l(G'') = \begin{cases} \frac{|E'|}{2} + \frac{\mathrm{OPT}_B}{2} + n(m+1) + \frac{1}{2} & \text{if } \mathrm{OPT}_B \text{ is odd} \\ \frac{|E'|}{2} + \frac{\mathrm{OPT}_B}{2} + n(m+1) + 1 & \text{otherwise.} \end{cases}$

This implies $\mathrm{OPT}_B = \begin{cases} 2l(G') - |E'| - 2n(m+1) & \text{if } l(G') < l(G'') \\ 2l(G') - |E'| - 2n(m+1) - 1 & \text{otherwise.} \end{cases}$

The first step of the proof is to show that each optimal coloring φ has to color G' such that C_1 and C_2 are monochromatic with different colors, and V contains as many red as blue vertices. This yields $l_\varphi \geq \frac{|E'|}{2} + \frac{\mathrm{OPT}_B}{2} + n(m+1)$.

On the other hand, we may use the edges in E_3 to balance the number of monochromatic edges in both colors (here the parity of $|E_3|$ is important). Thus only the number of bichromatic edges is important. This proves the claimed bounds for $l(G')$ and $l(G'')$. □

3 Bounds and Algorithms for Trees

For trees, we prove the bound $l(G) \leq \frac{n-1}{2} + \Delta \log_2 n$. The key to this is the following more general lemma.

Lemma 1 *Given a tree $G = (V, E)$, $|V| = n$, and $p_1, p_2 \in \mathbb{N}$ with $p_1 + p_2 = n - 1$, there is a red-blue coloring of V such that at least $p_1 + 1 - \Delta \log_2 n$ edges are monochromatic red and at least $p_2 + 1 - \Delta \log_2 n$ are monochromatic blue.*

From the lemma, we easily deduce the following.

Theorem 2 *Let G be a tree. Then $l(G) \leq \frac{m}{2} + \Delta \log_2 n$.*

The proof of Lemma 1 uses an inductive construction. Thus, there is an efficient algorithm for computing colorings with load at most $\frac{m}{2} + \Delta \log_2 n$. However, it is also possible to compute optimal colorings for trees efficiently.

Theorem 3 *On trees, MLCP can be solved in time $O(n^3)$.*

Proof.[Sketch] Let G be a tree. We think of G as a *directed* tree with an arbitrary root a at level 0, the successors $N(a) := \{v \in V \mid (a, v) \in E\}$ of a at level 1, etc. For each $v \in V$ we denote by T_v the *induced subtree* of G rooted in v. We define for each *arbitrary* subtree G' of G with root a',

$$\mathcal{L}_{G'} := \{(r, b) \mid (r, b) = L_\varphi \text{ for some coloring } \varphi \text{ of } G' \text{ with } \varphi(a') = \text{red}\},$$

the set of possible loads for G'. Suppose, we can efficiently compute $\mathcal{L}_G$. Since $|\mathcal{L}_G| \leq (n+1)^2$, we can also efficiently find the maximum load $l(G)$ of an optimal coloring by inspecting all $(r, b) \in \mathcal{L}_G$ and selecting the one with smallest maximum component. It is easy to see that $\mathcal{L}_G$ can be determined in polynomial time by iteratively computing $\mathcal{L}_{T_v}$ for all $v \in V$ in *reverse breadth first order.* The iteration is based on two operations: consider a subtree G' of G with root $a' \neq a$, $v \in V$ with $(v, a') \in E$, and the tree $v + G' := (V(G') \cup \{v\}, E(G') \cup \{(v, a')\})$ obtained by appending the edge (v, a') to G'. We define

$$v + \mathcal{L}_{G'} := \{(r + 1, b) \mid (r, b) \in \mathcal{L}_{G'}\} \cup \{(b + 1, r + 1) \mid (r, b) \in \mathcal{L}_{G'}\} \qquad (1)$$

For two subtrees G'_1, G'_2 of G that intersect only in their joint root a', let $G'_1 + G'_2 := (V(G'_1) \cup V(G'_2), E(G'_1) \cup E(G'_2))$ be the composite tree. We define

$$\mathcal{L}_{G'_1} + \mathcal{L}_{G'_2} := \{(r_1 + r_2, b_1 + b_2) \mid (r_1, b_1) \in \mathcal{L}_{G'_1}, (r_2, b_2) \in \mathcal{L}_{G'_2}\}. \qquad (2)$$

It is straightforward to prove that for all subtrees $G' = (V', E')$ of G with root a' and all $v \in V$ with $(v, a') \in E$, $\mathcal{L}_{v+G'} = v + \mathcal{L}_{G'}$. Moreover, for all subtrees $G'_1 = (V'_1, E'_1), G'_2 = (V'_2, E'_2)$ intersecting only in their joint root a', $\mathcal{L}_{G'_1+G'_2} = \mathcal{L}_{G'_1} + \mathcal{L}_{G'_2}$. We conclude that $\mathcal{L}_{T_v} = \sum_{v' \in N(v)} \mathcal{L}_{v+T_{v'}} = \sum_{v' \in N(v)} v + \mathcal{L}_{T_{v'}}$ for all $v \in V$. Considering the complexity of the operations (1) and (2) we see that $\mathcal{L}_G$ can be recursively computed in $\sum_{v \in V} \deg(v) \cdot O(n^4 + n^2) = O(n^5)$ steps.

We can reduce this time to $O(n^3)$ by considering only "relevant" loads, $\mathcal{R}_{G'} := \{(r, b) \mid (r, b) \in \mathcal{L}_{G'}, b = \min\{b' \mid (r, b') \in \mathcal{L}_{G'}\}\}$. $\mathcal{R}_G$ can be computed iteratively via operations similar to (1) and (2) that are performed on $\mathcal{R}_{G'}$ instead of $\mathcal{L}_{G'}$ and thus require only $O(n)$ and $O(n^2)$ steps, respectively. This yields the desired $O(n^3)$ bound.

The iterative procedure to compute the optimal load can be easily modified to actually compute an optimal coloring. □

4 Approximation Algorithms for Arbitrary Graphs

Let us first observe that the load of random colorings is less than $\frac{3}{4}m + O(\sqrt{\Delta m})$ with high probability. Since $\frac{1}{2}m$ is a trivial lower bound for l_φ, we obtain a $(1.5+\varepsilon)$–approximation algorithm if $\Delta = o(m)$. We will use the following Martingale inequality that can be found in McDiarmid [5]. It is an application of the well known inequality of Azuma [2].

Lemma 2 *Let $X_1, \ldots, X_n$ be independent random variables taking values in some sets $A_1, \ldots, A_n$. Let $f : \prod_{i\in[n]} A_i \to \mathbb{R}$ such that $|f(x) - f(y)| \le c_i$ whenever x and y differ only in the i-th coordinate. Let $X = (X_1, \ldots, X_n)$ and $\mu = E(f(X))$. Then for any $\lambda \ge 0$,*

$$\mathbb{P}(f(X) - \mu \ge \lambda) \le \exp(-2\lambda^2 / \sum_{i=1}^{n} c_i^2).$$

Theorem 4 *There is a coloring φ such that $l_\varphi \le \frac{3}{4}m + \sqrt{(\ln 2)\Delta m}$.*
A random coloring satisfies $\mathbb{P}\Big(l_\varphi \ge \frac{3}{4}m + q\sqrt{(\ln 2)\Delta m}\Big) \le 2^{-q^2+1}$.

Proof. Let $\varphi : V \to \{\text{red}, \text{blue}\}$ such that $\mathbb{P}(\varphi(v) = \text{red}) = \frac{1}{2} = \mathbb{P}(\varphi(v) = \text{blue})$ independently for all $v \in V$. Clearly, if two colorings φ_1, φ_2 differ only in the color of some vertex $v \in V$, then $|r_{\varphi_1} - r_{\varphi_2}| \le \deg(v)$. We compute $E(r_\varphi) = \sum_{e\in E} \mathbb{P}(\exists v \in e : \varphi(v) = \text{red}) = \frac{3}{4}m$. Since $\sum_{v\in V} \deg(v)^2 \le \sum_{v\in V} \deg(v)\Delta = 2\Delta m$, for $\lambda = \sqrt{(\ln 2)\Delta m}$ we have $\mathbb{P}(r_\varphi > \frac{3}{4}m + \lambda) < \frac{1}{2}$. Thus with positive probability, both r_φ and b_φ are at most $\frac{3}{4}m + \lambda$. In particular, a coloring with $l_\varphi \le \frac{3}{4}m + \lambda$ exists. The second statement follows in a similar manner. □

Theorem 5 *A coloring φ such that $l_\varphi \le \frac{3}{4}m + \sqrt{(\ln 4)\Delta m}$ can be constructed in $O(n^3)$ time.*

For the proof we need a derandomized version (Theorem 6) of Azuma's martingale inequality.

Theorem 6 ([6]) *Let $f(X) = \sum_{i,j} \alpha_{ij} X_i X_j$ be a quadratic form satisfying the assumptions of Lemma 2. Let $\delta \in (0,1)$ with $2\exp(-2\lambda^2/\sum_{i=1}^n c_i^2) \le 1 - \delta$. We can construct a $X \in \{0,1\}^n$ with $|f(X) - \mathbb{E}(f(X))| \le \lambda$ in $O(n^3 \log(\delta^{-1}))$ time.*

Proof. [Sketch of Theorem 5] Let (a_{ij}) be the adjacency matrix of the graph $G = (V, E)$ under consideration. We identify a two-coloring $\varphi : V \to \{\text{blue,red}\}$ with $X \in \{0,1\}^n$. Let $r(X) = \sum_{i=1}^n \sum_{j=1}^n \frac{a_{ij}\, X_i\, X_j}{2} + \sum_{i=1}^n \sum_{j=1}^n a_{ij} X_i (1 - X_j)$, and $b(X) = \sum_{i=1}^n \sum_{j=1}^n \frac{a_{ij}\,(1-X_i)\,(1-X_j)}{2} + \sum_{i=1}^n \sum_{j=1}^n a_{ij}\, X_i\,(1 - X_j)$. Thus, $r_\varphi = r(X)$, $b_\varphi = b(X)$ and $l_\varphi = f(X) := \max\{r(X), b(X)\}$. Now, if we consider f, the maximum of two quadratic forms, then the result can be proved by using Lemma 2 and Theorem 6. □

The dependence on Δ cannot be avoided. This is shown by star graphs. If $\Delta = o(m)$, then the resulting bound of $(\frac{3}{4} + o(1))m$ cannot be improved in general, since, for the complete graph K_n, $l_\varphi \geq \frac{3}{8}n^2 - \frac{1}{4}n = (\frac{3}{4} + o(1))m$ for all colorings φ. In a sense, almost all graphs have a load of $(\frac{3}{4} - o(1))m$.

Theorem 7 *Let $m \geq 12n$. For a random multi-graph $G = (V, E)$, $|V| = n$ obtained by choosing m edges from $\binom{n}{2}$ independently with repetition, we have $l(G) \geq \frac{3}{4}m - \sqrt{3mn}$ with probability* $1 - 2^{-n}$.

References

[1] A.A. Ageev, A.V. Fishkin, A.V. Kononov, S.V. Sevastianov, *Open Block Scheduling in Optical Communication Networks.* Preprint(2003), 21 pages.

[2] K. Azuma, *Weighted sums of certain dependent variables.* Tohoku Math. Journal 3(1967), 357 - 367

[3] G. Even, J. Naor, S. Rao, B. Schieber, *Fast Approximate Graph Partitioning Algorithms.* SIAM J. Comput. 28(6)(1999), 2187 - 2214.

[4] M. Karpinksi, *Approximability of the Minimum Bisection Problem: An Algorithmic Challenge.* In Proc. 27th Int. Symp. on Math. Foundations of Comp. Sc., LNCS 2420(2002), 59 - 67.

[5] C. McDiarmid, *Concentration.* In Probabilistic Methods for Algorithmic Discrete Mathematics, Algorithms Combin. 16(1998), 195 - 248.

[6] A. Srivastav, P. Stangier, *On quadratic lattice approximations.* In Proc. 4th Int. Symp. on Algorithms and Computation, LNCS 762(1993), 176 - 184.

Extended Distance-Hereditary Graphs

Méziane Aïder [1]

Faculté de Mathématiques, USTHB BP 32 El Alia, 16 111 Bab Ezzouar, Alger, Algeria

Extended abstract

We shall only consider finite, simple loopless, undirected connected graphs $G = (V, E)$, where V is the vertex set and E is the edge set. For vertices u and v in G, $P_G(u, v)$ denotes a shortest path in G from u to v. Its length will be denoted $d_G(u, v)$ (if no ambiguity can arise, it will not be referred to the graph G in our notations).

Following the notations used in [3], a graph $G = (V, E)$ is $(k, +)$-distance-hereditary if for all connected induced subgraphs $H = (Y, F)$ of G, we have :

$$\forall u, v \in Y, d_H(u, v) \leq d_G(u, v) + k.$$

Note that other notations are also used to designate thses graphs [1].

Obviously, we have the following :

Proposition 1 *If G is a $(k, +)$-distance-hereditary graph, then G is a $(k', +)$-distance-hereditary graph, for all $k' \geq k$.*

It is easy to see that $(0, +)$-distance-hereditary graphs are just the distance-hereditary graphs [2] and [5], and $(1, +)$-distance-hereditary graphs are the almost distance-hereditary graphs [1].

All these classes of graphs have been characterized in terms of forbidden induced subgraphs.

A C_n-configuration is just an induced C_n with all its eventual chords are incident to a same vertex.

Theorem 1 *[2] A graph is a $(0, +)$-distance-hereditary graph if and only if it neither contains C_5-configurations nor C_n-configurations, for $n \geq 5$, as induced subgraphs.*

Theorem 2 *[1] A graph G is $(1, +)$-distance-hereditary if and only if G neither contains a $2C_5$-configuration, nor a C_n-configuration, for $n \geq 6$, as induced subgraphs.*

[1] E-mail: `meziane_aider@yahoo.fr`.

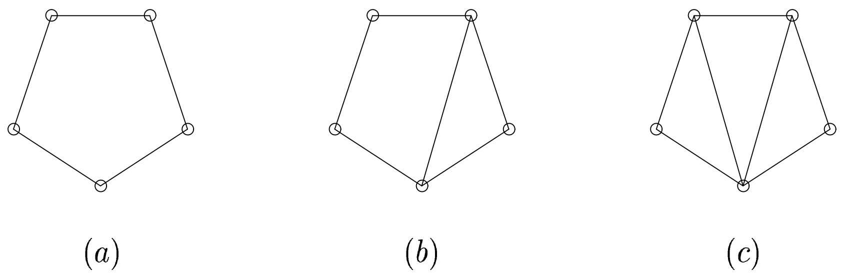


Figure 1. C_5-configurations

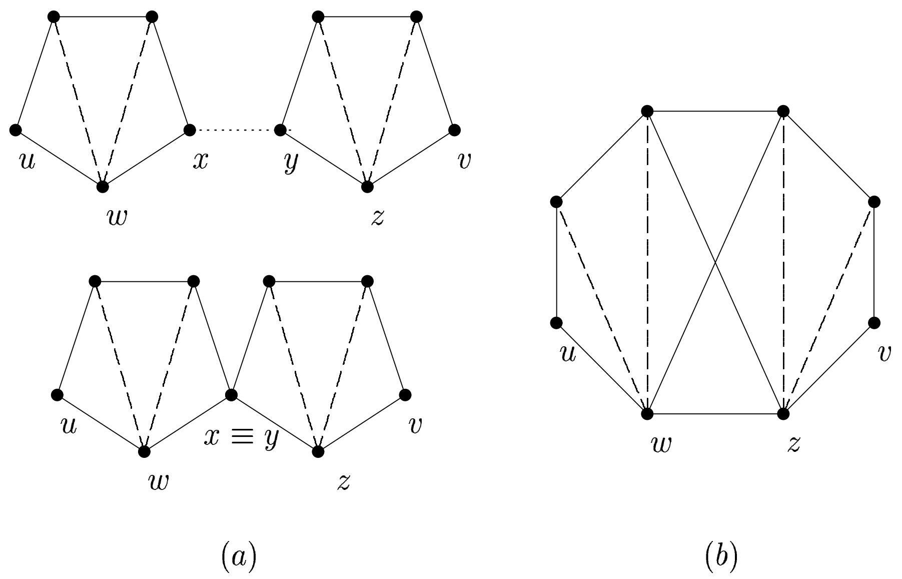


Figure 2. $2C_5$-configurations

If one looks carrefully to the $2C_5$-configuration of type (b), he can observe it consists of a split composition of two C_5-configurations.

In this talk, we are interested in a characterization of $(2,+)$-distance-hereditary graphs in terms of forbidden induced subgraphs.

For example, minimal forbidden induced subgraphs are :

- C_n-configurations, with $n \geq 7$;
- configurations obtained by combining (in the same way that in the definition of $2C_5$-configurations of type (a)) three minimal forbidden subgraphs of $(0,+)$-distance-hereditary (i.e. C_5-configurations) ;
- configurations obtained by combining a minimal forbidden subgraph of $(0,+)$-distance-hereditary (i.e. a C_5-configuration) with a minimal forbidden subgraph of $(1,+)$-distance-hereditary (i.e. a C_6-configuration or a $2C_5$-configuration)
- configurations obtained by combining some pairs of minimal forbidden subgraphs of $(1,+)$-distance-hereditary (i.e. C_5-configurations) ;

Our goal in this talk is to describe all these minimal forbidden induced subgraphs and to obtain a minimal characterization of these graphs.

References

[1] M. Aïder, Almost distance-hereditary graphs, Discrete Math., 242 (2002), 1-16.

[2] H. J. Bandelt and H. M. Mulder, Distance-Hereditary Graphs, J. of Combin. Theory Ser. B 41 (1986), 182-208.

[3] S. Cicerone, G. D'Ermiliis and G. Di Stefano, $(k,+)$-Distance- Hereditary Graphs, in : 27th International Workshop on Graph Theoretic Concepts in Computer Science (WG'01), Lecture Notes in Computer Science 2204 (2001), 66-77.

[4] S. Cicerone and G. Di Stefano, Graphs with bounded induced distance, Discrete Appl. Math., 108 (2001), 3-21.

[5] E. Howorka, A Characterization of Distance-Hereditary Graphs, Quart. J. Math. Oxford Ser. 28 (1977), 417-420.

Virtual Private Network Design Under Traffic Uncertainty

A. Altın [a] [1], E. Amaldi [b] [2], P. Belotti [b] [3], M. Ç. Pınar [a] [4]

[a] *Bilkent University, Department of Industrial Engineering, 06800, Bilkent, Ankara, Turkey*

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

Abstract

We propose different formulations as well as efficient solution approaches for the VPN design problem under traffic uncertainty with symmetric bandwidths.

Key words: VPN design, traffic uncertainty.

1 Introduction

A Virtual Private Network (VPN) service is similar to a private network service since it enables a group of nodes over a large underlying network to communicate with each other using an already available physical network like the Internet.

In this paper we deal with the polyhedral model of Ben-Ameur and Kerivin [1], which allows the traffic vector to belong to a polytope defined by some customer specific constraints. We offer three different formulations for the VPN design problem where the solution is allowed to be an arbitrary subgraph. We propose a Column Generation and a Cutting Plane Algorithm to solve these formulations efficiently.

We assume that we are given an undirected network $G = (V, E)$ and a set of VPN sites $W \subseteq V$. Each edge $\{i, j\} \in E$ is assigned a unit capacity reservation cost $c_{ij} > 0$. The aggregate traffic inflow and outflow bandwidths for each terminal $s \in W$ is symmetric, i.e. $b_s^+ = b_s^- = b_s \; \forall s \in W$. A traffic demand vector is $\vec{d} = (d_{st})_{s \in W, t \in W \setminus \{s\}}$. The set of demand pairs is $Q = \{(s,t) : s, t \in W, s \neq t\}$ and we are given a matrix $A \in \Re^{h*|Q|}$ where h is the number of constraints defining the traffic polytope $D = \left\{\vec{d} : A\vec{d} \leq \vec{a}\ ,\ \vec{a} \in \Re^h\right\}$. Each terminal $s \in W$ is required to route its traffic to site $t \in W \setminus \{s\}$ unsplittably on a single

[1] E-mail: `aysegula@bilkent.edu.tr`.
[2] E-mail: `amaldi@elet.polimi.it`.
[3] E-mail: `belotti@elet.polimi.it`.
[4] E-mail: `mustafap@bilkent.edu.tr`.

path P_{st} and the final solution, i.e., $P = \bigcup_{(s,t)\in Q} P_{st}$, is allowed to be an arbitrary subgraph of $G = (V, E)$. The problem is to find a least cost capacity installation so as to satisfy all possible traffic demands known to lie in the polytopic set D.

2 The Polyhedral Flow Formulation

In this section we present the following mixed IP formulation of the problem using the binary flow variable y_{ij}^{st}, which is 1 if the directed arc (i, j) is contained in the path going from terminal s to terminal t.

$$\min \sum_{\{i,j\}\in E} c_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j:\{i,j\}\in E} \left(y_{ij}^{st} - y_{ji}^{st}\right) = \begin{cases} 1 & i = s \\ -1 & i = t \\ 0 & \text{o.w.} \end{cases} \quad \forall i \in V, (s,t) \in Q \tag{2}$$

$$\sum_{(s,t)\in Q} d_{st}(y_{ij}^{st} + y_{ji}^{st}) \le x_{ij} \qquad \forall \{i,j\} \in E \tag{3}$$

$$A\vec{d} \le \vec{a} \tag{4}$$

$$y_{ij}^{st} \in \{0,1\} \qquad \forall \{i,j\} \in E, (s,t) \in Q$$

$$x_{ij} \ge 0 \qquad \forall \{i,j\} \in E$$

where the constraints (2) are the flow constraints. Constraint set (3) defines the amount of capacity reserved on the edge $\{i, j\}$ considering all feasible traffic scenarios defined by (4). Note that the constraints (3) are nonlinear. Our contribution at this point is to linearize these constraints so as to obtain the following compact mixed-integer formulation.

$$\min \sum_{\{i,j\}\in E} c_{ij}x_{ij} \tag{5}$$

$$\text{s.t.} \sum_{j:\{i,j\}\in E} \left(y_{ij}^{st} - y_{ji}^{st}\right) = \begin{cases} 1 & i = s \\ -1 & i = t \\ 0 & \text{o.w.} \end{cases} \quad \forall i \in V, (s,t) \in Q \tag{6}$$

$$\vec{w^{ij}}^T \vec{a} \le x_{ij} \qquad \forall \{i,j\} \in E \tag{7}$$

$$\vec{w^{ij}}^T A^T \ge \vec{y_{ij}} + \vec{y_{ji}} \qquad \forall \{i,j\} \in E \tag{8}$$

$$x_{ij} \ge 0, \vec{w^{ij}} \ge 0 \qquad \forall \{i,j\} \in E \tag{9}$$

$$y_{ij}^{st} \in \{0,1\} \qquad \forall \{i,j\} \in E, (s,t) \in Q \tag{10}$$

where $\vec{w^{ij}}$ is the vector of dual variables.

3 The Polyhedral Path Formulation

In this case we have the binary variable z_p, which is 1 if traffic is routed through path p in the optimal solution. Then our path formulation is

$$
\begin{aligned}
&\min \sum_{\{i,j\}\in E} c_{ij}x_{ij} && && (11)\\
\text{s.t.}\quad & \sum_{p\in P_{st}} z_p \geq 1 && \forall\,(s,t)\in Q && (12)\\
& \sum_{(s,t)\in Q} d_{st} \sum_{p\in P_{st}\cap P_{\{i,j\}}} z_p \leq x_{ij} && \forall\,\{i,j\}\in E && (13)\\
& A\vec{d} \leq \vec{a} && && (14)\\
& z_p \in \{0,1\} && \forall\,(s,t)\in Q, p\in P_{st} && (15)\\
& x_{ij} \geq 0 && \forall\,\{i,j\}\in E && (16)
\end{aligned}
$$

where (12) ensures that the demand pair $(s,t)\in Q$ communicates and (13) defines the capacity reservation on edge $\{i,j\}$ considering all possible traffic scenarios defined as in (14). Note that type (13) constraints are nonlinear and can be linearized judiciously to obtain

$$
\begin{aligned}
&\min \sum_{\{i,j\}\in E} c_{ij} \sum_{k=1}^{h} a_k w_k^{ij} && && (17)\\
\text{s.t.}\quad & \sum_{p\in P_{st}} z_p \geq 1 && \forall\,(s,t)\in Q && (18)\\
& \sum_{k=1}^{h} a_{k,st} w_k^{ij} \geq \sum_{p\in P_{st}\cap P_{\{i,j\}}} z_p && \forall\,(s,t)\in Q, \{i,j\}\in E && (19)\\
& z_p \in \{0,1\} && \forall\,(s,t)\in Q, p\in P_{st} && (20)\\
& \vec{w^{ij}} \geq 0 && \forall\,\{i,j\}\in E && (21)
\end{aligned}
$$

We propose to use a column generation algorithm to solve the path formulation,which can be summarized as follows:

- *Step 0* Let the inital set of paths include the pairwise shortest paths.
- *Step 1* Solve the path formulation with the current set of paths.
- *Step 2* For each $(s,t)\in Q$ if you find a path p such that $\sigma_{st} - \sum_{\{i,j\}\in p} \pi_{ij}^{st} > 0$, the add it to te current set of paths.
- *Step 3* Go to step 1 if new paths are added in Step 2. Otherwise stop!

4 The Polyhedral Cut Formulation

In this case we require connectivity over all cuts. Then the polyhedral cut formulation is as given below.

$$\min \sum_{\{i,j\}\in E} c_{ij} \vec{a}^T \vec{w^{ij}} \tag{22}$$

$$\text{s.t.} \sum_{\{i,j\}\in\delta(W)} \vec{w^{ij}} A_{st} \geq 1 \, \forall \, (s,t) \in Q \tag{23}$$

$$\vec{w^{ij}} \geq 0 \qquad \forall \, \{i,j\} \in E \tag{24}$$

where A_{st} denotes the column of A corresponding to the demand pair $(s,t) \in Q$. We propose to use a cutting plane algorithm to solve the above problem, which can be summarized as follows.

- *Step 0* Set the current cut to empty set.
- *Step 1* Solve the above problem with the current cut set.
- *Step 2* For each demand pair $(s,t) \in Q$ use the max flow-min cut theorem to determine the violated cut inequalities of type (23). If you find such an edge, then add that edge to the current cut.
- *Step3* If the cut set is updated, then go to Step 1, otherwise stop!

References

[1] Ben-Ameur, W., Kerivin, H., 2002. Routing of Uncertain Demands. *submitted.*

The Proper Interval Colored Graph problem for caterpillar trees

C. Àlvarez [1], M. Serna [2]

Dept. de Llenguatges i Sistemes Informàtics, UPC, Jordi Girona Salgado 1-3. 08034 Barcelona, Spain.

Abstract

This paper studies the computational complexity of the *Proper interval colored graph* problem (PICG), when the input graph is a colored caterpillar, parameterized by hair length. To prove our result we also study a graph layout problem the *Proper colored layout problem* (PCLP). We show a dichotomy result: the PICG and the PCLP are $\mathcal{NP}$-complete for colored caterpillars of hair length ≥ 2, while both problems are in P, for colored caterpillars of hair length < 2.

1 Introduction

A graph $G = (V, E)$ is called an *interval graph* if one can assign to each vertex $v \in V$ an interval on the real line I_v, in such a way that $(u, v) \in E \iff I_u \cap I_v \neq \emptyset$. When in addition for any pair of vertices $u, v \in V$ it holds that $I_u \not\subseteq I_v$, the graph is called a *proper interval graph*. Interval graphs have been studied intensively because of their wide applicability to practical problems [7]. Many efforts have been devoted to the study of problems in which one is asked to complete a graph or a colored graph into an interval or a proper interval graph as this kind of problems are used to model ambiguity in Physical Mapping or consistency in Temporal Reasoning [9]. In the colored versions of the above problems, the input is a graph together with a proper vertex coloring that uses k-colors, and the solution is a super-graph that, besides of being of the required type, is still properly colored by the given coloring. Most of those problems are known to be $\mathcal{NP}$-complete, the *Interval graph completion* [5] and the *Interval colored graph completion* (ICG) [4, 8]. In the case that the graph has degree bounded by a constant and it is further colored with k colors, there is a $O(n^{k-1})$ algorithm to solve the ICG [10]. For a fixed number of colors k, the problem is $\mathcal{NP}$-complete for $k \geq 4$ and in P for $k < 4$ [3]. In fact for 4 colors the problem is $\mathcal{NP}$-complete even for caterpillar trees [1]. Recall that a caterpillar with hairs of length at most h is formed by a chain, called the *backbone*. Each node in the backbone can be connected to several non intersecting paths of length at most h.

[1] E-mail: `alvarez@lsi.upc.es`.
[2] E-mail: `mjserna@lsi.upc.es`.

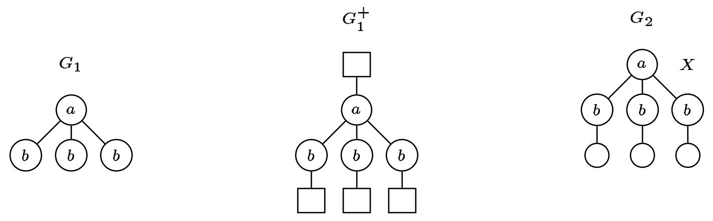


Figure 1. Some graphs

The parameterized version of the *Proper interval colored graph* (PICG) problem, with parameter the number of colors, is $W[1]$-hard, this implies the $\mathcal{NP}$-completeness of PICG and a polynomial time algorithm for constant number of colors [10, 11, 6]. We study the complexity of the PICG when the input graph is a colored caterpillar tree. We show its $\mathcal{NP}$-completeness for colored caterpillars of hair length ≥ 2, and provide a polynomial time algorithm for caterpillars of hair length < 2. To prove our result we reinforce the relationship between intervalizing problems and graph layout problems. Recall that given a graph $G = (V, E)$ with $|V| = n$ a *layout* φ of G is a one to one mapping $\varphi : V \longrightarrow [n]$. For a given layout φ, and any $1 \leqslant i < n$, let $V_i = \{v \mid \varphi(v) \leqslant i \text{ and } \exists u \; \varphi(u) > i \, (u, v) \in E\}$. For a given k-colored graph $(G = (V, E), \kappa)$, where κ is a proper k-coloring of G, a *colored layout* of (G, κ) is a layout φ of G such that for all $u \in V$ with $\varphi(u) > 1$, $\kappa(u) \notin \kappa(V_{\varphi(u)-1})$ and a *proper colored layout* of (G, κ) is a colored layout φ of G such that for all $u, v \in V$ and $x \in V$ with degree at least 2, if $(u, v) \in E$ and $\varphi(u) < \varphi(x) < \varphi(v)$ then it exists a vertex y such that $\varphi(v) < \varphi(y)$, $(x, y) \in E$. The *Colored layout problem* (CLP) asks whether a given k-colored graph has a proper colored layout. This problem is equivalent to the ICG problem [2]. We introduce another graph layout problem the *Proper colored layout* problem (PCLP) that asks whether a given a k-colored graph has a proper colored layout. The PCLP problem is not identical to the PICG problem, the graph G_1, given in figure 1, has a proper colored layout, but does not have any proper intervalization (a label inside a circle indicates a color, while a label outside a circle, if any, indicates a node name). However, we will show that the PICG problem can be formulated as an instance of the PCLP, for a particular graph class. Our main result is that the PICG and PCLP problems are $\mathcal{NP}$-complete for colored caterpillars of hair length ≥ 2 and in P for caterpillars of hair length 1 or 0. This contrasts with the fact that the *Interval graph completion* problem for trees is in P [12]. For the hardness results we provide a reduction from the *Multiprocessor Scheduling problem*, while the polynomial time results follows from a characterization in terms of forbidden subgraphs.

2 The reduction for the PICG problem

We start by establishing the relationship between the PICG and the PCLP. Given a k-colored graph $(G = (V, E), \kappa)$, a decoration of G is a new $k+1$-colored graph $(G^+ = (V', E'), \kappa')$ with $V' = V \cup V^+$ where $V^+ = \{u^+ \mid u \in V\}$, $E' = E \cup E^+$ where $E^+ = \{(u, u^+) \mid u \in V\}$, and for any u in V, $\kappa'(u) = \kappa(u)$ and $\kappa'(u^+) = c$, where c is a new color, therefore κ' is a $k+1$ coloring of G^+. We refer to G^+ as a *decorated graph.*

Theorem 1 $(G, \kappa) \in$ PICG *iff* $(G^+, \kappa') \in$ PCLP

Inspired in the schema used in [13] to show hardness for the bandwidth problem, we give a reduction from the *Multiprocessor Scheduling problem.* Recall that given a set of n tasks, having duration t_i, for $1 \leqslant i \leqslant n$, a deadline D and a set of m processors, determine whether

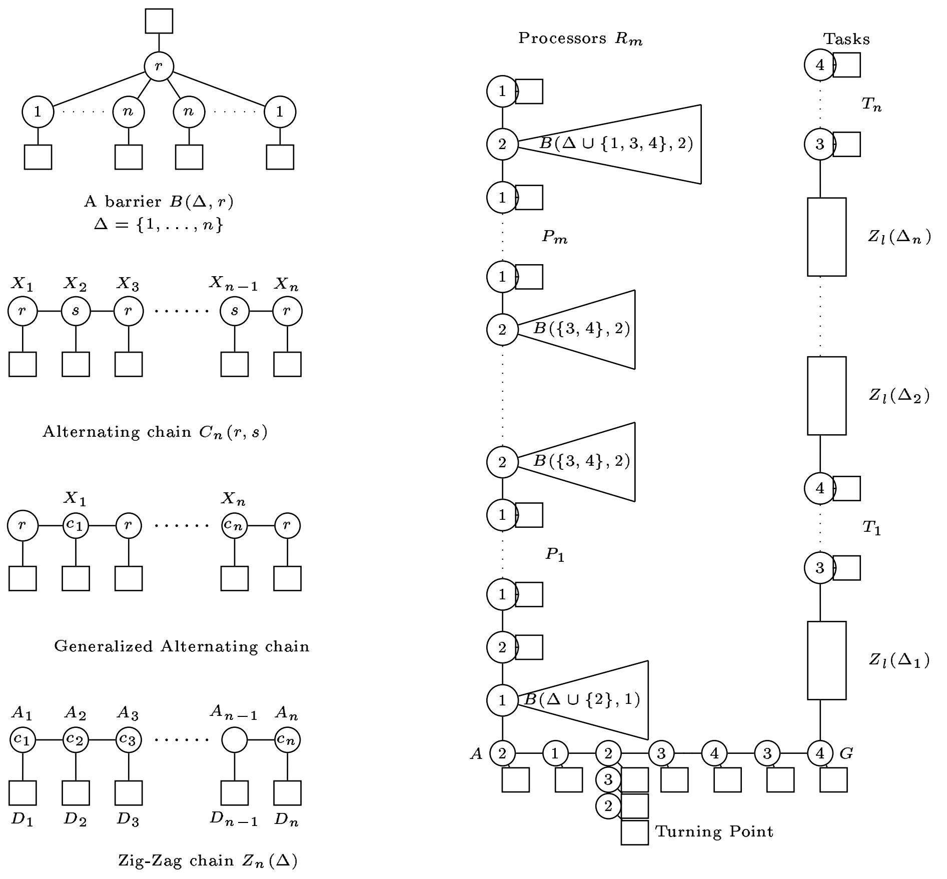

A barrier $B(\Delta, r)$
$\Delta = \{1, \dots, n\}$
Alternating chain $C_n(r, s)$
Generalized Alternating chain
Zig-Zag chain $Z_n(\Delta)$
Processors R_m
Tasks
$B(\Delta \cup \{1,3,4\}, 2)$
P_m
$B(\{3,4\}, 2)$
P_1
$B(\Delta \cup \{2\}, 1)$
T_n
$Z_l(\Delta_n)$
$Z_l(\Delta_2)$
T_1
$Z_l(\Delta_1)$
A
G
Turning Point

Figure 2. A sketch of the caterpillar $G(I)$ and some types of graphs used in its construction

the tasks can be assigned to processors so that all processors finish their work before the deadline D. Recall that the problem is $\mathcal{NP}$-complete in the strong sense, so we can assume that the duration of each task is polynomially bounded. We sketch the construction of a decorated caterpillar. Given an instance $I = (t_1, \dots, t_n, D, m)$ of the Multiprocessor Scheduling problem we construct the colored decorated caterpillar $G(I)$ (see figure 2) obtained by joining three different gadgets, one for the processors, one for the tasks, and a turning point. A processor is represented by an alternating chain, and the graph associated to the processors is formed by a series of barriers, separating the graphs corresponding to processors i and $i+1$, with two additional big barriers, one at the beginning and the other at the end. Each task is represented by an alternating chain. We join the graphs corresponding to two consecutive tasks with a zig-zag chain of length equal to the length of the backbone of the processors gadget. An additional zig-zag chain joins the first task with the node G of the turning point. We select a set of new different colors for each zig-zag chain (sets Δ_i). So, the set of colors is, in addition of $\{1, 2, 3, 4\}$, $\Delta = \cup_{i=1}^{n} \Delta_i$.

Theorem 2 *The instance $I = (t_1, \dots, t_n, D, m)$ of multiprocessor Scheduling has a solution iff the colored decorated caterpillar $G(I)$ has a proper colored layout. Therefore,* PICG *is $\mathcal{NP}$-complete for caterpillars with hair length ≥ 2.*

The previous reduction can be modified to show that the PCLP problem is $\mathcal{NP}$-complete for caterpillars with hair length at most 2. The changes affect the turning point and the task's gadget.

Theorem 3 PCLP *is $\mathcal{NP}$-complete for caterpillars with hair length 2.*

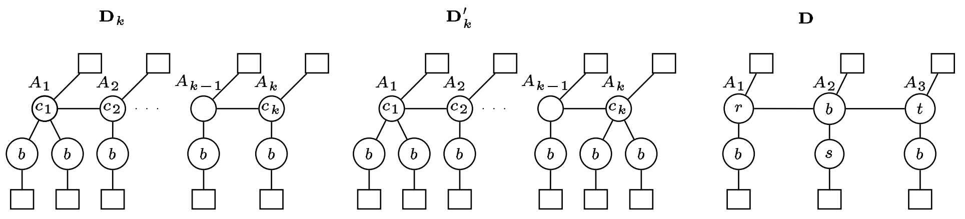


Figure 3. Some decorated caterpillars

3 Short haired Caterpillars

We consider now the case of caterpillars with hair length ≤ 1. The following technical lemma follows from the definitions.

Lemma 1 *If a k-colored graph $(G = (V, E), \kappa)$ has a proper colored super-graph that is a proper interval graph, then any of its subgraphs verify the same property.*

Based on the fact that the caterpillars D'_k given in Figure 3 and the caterpillar G_2 given in Figure 1 have no proper colored layout we show the following characterization.

Lemma 2 *A decorated colored caterpillar with hair length 1 has a proper colored layout iff it does not contain any instance of the subgraphs D'_k nor the subgraph G_2.*

We find polynomial time algorithm for checking the above property, and for the case of caterpillars with hair length 0.

Theorem 4 *The* PCLP *and the* PICG *problems are in P for caterpillars with hair length ≤ 1.*

Acknowledgements The authors thank the support of the FET Programme of the EU under contract number IST-2001-33116 (FLAGS) and the Spanish CICYT project TIC-2002-04498-C05-03

References

[1] C. Àlvarez, J. Díaz, and M. Serna. The hardness of intervalizing four colored caterpillars. *Discrete Mathematics* 235:245–253, 2001.

[2] C. Àlvarez, J. Díaz, and M. Serna. Intervalizing colored graphs is NP-complete for caterpillars with hair length 2. Technical Report LSI 98-9-R, Universitat Politècnica de Catalunya, 1998.

[3] H. Bodlaender, M. R. Fellows, and M. T. Hallet. Beyond NP-completeness for problems of bounded width: hardness for the W-hierarchy. In *ACM Symposium on Theory of Computing*, pages 449–458, 1995.

[4] M. R. Fellows, M. T. Hallet, and W. T. Wareham. DNA physical mapping: Three ways difficult. In T. Lengauer, editor, *Algorithms-ESA '93*, number 726 in Lecture Notes in Computer Science, pages 157–168. Springer-Verlag, September 1993.

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

[6] P. W. Goldberg, M. C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. *Journal of Computational Biology*, 2:139–152, 1995.

[7] M. C. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, New York, 1980.

[8] M. C. Golumbic, H. Kaplan, and R. Shamir. On the complexity of DNA physical mapping. *Advances in Applied Mathematics*, 15:251–261, 1994.

[9] M. C. Golumbic, H. Kaplan, and R. Shamir. Graph sandwich problems. *Journal of Algorithms*, 19:449–473, 1995.

[10] H. Kaplan and R. Shamir. Pathwidth, bandwidth and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing*, 25(3):540–561, June 1996.

[11] H. Kaplan, R. Shamir, and R. E. Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM Journal on Computing*, 28(5):1906–1922 (electronic), 1999.

[12] D. Kuo and G. J. Chang. The profile minimization problem in trees. *SIAM Journal on computing*, 23:71–81, Feb. 1994.

[13] B. Monien. The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 7(4):505–512, 1986.

Algorithms for finding minimum fundamental cycle bases in graphs

Edoardo Amaldi[1], Leo Liberti[2], Francesco Maffioli[3]

DEI, Politecnico di Milano, Piazza L. da Vinci 32, 20133 Milano, Italy

Nelson Maculan[4]

COPPE, Universidade Federal do Rio de Janeiro, P.O. Box 68511, 21941-972, Rio de Janeiro, Brazil

Abstract

We describe new heuristics for solving the problem of finding the fundamental cycle bases of minimum cost in a simple, undirected, biconnected graph G. Since each spanning tree of G is associated to a fundamental cycle basis, edge swaps are iteratively performed on the current spanning tree so as to improve the cost of the corresponding fundamental cycle basis. Furthermore, we establish graph-theoretical structural results that allow an efficient implementation of our algorithms.

1 Introduction

Let $G = (V, E)$ be a simple, undirected and biconnected graph with n nodes and m edges, weighted by a non-negative cost function $w : E \to \mathbb{R}^+$, which is extended to sets of edges in the natural way (if $F \subseteq E$, $w(F) = \sum_{e \in F} w(e)$). A set of cycles in the graph is a *cycle basis* if it is a basis of the cycle vector space. The cost of a set of cycles is the sum of the costs of all cycles in the set. Given any spanning tree of G with edge set $T \subseteq E$, the edges in T are called *branches* of the tree, and those in $E \backslash T$ are called the *chords* of G with respect to T. Any chord uniquely identifies a cycle consisting of the chord itself and the unique path in T connecting the two nodes incident on the chord. These $m - n + 1$ cycles are called *fundamental cycles* and they form a *Fundamental Cycle Basis* (FCB) of G with respect to T. It was shown that a cycle basis is fundamental if and only if each cycle in the basis contains at least one edge which is not contained in any other cycle in the basis [9]. Finding the Minimum Fundamental Cycle Basis (MIN FCB) of a graph is an $\mathcal{NP}$-hard problem [2]. Furthermore, it does not admit a polynomial-time approximation scheme unless $\mathcal{P} = \mathcal{NP}$; a $(4+\varepsilon)$-approximation algorithm was found for complete graphs, and a $2^{O(\sqrt{\log n \log \log n})}$-approximation algorithm for arbitrary graphs [7].

[1] E-mail: `amaldi@elet.polimi.it`.
[2] E-mail: `liberti@elet.polimi.it`.
[3] E-mail: `maffioli@elet.polimi.it`.
[4] E-mail: `maculan@cos.ufrj.br`.

Interest in minimum FCBs arises in a variety of application fields, such as electrical circuit testing [1], periodic timetable planning [6] and generating minimal perfect hash functions [3].

2 Edge-swapping local search and metaheuristics

Our local search for the MIN FCB problem is based on an iterative improvement of a current spanning tree, obtained by performing edge swaps. We start from an initial spanning tree grown by adding nodes to the tree in such a way that short fundamental cycles are completed early in the process (based on [8]). At each iteration, we identify the edge swap between branch and chord that leads to the largest decrease in FCB cost. This edge-swapping operation is inserted in a local search procedure.

Consider any given spanning tree T of G. For each branch $e \in T$, the *fundamental cut* of G induced by e is the edge set $\delta^e_T = \{\{u,v\} \in E \mid u \in S^e_T, v \in \bar{S}^e_T\}$, where S^e_T, $\bar{S}^e_T$ is the node partition induced by the removal of e from T. For any chord $f \in \delta^e_T$, let $\pi = (e,f)$ be the edge swap which consists in removing e while adding f to T. Denote by πT the resulting spanning tree. Now for each such edge swap π we calculate the cost difference Δ_π between the FCB of T and that of πT. Let Δ_{opt} be the largest such difference, and π_{opt} be the correspoding edge swap. The local search iteratively identifies π_{opt} and updates the current T with $\pi_{\text{opt}}T$ while π_{opt} is not the identity.

Applying an edge swap to a spanning tree may change the fundamental cycles and cut structure considerably. Hence, efficient procedures are needed to determine the cuts $\delta^e_{\pi T}$ for all $e \in \pi T$, and to compute Δ_π from the data at the previous iteration, namely from T, π and the cuts δ^e_T, for $e \in T$.

Some of the following structural properties are straightforward, others can be proved by careful case enumeration.

Efficient cut structure update:

- any edge swap $\pi = (e,f)$ applied to a spanning tree T, where $e \in T$ and $f \in \delta^e_T$, changes a cut δ^h_T if and only if $f \in \delta^h_T$;
- $\delta^h_{\pi T}$ can be determined by taking the symmetric difference $\delta^h_T \triangle \delta^e_T$ (see Figures 1-4 for a graphical sketch of the proof).

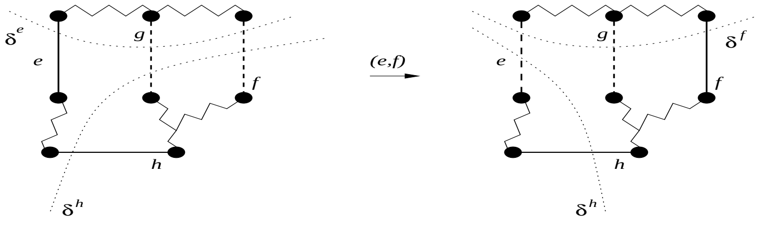


Figure 1. Let $g \in \delta^h \cap \delta^e$. Then $g \notin \pi(\delta^h)$.

Efficient cycle structure update (notation: γ^h_T is the unique fundamental cycle of G w.r.t. the chord h):

- if $h \notin \delta^e_T$, then γ^h_T is unchanged by π;

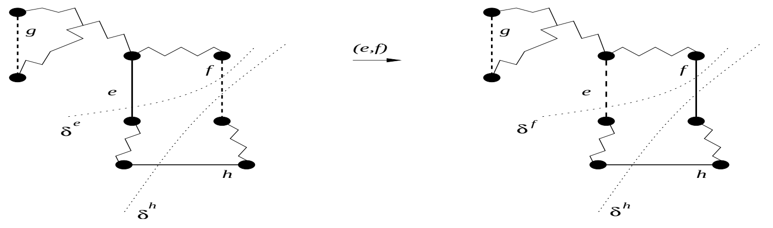


Figure 2. Let $g \notin \delta^h \cup \delta^e$. Then $g \notin \pi(\delta^h)$.

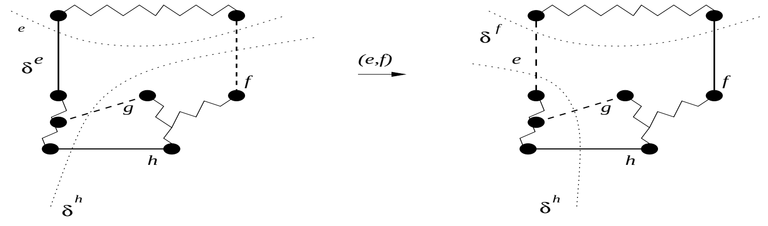


Figure 3. Let $g \in \delta^h$ and $g \notin \delta^e$. Then $g \in \pi(\delta^h)$.

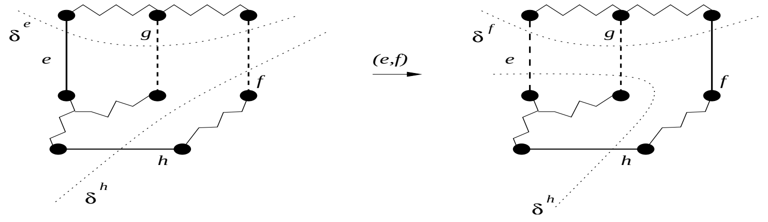


Figure 4. Let $g \notin \delta^h$ and $g \in \delta^e$. Then $g \in \pi(\delta^h)$.

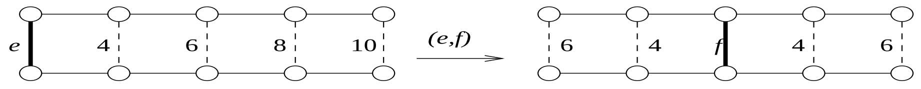


Figure 5. All edge weights are equal to 1 and the numbers indicated on the chords correspond to the costs of the corresponding fundamental cycles. The cut on the left has a difference between cheapest and most expensive cycle of $10 - 4 = 6$; after the edge swap the difference is $6 - 4 = 2$.

- if $h \in \delta^e_T$, then $\gamma^h_{\pi T}$ can be determined by taking the symmetric difference $\gamma^h_T \triangle \gamma^f_T$.

It can be verified that the complexity of identifying the best edge swap π_{opt} and applying it to T to obtain πT is $O(m^2 n^2)$.

The implementation of the local search algorithm described above is computationally intensive. For large-scale problems, we would like to test the edge swap only for a small subset of pairs e, f while minimizing the chances of missing pairs which yield large cost decreases. A good strategy is to focus on branches inducing fundamental cuts whose edges define fundamental cycles with "unbalanced" costs, i.e., with a large difference between the cheapest and the most expensive of those fundamental cycles. See Fig. 5 for a simple example.

To try to escape from local minima, we have included the above edge-swap move within two well-known metaheuristics: variable neighbourhood search (VNS) [4] and tabu search (TS) [5]. We used a basic implementation of VNS. Our implementation of the Tabu search, on the other hand, is a blend of classic TS and VNS. If π_{opt} is the identity, an edge swap that worsens the FCB cost is applied to the current solution and inserted in a tabu list. If all possible edge swaps are tabu or a pre-determined number of successive non-improving moves is exceeded, t random edge swaps are applied to the current spanning tree. The number t increases until a pre-determined limit is reached, and is then re-set to 1. The procedure runs until a given termination condition is met.

3 Some computational results

We ran extensive tests over three classes of graphs.

(1) *Square mesh graphs with unit edge costs.* These are $n \times n$ square meshes with nodes positioned at (p, q) where $p, q \in \mathbb{Z}$ and $0 \leq p, q < n$ (n^2 vertices and $2n(n-1)$ edges). Because of the high degree of symmetry of the graph topology and the uniform edge costs, these are considered hard instances where previous constructive heuristics [2, 3] performed badly, with FCB costs being on average three times as large as those of the solutions produced by our algorithms.

(2) *Random simple Euclidean weighted graphs.* The nodes are positioned randomly on a 20×20 square centered at the origin. Each edge between pair of nodes is randomly generated with probability p and cost equal to the Euclidean distance between its adjacent nodes. Our solutions were on average 50% better than those obtained with previous constructive methods [2, 3]. For small instances (10-15 nodes) our local search actually found the optimal solutions.

(3) *Application to periodic timetabling.* This application is described in [6]. Finding minimum FCBs of appropriate graphs leads to a more compact MIP formulation of a certain type of Periodic Event Scheduling Problem (PESP). We were able to find solutions between 5% to 20% better than those found by C. Liebchen using a purpose-built modification of Deo's methods.

References

[1] A. Brambilla, A. Premoli, Rigorous event-driven (red) analysis of large-scale nonlinear rc circuits, IEEE Transactions on Circuits and Systems–I: Fundamental Theory and Applications 48 (8) (2001) 938–946.

[2] N. Deo, G. Prabhu, M. Krishnamoorthy, Algorithms for generating fundamental cycles in a graph ACM Transactions on Mathematical Software (8) (1982) 26–42

[3] N. Deo, N. Kumar, J. Parsons, Minimum-length fundamental-cycle set problem: New heuristics and an empirical investigation, Congressus Numerantium 107 (1995) 141–154.

[4] P. Hansen, N. Mladenović, Variable neighbourhood search, in: F. Glover, G. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer, Dordrecht, 2003.

[5] A. Hertz, E. Taillard, D. de Werra, Tabu search, in: E. Aarts, J. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, Chichester, 1997, pp. 121–136.

[6] C. Liebchen, R. H. Möhring, A case study in periodic timetabling, in: D. Wagner (Ed.), Electronic Notes in Theoretical Computer Science, Vol. 66, Elsevier, 2002.

[7] G. Galbiati, E. Amaldi, On the approximability of the minimum fundamental cycle basis problem, Workshop on Approximation and Online Algorithms (ALGO-WAOA03), Budapest (in press in the Lecture Notes in Computer Science series) .

[8] K. Paton, An algorithm for finding a fundamental set of cycles of a graph, Communications of the ACM 12 (9) (1969) 514–518.

[9] M. Sysło, On some problems related to fundamental cycle sets of a graph, in: R. Chartrand (Ed.), Theory of Applications of Graphs, Wiley, New York, 1981, pp. 577–588.

The Multicommodity Multilevel Bottleneck Assignment Problem

Roberto Aringhieri[1] Roberto Cordone[2]

DTI - University of Milan, via Bramante 65, Crema 26013 ITALY

Abstract

The Multilevel Bottleneck Assignment Problem is defined on a weighted graph of L levels and consists in finding $L-1$ complete matchings between contiguous levels, such that the heaviest path formed by the arcs in the matchings has a minimum weight. The problem, introduced by Carraresi and Gallo [4] to model the rostering of bus drivers in order to achieve an even balance of the workload among the workers, though frequently cited, seems to have never been applied or extended to more general cases. In this paper, we discuss one possible extension, that is the introduction of *multicommodity* aspects to model different classes of workers.

Key words: Crew Rostering, Bottleneck Assignment

1 Introduction

The Bottleneck Assignment Problem is the search for a complete matching on a weighted bipartite graph, such that the weight of the heaviest edge in the matching is minimum. Its multi-level version is defined on a weighted graph of L levels and consists in finding $L-1$ complete matchings between contiguous levels, such that the heaviest path formed by the arcs in the matchings has a minimum weight. It was introduced by Carraresi and Gallo [4] to model the rostering of bus drivers in order to achieve an even balance of the workload among the workers. Their algorithm determines a starting feasible solution by solving a sequence of Bottleneck Assignment Problems on the single levels; then, furtherly improves the solution through a "stabilization" process. The final result, though not necessarily optimal, has a bounded gap with respect to the optimum, and is asymptotically optimal for a large time horizon. Our interest in this problem derives from a similar application: the rostering of workers' shifts for the junk removal company of Crema, in Italy. This practical case, however, requires a more complex model, because each driver is qualified to perform only a subset of the possible shifts.

Crew Rostering has been a lively field of study over the last decades. Most of the approaches in the literature, however, end up with a Set Partitioning model, whose variables correspond to the feasible sequences of shifts assigned to each worker [3]. Some approaches take into account all variables or a

[1] E-mail: `roberto.aringhieri@unimi.it`.
[2] E-mail: `cordone@dti.unimi.it`.

heuristic subset, while the others start with a reduced set of promising variables and apply column generation to introduce other variables only if necessary. Beasley and Cao [1] proposed a dynamic programming approach. Cappanera and Gallo [2] give a multi-commodity flow formulation for an airline crew rostering problem, which is strengthened by valid inequalities and solved with a general-purpose MIP solver.

The bottleneck approach by Carraresi and Gallo [4], though frequently cited, seems to have never been applied or extended to more general cases. Therefore, the main concern of this paper is to suggest one of the possible extension, that is the introduction of *multicommodity* aspect to model different classes of workers. After reporting a $\mathcal{NP}$-completeness proof, we propose a Lagrangean Decomposition [5]. Since the Lagrangean subproblem is not trivial, this paper introduces upper and lower bounding procedures which integrate the ideas in Carraresi and Gallo [4] in a more general framework.

2 Notation and $\mathcal{NP}$-completeness

Given a time horizon of L days, a weighted level graph $G(V,E)$ of L levels models the structure of the service. Its vertices, partitioned into subsets V_ℓ associated to the single days ($V = \cup_{\ell=1}^{L} V_\ell$) correspond to the shifts. Without loss of generality, one can assume the number of shifts per day to be equal to the number of workers n ($V_\ell = \left\{u_1^\ell, \ldots, u_n^\ell\right\}$ for $\ell = 1, \ldots, L$): dummy shifts can be added to guarantee this result. Each shift u_i^ℓ implies a workload w_i^ℓ. The edges of graph G connect only vertices in contiguous levels. They also model trade union agreements: edge $\left(u_i^\ell, u_j^{\ell+1}\right)$ exists if and only if a worker is allowed to perform shift i in day ℓ and shift j the day after. Let S_i^ℓ denote the subset of vertices in $V_{\ell+1}$ which are linked to vertex u_i^ℓ and P_i^ℓ the subset of vertices in V_ℓ which are linked to vertex $u_i^{\ell+1}$. The workers form K classes: there are n_k workers of class k (with $\sum_{k=1}^{K} n_k = n$), and they are able to perform only a specific subset T_k of the shifts in V.

Proposition 1 *It is $\mathcal{NP}$-complete to determine whether the* MMBA *problem admits any feasible solution.*

Proof. Given a *SAT* instance with n variables x_i and m clauses C_j, build the following *MMBA* instance. Graph $G(V,E)$ is made up of $L = m+1$ levels of $2n$ vertices each. The vertices u_i^L and u_{i+n}^L of level L are associated to variable x_i ($i = 1, \ldots, n$). Vertex u_1^j in each of the other $m = L-1$ levels is associated to the corresponding clause C_j ($j = 1, \ldots, m$). In each level from 1 to m, the first n vertices (u_i^j for $i = 1, \ldots, n$) are linked to all vertices in the following level, while the last n vertices (u_{i+n}^j) are only linked to the last n vertices. There are $K = 2n$ classes of workers, which consist of single workers: class i corresponds to literal x_i, class $i+n$ to $\bar{x}_i$. Subset T_i includes all vertices of the graph, apart from the $2n-2$ vertices of level L which are associated to variables different from i and the clause vertices u_1^j such that clause C_j is not satisfied by literal x_i; a corresponding definition holds for subset T_{i+n} and literal $\bar{x}_i$. By construction, the path of worker i ends either in u_i^L or in u_{i+n}^L. In the latter case, this path only visits vertices u_r^ℓ with $r > n$. Therefore, n of the $2n$ paths are confined in this half of the graph, while the other n paths are confined in the other half. More specifically, for each pair of complementary literals, one only concerns vertices u_r^ℓ with $r > n$ and the other only vertices with $r \leq n$. The clause vertices can only be visited by paths corresponding to satisfying literals. Thus, a feasible solution to the *MMBA* identifies a satisfying truth assignment. $\square$

3 A model

The problem admits the following mathematical formulation. Let $y_{ij}^{\ell} = 1$ if a worker of any group is assigned to shift i in day ℓ and to shift j in day $\ell + 1$, 0 otherwise. Let $x_{ij}^{k\ell} = 1$ if a worker of group k is assigned to shift i in day ℓ and to shift j in day $\ell + 1$, 0 otherwise. Note that $x_{ij}^{k\ell}$ is undefined when the workers of class k are unable to perform either shift i in day ℓ or shift j in day $\ell + 1$. Let s_i^{ℓ} be the total workload from day 1 to day ℓ of the worker performing shift i in day ℓ, and z the maximum total workload over all workers.

$$
\begin{array}{lllr}
\mathbf{P_1} : \min & z & & \\
\text{s.t.} & \displaystyle\sum_{j \in S_i^{\ell}} y_{ij}^{\ell} = 1 & i = 1, \ldots, n,\ \ell = 1, \ldots, L-1 & (1) \\
& \displaystyle\sum_{i \in P_j^{\ell}} y_{ij}^{\ell} = 1 & i = 1, \ldots, n,\ \ell = 1, \ldots, L-1 & (2) \\
& \displaystyle\sum_{k=1}^{K} x_{ij}^{k\ell} = y_{ij}^{\ell} & \left(u_i^{\ell}, u_j^{\ell+1}\right) \in E & (3) \\
& \displaystyle\sum_{j \in P_i^{\ell-1}} x_{ji}^{k\ell} = \sum_{j \in S_i^{\ell}} x_{ij}^{k\ell} & u_i^{\ell} \in V_2, \ldots, V_{L-1},\ k = 1, \ldots, K & (4) \\
& \displaystyle\sum_{i=1}^{n} \sum_{j \in S_i^1} x_{ij}^{k1} = n_k & k = 1, \ldots, K & (5) \\
& s_i^1 = w_i^1 & i = 1, \ldots, n & (6) \\
& \displaystyle s_i^{\ell} \geq w_i^{\ell} + \sum_{j \in P_i^{\ell-1}} s_j^{\ell-1} y_{ji}^{\ell-1} & i = 1, \ldots, n,\ \ell = 2, \ldots, L & (7) \\
& z \geq s_i^L & i = 1, \ldots, n & (8) \\
& x_{ij}^{k\ell} \in \{0, 1\} & \left(u_i^{\ell}, u_j^{\ell+1}\right) \in E,\ k = 1, \ldots, K & (9) \\
& y_{ij}^{\ell} \in \{0, 1\} & \left(u_i^{\ell}, u_j^{\ell+1}\right) \in E & (10)
\end{array}
$$

3.1 A Lagrangean approach

If one relaxes in a Lagrangean fashion constraints (3), the problem decomposes in two classes of subproblems. The former includes K min-cost flow problems, which only concern the $x_{ij}^{k\ell}$ variables

$$
\begin{array}{llr}
\mathbf{L_1^k} : \min & \displaystyle\sum_{(u_i^{\ell}, u_j^{\ell+1}) \in E} \lambda_{ij}^{\ell} x_{ij}^{k\ell} & (11) \\
\text{s.t.} & (4)\ (5)\ (9) &
\end{array}
$$

where the integrality constraints (9) are redundant. Also notice that the underlying graphs are acyclic.

The latter is an unusual assignment problem, concerning the y_{ij}^{ℓ}, s_i^{ℓ} and z variables:

$$\mathbf{L_2} : \min \quad z + \sum_{(u_i^{\ell}, u_j^{\ell+1}) \in E} \lambda_{ij}^{\ell} y_{ij}^{\ell} \qquad \text{s.t.} \quad (1)\ (2)\ (6)\ (7)\ (8)\ (10) \tag{12}$$

whose objective function is composed of two terms. The first term leads to the bottleneck assignment problem introduced in Carraresi and Gallo [4], for which an asymptotically optimal approximation algorithm exists. The second one leads to a classical assignment problem: constraints (6), (7) and (8) are redundant since z is here a free variable.

To solve $\mathbf{L_2}$, we obtain a lower bound by optimizing separately the bottleneck and the min-cost subproblems, and summing the resulting optimal values. Notice that the bottleneck subproblem, which is the harder of the two, does not depend on the Lagrangean multipliers λ_{ij}^{ℓ}. Therefore, even if the multipliers are iteratively updated, e. g. by a subgradient procedure, it is solved once for all at the beginning of the computation. On the contrary, the min-cost subproblem needs to be solved at each step.

We observe that the algorithms for both the bottleneck and the min-cost assignment problems are based on the improvement of a given starting solution through the determination of augmenting paths. Adapting these procedures, it is possible to obtain an upper bound for the problem $\mathbf{L_2}$.

4 Conclusions

In this work, we have presented a new problem, derived from a real-world application, which extends the classical paper by Carraresi and Gallo [4] on the Multilevel Bottleneck Assignment Problem. We have also outlined a possible solution scheme, based on a Lagrangean approach, which provides lower and upper bounds on the optimum.

Another possible solving approach could take into account the special "onion-like" structure in which $T_1 \subseteq \ldots \subseteq T_K$. This case is not furtherly discussed here, because it does not hold in our practical application, though it could characterize other real-world cases.

A different approach can consider a two-phase model: the first one distributes the shifts among the workers' classes; then, a second phase models, for each class, the assignment of the shifts to the workers. Following this approach, approximation algorithms can be derived.

References

[1] J. E. Beasley and B. Cao (1998), A Dynamic Programming based algorithm for the Crew Scheduling Problem, *Computers & Operations Research* 53, 567–582

[2] P. Cappanera and G. Gallo (2003), On the Airline Crew Rostering Problem, *to appear in Operations Research*

[3] A. Caprara, P. Toth, D. Vigo and M. Fischetti (1998), Modeling and Solving the Crew Rostering Problem, *Operations Research* 46, 820–830

[4] P. Carraresi and G. Gallo (1984), A Multi-level Bottleneck Assignment Approach to the bus drivers' rostering problem, *European Journal of Operational Research* 16, 163–173

[5] M. Guignard and S. Kim (1987), Lagrangean decomposition: a model yielding stronger Lagrangean bounds, *Mathematical Programming* 39, 215-228.

An asymmetric vehicle routing problem arising in the collection and disposal of special waste

Roberto Aringhieri [1]

DTI, Università di Milano

Maurizio Bruglieri [2], Federico Malucelli [3]

DEI, Politecnico di Milano

Maddalena Nonato [4]

Dipartimento di Ingegneria, Università di Ferrara

Abstract

In this paper we consider a particular pick-up and delivery vehicle routing problem, with unit vehicle capacity and possible compatibility constraints between consecutive operations. The problem arises in the collection and disposal of bulky recyclable waste, where containers of different types, used to collect different waste materials, must be picked-up to be emptied at suitable disposal plants and replaced by empty containers alike. Disposal plants depend on the material and are located in different sites. Here we provide a graph model based on an Asymmetric Vehicle Routing formulation and discuss heuristic algorithms. Preliminary computational results obtained on real data are reported.

1 Introduction

We consider the collection of bulky recyclable waste as usually deployed in Italy, where several *collection points* (so called "isole ecologiche") are located in the city suburbs or at some other strategic sites. Several containers are present at each collection point, one container for each type of collected waste material: e.g., paper, metals, green and garden waste, wood, glass, etc. Users bring their waste and dispose it into the appropriate container, according to the material. Containers are of different type, depending on the access side (left, right or rear) or on the presence of a compacting equipment, and this varies from collection point to collection point. Once a container is full, a disposal request is issued, consisting of the following two actions, to be carried out not necessarily in this order:

[1] E-mail: `roberto.aringhieri@unimi.it`.
[2] E-mail: `bruglier@elet.polimi.it`.
[3] E-mail: `malucell@elet.polimi.it`.
[4] E-mail: `mnonato@ing.unife.it`.

i) the full container is brought to a qualified *disposal plant* to be emptied;

ii) an empty container of the same kind is brought to the collection point.

A fleet of homogeneous vehicles is available. Each vehicle can carry a single container at a time, either empty or full. Traditionally, the two actions composing a service request are carried out as a whole by a single vehicle, as described in [2]. On the other hand, splitting the two actions introduces a substantial degree of freedom, as it emerges in [4], since any sequence of operation is allowed and the loading and unloading of a container is not necessarily assigned to the same vehicle.

The problem addressed in this paper is the following. Consider a fleet of homogeneous vehicles hosted at a single depot, and a set of additional empty containers of given types stored at the depot or at some other sites. Assume that the location of disposal plants for each type of material is given and all travel times along the road network are known. Given a set of service requests, the aim is to determine the vehicle routes starting and ending at the depot involving pick-up of full containers at collection points, dumping operations at appropriate disposal plants, delivery of empty containers where required, while minimizing both the number of vehicles and the global travelled time.

The problem can be seen as a particular Asymmetric Vehicle Routing Problem (AVRP) on a suitable graph (see section 2) whose peculiar structure allows us to devise efficient heuristic algorithms. In a preliminary computational experience we compare the results of our algorithms both with those currently implemented in some real life cases of a regional area in central Italy (section 4) and with the exact optimal solutions (see section 3).

Closely related problems are addressed in [2], [4] and [6], but as far as we know the general case, dealing with different types of containers and multiple disposal facilities has never been tackled.

2 The graph structure and the AVRP formulation

For reason of simplicity we consider the availability of a single depot.

Let $\{1,\dots,n\}$ be the service requests. Each service request i is characterized by a material μ_i, a container type β_i, and a collection point γ_i. If two requests refer to the same container type, they are said to be *compatible*. The graph G supporting our Vehicle Routing model is not directly related to the physical network. The nodes of graph G represent service requests, in particular for each request i we have two nodes f_i and e_i representing the full container to be brought to a disposal plant for material μ_i and the request of an empty container of type β_i to replace the full one at γ_i, respectively. Moreover, we consider a node 0 representing the depot, K nodes $d'_1,\dots,d'_K$ representing the possible pick-up of K available empty containers located at the depot and K nodes $d''_1,\dots,d''_K$ representing the delivery of the same containers to the depot.

Notice that we do not explicitly model nodes representing disposal plants. Indeed disposal operations are embedded into the arcs. Two main classes of arcs are given.

Arcs connecting f_i and e_j, for any pair of compatible requests (i,j) (not necessarily $i \neq j$). Arc (f_i,e_j) corresponds to picking up the full container of request i, taking it to and dumping it at the closest disposal plant for material μ_i on the way to the collection point of request j, where the empty container is delivered. Note that once the pair of requests is known, it is easy to determine the closest disposal plant and to evaluate the travel time which should include also the time needed to carry out the loading-unloading operations as well as the dumping service. If $i = j$, then we have arc (f_i, e_i) corresponding to the simplest case of picking up a full container, emptying it at the closest disposal plant and bringing it back to its original site. Arcs in this class model vehicles activities when loaded with a container, either full or empty, and are said *loaded arcs*.

A second class of arcs, so called *unloaded arcs*, concerns arcs connecting e_i to f_j for any pair of service requests (i, j). Arc (e_i, f_j) models a vehicle that has just delivered an empty container of type β_i at collection point γ_i, and travels unloaded up to collection point γ_j in order to pick-up the full container of request j. Note that request i and j do not need to be compatible. The travel time related to this arc is simply the travel time from γ_i to γ_j, including container loading and unloading times. If $i = j$, arc (e_i, f_i) corresponds to bringing an empty container to satisfy request i and switch it with the full container.

Furthermore, since vehicles at the depot are supposed to be unloaded, we consider all arcs going from 0 to each f_i node and all arcs from 0 to all $d'_1, \ldots, d'_K$ nodes and from the latter nodes to every compatible node e_i. The arcs $(0, f_i)$ model the loading of a full container on an unloaded vehicle whereas the arcs $(0, d'_k)$, $k = 1, \ldots, K$ model the loading of an empty container available at the depot on an unloaded vehicle both at the beginning of a vehicle route. Vice versa for the delivery of containers to depot we consider all arcs going from each node f_i to every compatible node $d''_1, \ldots, d''_K$ and all arcs from the latter nodes to depot 0. Thus the arcs (f_i, d''_k), $k = 1, \ldots, K$ correspond to bring back to the depot the empty container obtained after having satisfied the emptying request i, whereas arcs $(d''_k, 0)$ represent the unloading of the same container at the depot. Finally, all arcs from every node e_i to the depot and from every node $d''_1, \ldots, d''_K$ to any node $d'_1, \ldots, d'_K$ are added: the former ones represent the going back to the depot of an unloaded vehicle at the end of its route, whereas the latter ones model the unloading of a container from a vehicle and the loading of another container both at the depot during the vehicle route.

Observe that all arcs correspond to operations that can be executed by a single vehicle. A vehicle route is a cycle on G starting and ending at 0, feasible routes have length less than or equal to a maximum given duration T. Therefore it is easy to see that a collection of feasible routes passing exactly once by all nodes of the graph and at most once by $d'_1, \ldots, d'_K$, $d''_1, \ldots, d''_K$ is a feasible solution of the waste disposal problem. Therefore solving the corresponding AVRP on G would solve our problem.

3 A mathematical model for the refuse collection problem

Later on we use the following notations:

n $\equiv$ cardinality of service requests;

K $\equiv$ cardinality of empty containers available at the depot;

F $\equiv \{f_i : i \in 1, \ldots, n\}$ set of nodes for the emptying requests;

E $\equiv \{e_i : i \in 1, \ldots, n\}$ set of nodes for the requests of return of empty containers;

D' $\equiv \{d'_1, \ldots, d'_K\}$ set of empty containers available at the depot;

D'' $\equiv \{d''_1, \ldots, d''_K\}$ set of empty containers which must be returned to the depot;

P $\equiv$ set of disposal plant;

V $\equiv$ $F \cup E \cup D' \cup D'' \cup \{0\}$;

A $\equiv$ arc set (see Table 1);

T $\equiv$ maximum duration for a vehicle route;

c_{uv} $\equiv$ operational time to serve node v after node u, $\forall$ $(u, v) \in A$ (see Table 1);

t_{ij} $\equiv$ time employed by the vehicles to move from site i to site j;

σ_h $\equiv$ time for emptying a container in disposal plant h, $\forall h \in P$;

τ' $\equiv$ time for loading a container;

τ'' $\equiv$ time for unloading a container;

L $\equiv$ lower bound on the number of vehicles employed;

U $\equiv$ upper bound on the number of vehicles employed;

Arcs (u, v)	Operational times c_{uv}	Nodes
$(0, f_i)$	$t_{0\gamma_i} + \tau'$	$\forall f_i \in F$
$(0, d'_k)$	τ'	$\forall d'_k \in D'$
(d'_k, e_i)	$t_{0\gamma_i} + \tau''$	$\forall d'_k \in D'$, $\forall e_i \in E$ which are compatible
$(e_i, 0)$	$t_{\gamma_i 0}$	$\forall e_i \in E$
$(d''_k, 0)$	0	$\forall d''_k \in D''$
(f_i, d''_k)	$t_{\gamma_i h_i} + \sigma_{h_i} + t_{h_i 0} + \tau''$ (*)	$\forall f_i \in F$, $\forall d''_k \in D''$
(e_i, f_j)	$t_{\gamma_i \gamma_j} + \tau'$	$\forall e_i \in E$, $\forall f_j \in F$
(f_i, e_j)	$t_{\gamma_i h_i} + \sigma_{h_i} + t_{h_i \gamma_j} + \tau''$ (**)	$\forall f_i \in F$, $\forall e_j \in E$ which are compatible
(d''_k, d'_k)	τ'	$\forall d''_k \in D''$, $\forall d'_k \in D'$

Table 1
This table contains the definition of the arc set A and the respective costs. (*) Here h_i represents the nearest dump compatible with the garbage of f_i along the path from site γ_i to the depot.
(**) Here h_i represents the nearest dump compatible with the garbage of f_i along the path from site γ_i to site γ_j.

Mathematically, the waste collection problem can be formulated as the following Mixed Integer Linear Program (MILP):

$$\min z = M \sum_{v \in \delta^+(0)} x_{0v} + \sum_{(u,v) \in A} c_{uv} x_{uv} \tag{1}$$

$$\sum_{v \in \delta^+(u)} x_{uv} = 1 \qquad \forall u \in F \cup E \tag{2}$$

$$\sum_{v \in \delta^+(u)} x_{uv} = \sum_{v \in \delta^-(u)} x_{vu} \qquad \forall u \in V \tag{3}$$

$$\sum_{v \in \delta^+(u)} x_{uv} \leq 1 \qquad \forall u \in D' \cup D'' \tag{4}$$

$$\sum_{u \in \delta^+(0)} x_{0u} \leq U \tag{5}$$

$$\sum_{u \in \delta^+(0)} x_{0u} \geq L \tag{6}$$

$$t_v + (T + c_{uv})(1 - x_{uv}) \geq c_{uv} + t_u \qquad \forall (u,v) \in A \setminus BS(0) \tag{7}$$

$$t_u + c_{u0} x_{u0} \leq T \qquad \forall u \in \delta^-(0) \tag{8}$$

This formulation employs binary variables x_{uv} for all $(u, v) \in A$, indicating if some vehicle traverses arc (u, v) or not and continuous variables t_u for all $u \in V$ representing the time when the node u is visited by some vehicle. The objective function is the weighted sum of the number of vehicle actually used and the total running time (M represents a constant suitable big).

4 Preliminary computational results

The particular graph topology described in section 2 suggests several neighborhood structures that can be exploited within a local search framework (see [3]). The starting solution of the local search is computed by applying a modified Clarke and Wright algorithm [5].

A preliminary computational campaign has been carried out on real data provided by the waste collection company operating in the regional area of Perugia. The company serves ten collection points distributed in an area of about 450 km^2, six different types of containers, three disposal centers used to recycle ten different types of material. The problem instances, deriving from the daily operations, involve three vehicles and up to 11 service requests, they require quite long hauls and they have a time limit of 375 minutes. Even though these instances are quite small, the first results provided by the proposed heuristics favorably compare with the exact solution values.

Moreover the resulting routes are extremely better than those operated by the company, saving travel time and in certain cases saving also one vehicle out of the three devoted to this kind of service. In the light of the obtained results, the company is thinking to extend the service also to industrial waste collection, which would significantly increase the size of the instances. Here we report some results on some real instances. We report the total travel time of the optimal solution in minutes (Opt), the percentage gap between the optimal solution and the modified Clarke and Wright (Δ CW), the percentage gap between the optimal solution and the local search (Δ LS), and the percentage gap between the local search and the company solution (Δ G). As far as number of vehicles is concerned, entries in boldface report the cases in which the local search is able to spare a vehicle compared to the company solution. CPU times on a 2GHz PC are also reported.

Instance	**Req**	**Opt**	**Δ CW**	**Δ LS**	**Δ LS-G**	**CPU Opt**	**CPU LS**
17.11.2003	8	668	7.04	**5.54**	11.91	3.52	0.03
18.11.2003	3	300	14.00	0.00	0.00	0.1	0.05
19.11.2003	7	615	0.00	0.00	0.00	3.44	0.04
20.11.2003	8	701	21.83	0.71	15.16	7.56	0.05
21.11.2003	6	626	**2.56**	**0.15**	3.00	0.1	0.01
22.11.2003	9	843	15.18	15.18	3.09	21.32	0.03
24.11.2003	8	684	24.12	**0.88**	3.19	3.05	0.06
25.11.2003	8	575	2.61	2.61	13.90	19.42	0.07
26.11.2003	6	599	1.17	1.17	12.05	6.71	0.03
27.11.2003	8	839	9.30	8.82	6.79	5.78	0.01
28.11.2003	6	606	2.97	2.97	12.02	0.1	0.01
29.11.2003	11	882	2.72	1.59	19.98	0.1	0.07

Table 2
Some results on real case instances

References

[1] E. Aarts and J.K. Lenstra (eds.), Local Search in Combinatorial Optimization, John Wiley and Sons, Chichester, UK, 1997.

[2] C. Archetti, M.G. Speranza, Collection of waste with single load trucks: a real case, `www.eco.unibs.it/dmq/speranza/`

[3] R. Aringhieri, M.Bruglieri, F.Malucelli, M.Nonato, “A particular vehicle routing problem arising in the collection and disposal of special waste”, Tristan V, Le Gosier, Guadeloupe, French West Indies.

[4] L. Bodin, A. Mingozzi, R. Baldacci, M. Ball, “The Rollon–Rolloff Vehicle Routing Problem”, Transportation Science 34 (3) 271-288 (2000).

[5] G.Clarke and J.V.Wright, “Scheduling of veichles from a central depot to a number of delivery points”, Operations Research 12, 568–581 (1964)

[6] L. De Muelemeester, G.Laporte, F.V.Louveaux, and F. Semet, “Optimal Sequencing of Skip Collections and Deliveries,” Journal of Operational Research Society 48, 57–64 (1997).

[7] P. Toth, D. Vigo (eds). The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. S.I.A.M., Philadelpia, PA, 2002.

A New Local Condition for a Graph to be Hamiltonian

Armen S. Asratian[1]

Department of Mathematics, Linköping University, S-581 83, Linköping, Sweden

Abstract

Let G be a connected graph and for each vertex w in G $M_2(w)$ denote the set of all vertices whose distance from w does not exceed 2. The ball of radius 2 centered at w is a subgraph of G induced by the set $M_2(w)$.

We prove that a connected graph G is hamiltonian if every ball of radius 2 in G is 2-connected and $d(u) + d(v) \geq |M_2(w)| - 1$ for every path uwv with $uv \notin E(G)$. This implies the following local analogue of a theorem of Nash-Williams: a connected r-regular graph G is hamiltonian if every ball of radius 2 in G is 2-connected and $r \geq (|M_2(w)| - 1)/2$ for each vertex w of G.

Key words: Hamilton cycle, local condition, ball

We use [9] for terminology and notation not defined here and consider finite simple graphs only. Let $V(G)$ and $E(G)$ denote, respectively, the vertex set and edge set of a graph G, and let $d(u, v)$ denote the distance between vertices u and v in G. For each vertex u of G, we denote by $N(u)$ and $M_2(u)$ the sets of all $v \in V(G)$ with $d(u, v) = 1$ and $d(u, v) \leq 2$, respectively. The number $|N(u)|$ is denoted by $d(u)$. For a vertex u of a graph G the ball of radius 2 centered at u is a subgraph of G induced by the set $M_2(u)$. A classical result on hamiltonian graphs is the following theorem.

Theorem A (Ore [13]). Let G be a graph on at least 3 vertices such that $d(u) + d(v) \geq |V(G)|$ for each pair of nonadjacent vertices u, v. Then G is hamiltonian.

Later, several authors have found (see, for example [10]) that Ore's condition in Theorem A can be relaxed by one if we allow a class of exceptions. Set

$\mathbf{K} = \{G : K_{n,n+1} \subseteq G \subseteq K_n \vee \ K^c_{n+1}, \ \ n \geq 3\}$ where $\vee$ denotes join.

Theorem B. Let G be a 2-connected graph on at least 3 vertices such that $d(u)+d(v) \geq |V(G)|-1$ for each pair of nonadjacent vertices u, v. Then either G is hamiltonian or $G \in \mathbf{K}$.

The next result shows that for regular graphs the condition of Theorem B is enough for hamiltonicity.

Theorem C (Nash-Williams [12]). A 2-connected regular graph G is hamiltonian if $d(u) \geq \frac{1}{2}(|V(G)|-1)$.

[1] E-mail: `asratian@sm.luth.se`.

Clearly, every graph G satisfying one of the conditions in Theorems A, B and C has diameter 2. ln fact most of the existing generalizations of Theorem A only apply to graphs G with large edge density ($|E(G)| \geq constant \cdot |V(G)|^2$) and small diameter ($o(|V(G)|)$).

The present author and N.K.Khachatrian [1-4] developed some local criteria for the existence of Hamilton cycles in a connected graph which are analogues of the global criteria due to Dirac, Ore and others. The idea was to show that the global concept of hamiltonicity can, under rather general conditions, be captured by local phenomena, using the structure of balls of small radii. This local approach gives the possibility to find new classes of graphs with Hamilton cycles which, in particular, also contain infinite subclasses of graphs with small edge density and large diameter.

Let L_i be the set of all connected graphs G with $|V(G)| \geq 3$ where

$$d(u) + d(v) \geq |N(u) \cup N(v) \cup N(w)| - i, \text{ for each path } uwv \text{ with } uv \notin E(G).$$

Clearly $L_i \subseteq L_{i+1}$, for each i, and the set L_0 contains all graphs satisfying the condition of Theorem A.

Theorem D (Asratian and Khachatrian [3]). Every graph $G \in L_0$ is hamiltonian.

Some other properties of the graphs in the set L_0 were found in [5-7].

Let $Q_1 = \{G : K_{n,n} \subseteq G \subseteq K_n \vee K_n^c, n \geq 3\}$ and $Q_2 = \{K_{n,n} : n \geq 3\}$. It was proved the following:

- A graph $G \in L_0$ is pancyclic if and only if $G \notin Q_2$ [5].

- A graph $G \in L_0$ is Hamilton-connected if and only if G is 3-connected and $G \notin Q_1$ [6].

- A graph $G \in L_0$ is vertex pancyclic if and only if $G \notin Q_2$ and each vertex of G lies on a triangle [7].

Theorem D was improved in the the following way:

Theorem E (Asratian et al.[8]). Let G be a graph $G \in L_1$ where $|N(u) \cap N(v)| \geq 2$ for each pair of vertices u, v with $d(u, v) = 2$. Then either G is hamiltonian or $G \in \mathbf{K}$.

Some other results on hamiltonian properties of graphs from the set L_1 can be found in [8,11,14].

Clearly, if $w \in N(u) \cap N(v)$ then $N(u) \cup N(w) \cup N(v) \subseteq M_2(w)$. Therefore Theorem D implies the following

Corollary F. Let G be a connected graph on at least 3 vertices where $d(u)+d(v) \geq |M_2(w)|-1$ for each path uwv with $uv \notin E(G)$. If $|N(u) \cap N(v)| \geq 2$ for each pair of vertices u, v with $d(u, v) = 2$, then G is either hamiltonian or $G \in \mathbf{K}$.

It is not difficult to see that if $|N(u) \cap N(v)| \geq 2$ for each pair of vertices u, v with $d(u, v) = 2$ in G then every ball of radius 2 in G is 2-connected.

Our main result is the following theorem which is a joint generalization of Theorems A, B, C and Corollary F.

Theorem 1. Let G be a connected graph with $|V(G)| \geq 3$ where $d(u) + d(v) \geq |M_2(w)| - 1$ for every path uwv with $uv \notin E(G)$. If all balls of radius 2 in G are 2-connected, then either G is a hamiltonian graph or $G \in \mathbf{K}$.

Since the set $\mathbf{K}$ contains no regular graphs, Theorem 1 implies the following corollary:

Corollary 2. Let G be a connected r-regular graph where every ball of radius 2 is 2-connected and $r \geq \frac{1}{2}(|M_2(w)| - 1)$, for each vertex w. Then G is a hamiltonian graph.

Let us point out that there are infinite classes of graphs G with small edge density ($\Delta(G) \leq$ *constant*) and large diameter ($\geq$ *constant* $\cdot |V(G)|$) which satisfy the conditions of Theorem 1 and Corollary 2.

References

[1] A.S.Asratyan and N.K.Khachatryan, Two theorems on hamiltonian graphs (Russian), *Mat.Zametki*, **35**(1984), no.1, 55-61 English translation *Math. Notes*, **35** (1984), no.1, 32-35

[2] A.S. Asratyan and N.K. Khachatryan, Investigation of the Hamiltonian property of a graph using neighborhoods of vertices. (Russian) Akad. Nauk Armyan. SSR Dokl. 81 (1985), no. 3, 103–106.

[3] A.S.Asratian and N.K. Khachatrian, Some localization theorems on hamiltonian circuits, *J. Combin. Theory Ser. B* **49** (1990) 287 - 294

[4] A.S. Asratian and N.K. Khachatrian, Stable properties of graphs, *Discrete Mathematics* **90** (1991) 143-152

[5] A.S. Asratyan and G.V.Sarkisyan, On cyclic properties of some hamiltonian graphs (Russian), *Diskretnaja Matem.*, **3** (1991) 91-104

[6] A.S. Asratian, A criterion for some hamiltonian graphs to be Hamilton–connected, *Australasian J. Combin.* **10** (1994) 193-198

[7] A.S. Asratian and G.V.Sarkisian, Some panconnected and pancyclic properties of graphs with a local Ore-type condition. Graphs Combin. 12 (1996), no. 3,209–219

[8] A.S.Asratian, H.Broersma, J.van den Heuvel, H.J.Veldman, On graphs satisfying a local Ore - type condition,*J.Graph Theory*,**21**(1996) 1-10

[9] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications.* (Macmillan, London and Elsevier, New York, 1976).

[10] H. A. Jung, On maximal circuits in finite graphs, *Annals of Discrete Math.*, **3** (1978) 129–144.

[11] R.Li and R.H. Shelp, Some hamiltonian properties of L_1-graphs, Discrete Math., 223 (2000), 207-216

[12] C.St.J.A. Nash-Williams, Hamilton arcs and circuits in finite graphs, Recent Trends in Graph Theory, Springer, Berlin (1971), pp.197-210

[13] O. Ore, Note on hamiltonian circuits, Amer. Math. Monthly **67** (1960) 55.

[14] A.Saito, A local Ore-type condition for graphs of diameter two to be Hamiltonian, J.Combin.Math.Combin.Comput. 20(1996) 243-249

Knowledge State Algorithms and the 2-Server Problem

Wolfgang W. Bein[1]

Department of Computer Science, University of Nevada, Las Vegas, Nevada 89154

We introduce the novel concept of knowledge states. A knowledge state simply states conditional obligations of an adversary, by fixing a work function, and gives a distribution for the algorithm. When a knowledge state algorithm receives a request, it then calculates one or more "subsequent" knowledge states, together with a probability of transition to each. The algorithm then uses randomization to select one of those subsequents to be the new knowledge state. Although the formal definition of knowledge state algorithms appears, as yet, in no publication, many well-known algorithms can be viewed as knowledge state algorithms.

We have used optimization techniques to construct a non-trivial knowledge state algorithm for the 2-server problem on the line with competitive ratio $\frac{71}{36} \approx 1.972$. As much as one avenue of investigation might be to further improve this result for the line, what we really envision is to use this technique to finally settle the question of whether there exists an online algorithm with competitive ratio better than 2 for general spaces – a notorious open problen in online algorithms. For progress in that direction, we consider the class of metric spaces $M_{2,4}$, which consists of all metric spaces where every distance is either 1 or 2, and where the perimeter of every triangle is either 3 or 4. Using the knowledge state approach we are able to obtain the result that there exists a C-competitive randomized online algorithm for the 2-server problem in the space $M_{2,4}$ with $C = \frac{173+\sqrt{137}}{112} \approx 1.649149106$.

[1] E-mail: `bein@cs.unlv.edu`. Research supported by NSF grant CCR-0132093. Joint work with Lawrence L. Larmore.

Network design with grooming constraints

Pietro Belotti[1], Federico Malucelli[2]

DEI, Politecnico di Milano, P.zza L. da Vinci 32, 20133 Milano, Italy

Abstract

Networks are physically and logically decomposed into layers with different technological features. Often, the routing of a demand through a non-multiplexing layer is made by *grooming* several demands at another, multiplexing-capable layer, thus using less capacity on the former but more on the latter. The problem of designing such a multi-layer network so as to route a set of traffic demands can be solved by embedding multiplexing into a well-suited model. We restrict to a two-layer problem as this is most common in today's network world, then we represent grooming through a model based on paths and semi-paths, and propose a row-column generation approach to solve a set of problems on real-world large networks.

1 Introduction

Consider a network dimensioning problem: a graph $G = (V, E)$, representing the network topology, and a set Q of traffic demands expressed by triplets (s_k, t_k, v_k), $k \in Q$, are given. The solution is the equipment to install on the initial topology so as to route all traffic demands at a minimum installation cost. This subject is well known in the recent literature; the reader may find a survey of network design problems in [3].

Suppose now the network is structured in two layers, with different costs and characteristics. The upper layer has greater capacity but cannot split demands, whereas at the lower layer, demands can split but capacity is limited. We assume all demands originate and end at the low layer. This design-routing problem can be modelled through a multi-layer structure as depicted in Figure 1: the high capacity upper layer has no demultiplexing capability, hence two groomed demands must be transferred to the multiplexing-capable low layer in order to be split. Industrial application regard the upper layer and the inter-layer capacity within a node as the most expensive equipment.

We focus in this work on the design of a multi-layer network that minimises the weighted sum of node and link capacity, provided the different capabilities each layer offers. In section 2 we outline a general path-based model for multi-layer networks; in section 3 we describe a row-column generation heuristic, and in section 4 we report some initial results on large real-world networks.

[1] E-mail: `belotti@elet.polimi.it`.

[2] E-mail: `malucell@elet.polimi.it`.

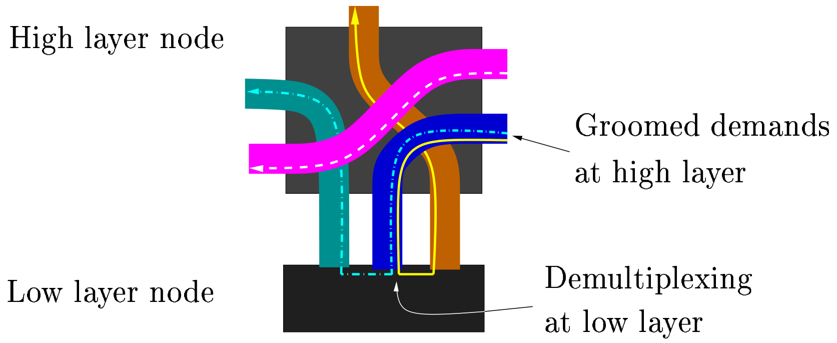


Figure 1. Grooming on multi-layer networks.

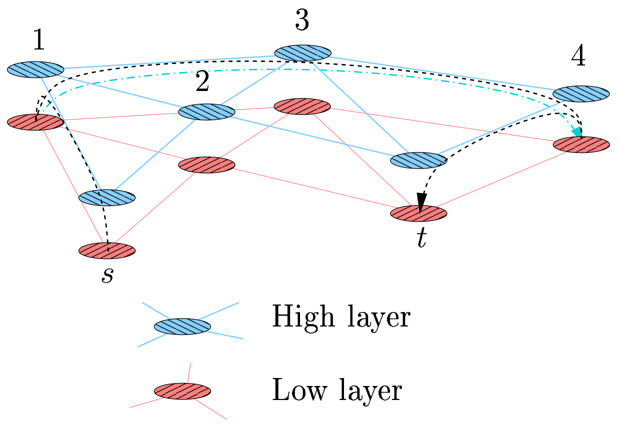


Figure 2. Paths and semi-paths.

2 Optimisation model

We consider a graph G' consisting of two super-imposed graphs equal to the original G: the upper layer G_h and the lower layer G_l; an edge is added between the upper and lower parts of a node $i \in V$. Hence, $G' = (V_h \cup V_l, E_h \cup E_l \cup E_m)$, where V_h and V_l are a copy of V, E_h and E_l are a copy of E, and E_m is the set of intra-node channels for communication between the two layers.

Further notation is needed: we call P the set of all paths in G'; notice that this implies that two paths may cross the same set of edges in G yet be different in G' as each edge/node has an upper and a lower layer part. Given a nodepair (s, t), the set of paths from s to t is denoted by P^{st}, while P_e is the set of paths across a given edge e. We define integer variables z_p for each path $p \in P$ from s to t, standing for the amount of traffic of demand (s, t) carried on path p, and impose that all traffic demands (s_q, t_q, d_q), $q \in Q$ be satisfied:

$$\sum_{p \in P^{s_q t_q}} z_p \geq d_q \qquad \forall q \in Q \tag{1}$$

Consider the dashed (s, t)-path in Figure 2, leaving s from the low layer, crossing nodes 1, 2 and 3, and low layer nodes at 1 and 3. The latter are visited so as to groom this (s, t)-path and the dash-dotted $(1, 3)$-path and hence to allocate a single channel on the semi-path defined on edges $\{1, 3\}$ and $\{3, 4\}$.

Hence, network capacity is implicitly defined on path portions starting and ending in low-layer nodes. We call these path portions *semi-paths*. We define, for all semi-path $s \in P$, an integer variable y_s

standing for the capacity to be allocated to semi-path s. As in classical network design problems, where the capacity on an edge e must suffice for the total traffic flowing through e, y_s is an upper bound to the traffic on paths containing s:

$$\sum_{p \in P : s \subseteq p} z_p \leq y_s \qquad \forall s \in P \tag{2}$$

Semi-paths originating/ending in node i also account for the inter-layer capacity needed at the node, and therefore represent the capacity to be installed on the upper layer network and are associated a cost c_s equal to the sum of twice the inter-layer channel cost and the edge cost. If the lower layer can also transport traffic, we can define another capacity variable x_e associated to each low layer edge and impose that it be sufficient to carry the low-layer traffic:

$$\sum_{p \in P : e \subseteq p} z_p \leq x_e \qquad \forall e \in E_l \tag{3}$$

The objective function $\sum_{s \in P} c_s y_s + \sum_{e \in E_l} c_e x_e$, given by the upper layer capacity and the inter-layer channel cost, included in c_s, plus the cost of the lower layer capacity, is to be minimised subject to (1), (2), (3) and integrality constraints for x, y and z variables.

3 A row-column generation approach

Path formulations for network design problems are bound to be solved by methods considering a small subset of columns, possibly adjoining cheaper columns as the algorithm exploits useful dual information about the problems solved at each iteration. First, a set of paths is generated for each demand nodepair (s, t). Then, we create variables y_s using pairs of paths sharing at least a certain number K of edges. Most likely, these do not form the optimal solution, and a *column generation* algorithm (for an introduction, see e.g. [1, 2]), that repeatedly generates new paths and semi-paths is needed to reach optimality. If integrality constraints are relaxed, the primal and dual formulations are as follows:

$$\begin{array}{lll}
\textbf{(P)} & \min \sum_{s \in P} c_s y_s + \sum_{e \in E_l} c_e x_e & \\
(\sigma_q) & \sum_{p \in P^{s_q t_q}} z_p \geq d_q & \forall q \in Q \\
(\xi_s) & - \sum_{p \in P : s \subseteq p} z_p + y_s \geq 0 & \forall s \in P \\
(\eta_e) & - \sum_{p \in P : e \subseteq p} z_p + x_e \geq 0 & \forall e \in E_l
\end{array}$$

$$x_p \geq 0 \quad \forall p \in P, \qquad y_s \geq 0 \quad \forall s \in P, \qquad x_e \geq 0 \quad \forall e \in E_l$$

$$
\begin{array}{llll}
(\mathbf{D}) & & \max \sum_{q \in Q} d_q \sigma_q & \\
(z_p) & \sigma_q \leq \sum_{s \in P: s \subseteq p} \xi_s + \sum_{e \in E_l : e \in p} \eta_e & & \forall p \in P^{s_q t_q}, q \in Q \\
(y_s) & \xi_s \leq c_s & & \forall s \in P \\
(x_e) & \eta_e \leq c_e & & \forall e \in E_l \\
& \xi_s \geq \quad \forall s \in P, \quad \sigma_q \geq 0 \quad \forall q \in Q, \quad \eta_e \geq 0 \quad \forall e \in E_l & &
\end{array}
$$

From a solution of a formulation with a subset of columns, the dual information is used to prove optimality through the dual constraint associated to variable z_p. Those paths not yet considered by the formulation but whose dual constraint is violated by the current solution are a proof that the primal is not optimal, as this z_p variable has negative reduced cost.

Were it not for the semi-path variables y_s, the *pricing* problem, i.e. finding a variable z_p with (the most) negative reduced cost, would boil down to solving a shortest path problem on graph G where arc lengths are the current optimal value of η variables; if the shortest path has length lesser than σ_q, then we have found a variable z_p with negative reduced cost to include in the next iteration. However, variables y_s are also partially included, as they are also defined on P. Hence, we have to generate these variables in order to improve the subproblem at each iteration. This complicates the procedure a lot, as y_s variables are defined in constraints (2), hence we also must generate rows. We chose therefore to incorporate the creation of y and z variables into a single process, which creates new paths and semi-paths based on the optimal primal (x^*, y^*, z^*) and dual (σ^*, ξ^*, η^*) solution.

Consider a graph equivalent to G where all edges $e \in E$ have length η_e^*. We add an edge $\{i, j\}$ for each $s \in P$ from i to j, with length ξ_s^*. Solving a shortest path problem for each nodepair (s, t) on this auxiliary graph gives a new path which may follow some low-layer edges, some semi-path already defined or, if two or more low-layer edges are crossed consecutively, suggest the definition of a new semi-path. If its total lengths is lesser than σ_q^* we have found a variable with negative reduced cost and, possibly, a set of semi-paths to be added to the formulation. Notice that each new semi-path implies adding a new column and a new row in the formulation.

4 Preliminary results and future work

We have tested the algorithm outlined above on seven network topologies from real-world design problems and report the results in Table 3. For each instance, we report the number of demands $|Q|$, the number of nodes $|V|$ and of edges $|E|$ in the network, the number of paths/columns at the beginning ($|P|_{(t=0)}$) and at the end ($|P|_{\text{final}}$) of the algorithm, the optimal LP value z_{lp} and the value z_{rounded} of a feasible solution obtained by rounding up the LP solution, the time to create the initial paths t_{path}, and the total solution time t_{total}.

We observe that, given the size of all instances, even solving the LP relaxation may prove difficult if one uses a classical modelling approach. We have managed to keep solution times low by adopting an *ad hoc* model. The algorithm, although efficient in some cases, needs to be improved in order to obtain better integer solution.

The general framework we have presented can be applied to several problems in network design,

Table 3
Preliminary performance results of our row and column generation algorithm.

$\lvert Q\rvert$	$\lvert E\rvert$	$\lvert V\rvert$	$\lvert P\rvert_{(t=0)}$	$\lvert P\rvert_{\text{final}}$	z_{lp}	z_{rounded}	t_{path}	t_{total}
13	124	84	150	278	10095.65	32454.31	00:00.30	00:08.40
50	47	42	250	335	470805.43	539393.23	00:00.60	00:06.00
64	39	29	320	391	177805.69	206475.34	00:00.10	00:04.10
83	124	84	1745	1991	292271.12	312630.02	00:01.80	00:56.52
202	259	114	1765	2378	191915.40	197166.25	00:02.50	01:36.10
395	90	80	1975	2096	140508.41	194390.88	00:01.20	00:47.70
495	563	358	4075	5656	440483.19	1519728.44	00:10.90	20:13.40

such as routing and wavelength assignment in Wavelength Division Multiplexing, and in the design of mixed packet-circuit networks to take advantage of statistical multiplexing. Also, in order to improve the efficiency of our method, a more suited path generation procedure is needed. A branch and price algorithm is currently under development.

Acknowledgments

The authors would like to thank Alessandro Cisco for implementing and testing the algorithm.

References

[1] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh and P.H. Vance, “Branch-and-Price: Column Generation for Huge Integer Programs”, *Operations Research*, 46:316-329, 1998.

[2] P.C. Gilmore and R.E. Gomory, “A linear programming approach to the cutting stock problem”. *Operations Research*, 9:848–859, 1961.

[3] D. Yuan, “An annotated bibliography in communication network design and routing”, in *Optimization models and methods for communication network design and routing*, PhD dissertation no. 682 (2001), Institute of Technology, Linköpings Universitet, Linköping (Sweden).

Tree Decompositions of Graphs: Saving Memory in Dynamic Programming [4]

Nadja Betzler [1] Rolf Niedermeier [2] Johannes Uhlmann [3]

Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Sand 13, D-72076 Tübingen, Germany

Abstract

We propose an effective heuristic to save memory in dynamic programming on tree decompositions when solving graph optimization problems. The introduced "anchor technique" is closely related to a tree-like set covering problem.

1 Introduction

Recently, tree decompositions of graphs have received considerable interest in practical applications. Intuitively, a graph with small treewidth allows for efficient solutions of otherwise hard graph problems. The core tool in this solution process—besides constructing a tree decomposition of (hopefully) small width—is dynamic programming on tree decompositions [3]. As a rule, both the running time of this dynamic programming and the memory space consumption are exponential with respect to the treewidth. Indeed, also by our own practical experiences, the main bottleneck often is memory consumption rather than running time.

Attacking the "memory consumption problem" is subject of recent research. For instance, Aspvall et al. [2] deal with the problem by trying to find an optimal traversal of the decomposition tree in order to minimize the number of dynamic programming tables stored simultaneously. Bodlaender and Fomin [4] theoretically investigate a new cost measure for tree decompositions and try to construct better tree decompositions in this way. By way of contrast, in this experimentally oriented paper we sketch a new approach that tries to decrease the memory consumption by employing a (tree-like) set covering technique with some heuristic extensions. The point here is that our so-called "anchor technique" applies whenever, in principle, one needs to store *all* dynamic programming tables of a given tree decomposition. This is usually the case when one wants to solve the optimization version of problems, i.e., to actually construct an optimal solution (or all of them). Here, the anchor technique tries to minimize the redundancy of information stored by avoiding to keep all dynamic programming tables in memory. [5] So far, according to our experiments the anchor technique seems

[1] E-mail: `betzler@informatik.uni-tuebingen.de`.
[2] E-mail: `niedermr@informatik.uni-tuebingen.de`.
[3] E-mail: `johannes@informatik.uni-tuebingen.de`.
[4] Supported by the Deutsche Forschungsgemeinschaft (DFG), research project "PEAL," NI 369/1, and junior research group "PIAF," NI 369/4.
[5] Aspvall et al.'s [2] technique only seems to apply with respect to the decision version of a problem where one only needs to do a bottom-up traversal of the decomposition tree. Bodlaender and

to give the best results when dealing with path decompositions (with memory savings of around 95%) and nice tree decompositions (with memory savings of around 80%).

2 Basic Ideas and the Anchor Technique

To describe our new technique, we first need to introduce some notation [3]. Let $G = (V, E)$ be a graph. A *tree decomposition* of G is a pair $\langle \{X_i \mid i \in I\}, T \rangle$, where each X_i is a subset of V, called a *bag*, and T is a tree with the elements of I as nodes. The following three properties must hold: $\bigcup_{i \in I} X_i = V$; for every edge $\{u, v\} \in E$, there is an $i \in I$ such that $\{u, v\} \subseteq X_i$; and for all $i, j, k \in I$, if j lies on the path between i and k in T, then $X_i \cap X_k \subseteq X_j$. The *width* of $\langle \{X_i \mid i \in I\}, T \rangle$ equals $\max\{|X_i| \mid i \in I\} - 1$. The *treewidth* of G is the minimum k such that G has a tree decomposition of width k.

If the above tree is only a path, then we speak of a *path decomposition.* A tree decomposition with a particularly simple (and useful with respect to dynamic programming, cf. [1, 3]) structure is given by the following. A tree decomposition is called a *nice tree decomposition* if the following conditions are satisfied: Every node of the tree T has at most two children; if a node i has two children j and k, then $X_i = X_j = X_k$; and if a node i has one child j, then either $|X_i| = |X_j| + 1$ and $X_j \subset X_i$ or $|X_i| = |X_j| - 1$ and $X_i \subset X_j$. It is not hard to transform a given tree decomposition into a nice tree decomposition.

Tree decomposition based algorithms usually proceed in two phases. First, given some input graph, a tree decomposition of bounded width is constructed. Second, one solves the given problem (such as Dominating Set) using dynamic programming. Here, we are only concerned with the second step. Dynamic programming to solve the optimization version of a problem again works in two phases—first bottom-up from the leaves to the root (which can be chosen arbitrarily) and then top-down from the root to the leaves in order to actually construct the solution. To do this two-phase dynamic programming, however, one has to store all dynamic programming tables, each of them corresponding to a bag of the tree decomposition. Each bag B usually leads to a table that is exponential in its size. For instance, the table size in case of Dominating Set is $3^{|B|}$ [1]. Clearly, even for modest bag sizes this leads to enormous memory consumption. This is the point where the anchor technique comes into play.

Trying to minimize space consumption, we have to obey that for *each* graph vertex we need to store important information. Hence, on the one hand, we want to minimize the memory space consumed by the tables and, on the other hand, we have to make sure that no information is lost. In a natural way, this leads to a special Weighted Set Cover problem: Each bag B translates into a set containing its vertices and an associated weight exponentially depending on its table size (for Dominating Set, this is $3^{|B|}$). Loosing no information then simply means that we have to cover the base set consisting of all graph vertices V by a selection of the "bag sets." To use as little memory as possible then means to do the selection of the bag sets such that their total weight is minimized while keeping each vertex from V covered. This is nothing but a Tree-Like Weighted Set Cover (TWSC) problem with exponential weight distribution. This formalization inspired theoretical work on TWSC with applications also in computational biology [5]. In our setting already simple data reduction rules suffice to obtain optimal solutions of TWSC in a fast way. We call the bags in the set cover *anchors.* After finding a set of anchors, we can simplify the decomposition tree with respect to the memory requirements by setting our pointers directly from anchor to anchor. That means,

Fomin's [4] measure tries to minimize the cost when assuming that all tables need to be stored whereas we try to avoid storing all tables.

grid graphs				path decomposition			nice path decomposition		
name	$\|V\|$	$\|E\|$	pw	nob	noa	mem	nob	noa	mem
`grid_10_10`	100	180	10	89	9	89.88%	179	17	93.26%
`grid_10_30`	300	560	10	289	27	90.66%	578	44	94.81%
`grid_10_50`	500	940	10	489	45	90.80%	978	58	95.98%

Table 4

We compare the memory savings for three different grid graphs, assuming a table size of $2^{|B|}$ for each bag B, as we also do in the subsequent tables. (Assuming size $3^{|B|}$ would even improve the saving effect.) Herein, pw denotes the width of the underlying path decomposition, nob denotes the number of bags, noa denotes the number of anchors, and mem denotes the percentage of memory saved.

$\|V\|$	$\|E\|$	nob	noa	maxb	maxa	nomb	av10b	al	noh	maxh	avh	mem
100	197	578.8	55.5	12.3	4.5	24.1	11.5	52.1	13.8	12.1	9.9	74.9 %
150	297	1023.9	85.1	16.8	4.9	25.9	15.2	80.3	21.5	16.7	12.0	73.5 %
200	397	1573.7	114	21.1	4.5	24.6	18.7	108.1	28.1	20.9	13.8	77.8%

Table 5

We compare graphs generated with the BRITE tool. Each row represents the average numbers taken over 15 graphs each time. We only consider nice tree decompositions here. Besides the figures used in Table 4, we additionally measured maxb (maximum bag size), maxa (maximum anchor size), nomb (number of maximum size bags), av10b (average size of the 10 percent biggest bags), al (number of anchors which are leaves), noh (number of help anchors, i.e., additional anchors found by the heuristic), maxh (size of maximum help anchor), and avh (average size of help anchors).

apart from the table involved in the current update process, we do not have to store any bag tables that are not anchors.

Note that with respect to path decompositions the goal of minimizing the overall memory consumption has a one-to-one correspondence to the optimization goal of the Path-like Weighted Set Cover problem as described above. Going to trees, however, the formalization as TWSC does not take into account the additional costs that are due to the modified pointer structure caused by the anchor technique (cf. Section 3).

3 Computational Analysis and Results

We performed some first empirical tests of the anchor technique on path and nice tree decompositions. To do so, we implemented some simple and efficient polynomial-time data reduction rules to find anchors. Having the anchors at hand, it then is rather simple to adapt the pointer structure between the dynamic programming tables. Without going into details, let us only mention that the algorithmic and implementation overhead for adding the pointer technique turned out to be negligible in terms of the overall running time of dynamic programming.[6] Thus, to start with a summary of our empirical findings, we may say that the anchor technique is easy to implement, it costs very little additional running time, and it leads to significant savings concerning memory use. We begin with first results on path decompositions of grid graphs.[7] In this case, we obtained the strongest memory saving of around 90–95% and more. Table 4 shows our results in more detail.

The problem with TWSC in case of trees instead of paths lies in the memory costs for the pointers (reflecting the (modified) tree structure) which are neglected in the TWSC

[6] We found this when doing tests with Dominating Set.

[7] Here an optimal path decomposition is an optimal tree decomposition as well.

$\|V\|$	$\|E\|$	nob	noa	maxv	maxa	nomv	av10v	al	noh	maxh	avh	mem
200	415	1595	84	44	6	6	36.2	74	13	42	28	87.2%
200	371	1459	88	38	6	8	32.1	73	16	38	28.5	63.4 %
200	425	1662	93	44	7	7	36.9	83	23	44	22.1	66.4%
200	372	1484	95	37	6	4	31.1	79	22	36	19.6	74.9%
200	410	1636	82	49	6	6	40.3	73	13	47	33.8	87.4%
200	415	1735	83	43	6	14	38.4	76	21	43	26.5	75.1%

Table 6

We compare random graphs generated with LEDA (single graphs, no average numbers). Again only nice tree decompositions are considered. Same figures as in Table 5.

formalization. In fact, formulating the problem in terms of the DOMINATING SET problem, to get the best savings concerning memory one should minimize the sum $\sum_{B\in A} 3^{|B|} \cdot |C|$, where A denotes the set of anchors and C denotes the set of "anchor children" of B in the tree decomposition. The second multiplicative factor is ignored in the formalization as TWSC and it turned out to be a crucial source of memory demands. To mitigate this effect, we designed a heuristic that extends an anchor set solution found by the TWSC optimization. In few words, the central objective of our heuristic is that we do not have to store a table (including its pointers) which has a memory consumption greater than the maximum memory consumption of a bag of the tree decomposition without anchors. We defer the details to the full paper. As to (nice) tree decompositions, we performed tests with so-called "Internet graphs" as produced by the BRITE topology generator and random graphs generated by the LEDA library. We achieved memory savings of around 80%, see Tables 5 and 6. The worst cases for the anchor technique seem to be tree decompositions with some big bags close to the root and lots of small anchor leaves. These difficulties also seem to be the reason that the technique so far not always achieves satisfactory improvements on memory consumption when dealing with "normal" (i.e., non-nice) tree decompositions. On the positive side, however, with anchors one can "always" afford to use nice tree decompositions instead of normal ones without loss of (memory) efficiency. Nice tree decompositions significantly simplify the dynamic programming as exhibited in the case of DOMINATING SET [1].

To summarize, the anchor technique significantly reduces the space consumption of dynamic programming on tree decompositions of graphs for solving (hard) optimization problems. Our technique not only applies to path/tree decomposition based dynamic programming but also appears to be useful for dynamic programming on branch decompositions. The time overhead caused by our method was negligible in all our experiments. In ongoing and future work, we try to extend our experiments to further classes of graphs. We will also try to further mitigate the gap between the formalization of the anchor idea in terms of TREE-LIKE WEIGHTED SET COVER and the practically relevant point of keeping the tree/table pointer structure reasonably simple.

References

[1] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, **33(4)**: 461–493, 2002.

[2] B. Aspvall, A. Proskurowski, and J. A. Telle. Memory requirements for table computations in partial k-tree algorithms. *Algorithmica* **27**: 382–394, 2000.

[3] H. L. Bodlaender. Treewidth: Algorithmic techniques and results. In *Proceedings 22nd MFCS*, Springer-Verlag LNCS 1295, pp. 19–36, 1997.

[4] H. L. Bodlaender and F. V. Fomin. Tree decompositions with small cost. In *Proceedings 8th SWAT*, Springer-Verlag LNCS 2368, pp. 378–387, 2002.

[5] J. Guo and R. Niedermeier. Exact algorithms for Tree-like Weighted Set Cover. Manuscript, submitted for publication, February 2004.

Clique, chromatic, and Lovász numbers of certain circulant graphs

Valentin E. Brimkov [1]

Department of Mathematics and Computer Science, SUNY Fredonia, NY, USA

Abstract

In this paper we first study some properties of circulant graphs of degree four. In particular, we determine their clique and chromatic numbers and address issues related to their Lovász theta function. We also consider generalizations of circulant graphs, called generalized circulants, that possess certain interesting properties, such as high connectivity and comparatively small clique number. Classes of generalized circulants may have an arbitrarily large gap between the clique and the chromatic number.

Key words: circulant graph, clique number, chromatic number, Lovász theta-function

1 Introduction

In this work we study properties of circulant graphs and their generalizations. Various applications of circulant graphs are known in counting and combinatorics, telecommunication networks, VLSI design, and distributed computing. Low-degree circulants provided a basis for some classical parallel and distributed systems [4, 13] as well as for certain data alignment networks for complex memory systems [14]. Specifically, circulant graphs of degree four have been used in the design of local networks and interconnection subsystems [3]. Recent work [2] presents a class of circulants of degree four with minimal topological distances. They have been used as a basis for constructing an optimal interconnection network for parallel computers with a very high degree of fault-tolerance [2] and for designing networks for massively parallel computers [15] and optimal VLSI [9]. Circulant graphs of degree two and four have also been involved (by the name of "spirographs") in studies on the structure of a digital line, which is the most fundamental primitive used in computer graphics and image analysis [6].

In this note we first determine the clique and the chromatic numbers of circulant graphs of degree four. We also address issues related to their Lovász theta function, known also

[1] E-mail: `brimkov@cs.fredonia.edu`. Part of this work has been done while the author was visiting Dipartimento di Elettronica ed Informatica, Università di Padova, Italy, and the International Centre for Theoretical Physics, Trieste, Italy.

as the Lovász number. The latter has been devised by Lovász [11] in order to estimate the Shannon capacity of a graph [12]. We also consider generalizations of circulant graphs, called generalized circulants, that possess certain interesting properties, such as high connectivity and comparatively small clique number. Classes of generalized circulants may have a large gap between the clique and the chromatic number. Most of the proofs are too lengthy to be reported here and will be included in the full paper.

2 Preliminaries

Here we recall some well-known graph-theoretic notions and introduce certain denotations used in the sequel. Let $G(V, E)$ be a simple graph. The *complement graph* of G is the graph $\overline{G}(V, \overline{E})$, where $\overline{E}$ is the complement of E to the set of edges of the complete graph on V. By $\omega(G)$ and $\chi(G)$ we denote the clique and the chromatic numbers of G, respectively. For any graph G we have $\omega(G) \leq \chi(G)$.

An $n \times n$ matrix $A = (a_{i,j})_{i,j=0}^{n-1}$ is called *circulant* if its entries satisfy $a_{i,j} = a_{0,j-i}$, where the subscripts belong to the set $\{0, 1, \ldots, n-1\}$ and are calculated modulo n. In other words, any row of a circulant matrix can be obtained from the first one by a number of consecutive cyclic shifts. A *circulant graph* is a graph with a circulant adjacency matrix. By $C_{n,j}$ we will denote a circulant graph of degree four, with vertex set $\{0, 1, \ldots, n-1\}$ and edge set $\{(i, i+1 \bmod n), (i, i+j \bmod n), i = 0, 1, \ldots, n-1\}$, where $1 < j \leq \frac{n-1}{2}$ is the *chord length.* See Fig. 1 for illustration.

Several equivalent definitions of the Lovász number of a graph are available [10]. One of them is the following. Given a graph G, let $\mathbf{A}$ be the family of matrices A such that $a_{ij} = 0$ if v_i and v_j are adjacent in G. Let $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A)$ be the eigenvalues of A. Then $\theta(A) = \max_{A \in \mathbf{A}}\{1 - \frac{\lambda_1(A)}{\lambda_n(A)}\}$. The function $\theta(G)$ is computable in polynomial time with arbitrary precision, although being "sandwiched" between $\omega(G)$ and $\chi(G)$, whose computation is NP-hard in the general case. More precisely, we have $\omega(G) \leq \theta(\overline{G}) \leq \chi(G)$. For various results, applications, and related references see the surveys by Knuth [10].

3 Basic contributions

3.1 Clique and chromatic numbers of $C_{n,j}$

We assume that $n \geq 6$, the cases for $n \leq 5$ being trivial. Then the clique and chromatic numbers of a circulant graph $C_{n,j}$ are determined as follows.

$$\omega(C_{n,j}) = \begin{cases} 3, & \text{if } j = 2, \text{ or if } n \text{ is odd and } j = \lfloor \frac{n}{2} \rfloor, \\ & \text{or if } n \text{ is divisible by 3 and } j = \frac{n}{3}; \\ 2, & \text{otherwise} \end{cases}$$

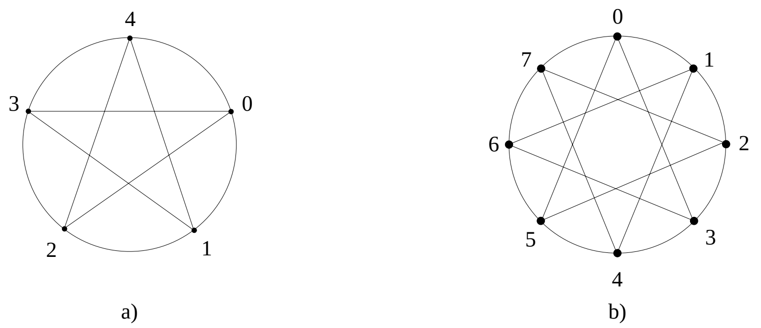


Figure 1. Two circulant graphs of degree 4. a) $C_{5,2}$, that is the well-known complete graph K_5; b) $C_{8,3}$.

$$\chi(C_{n,j}) = \begin{cases} 2, & \text{if } n \text{ is even and } j \text{ is odd;} \\ 4, & \text{if } n \text{ is divisible by 3 and } j=2, \text{ or if } n = F_k \text{ is an} \\ & \text{odd Fibonacci number and } j = F_{k-1} \text{ or } j = F_{k-2}; \\ 3, & \text{otherwise} \end{cases}$$

3.2 Lovász number of $C_{n,j}$

It has been noticed by several authors that the value of the Lovász number $\theta(\overline{G})$ is usually closer to $\omega(G)$ rather than to $\chi(G)$ (see, e.g., [7, 5]). Here we provide one more peculier example of a class of graphs with this property. More precisely, for any odd integer $n \geq 7$ we have that $3 < \theta(\overline{C}_{n,\lfloor \frac{n}{2} \rfloor}) < 3.3177\ldots$,[2] and as n increases, $\theta(\overline{C}_{n,\lfloor \frac{n}{2} \rfloor})$ strictly decreases and approaches the clique number $\omega(C_{n,\lfloor \frac{n}{2} \rfloor}) = 3$. To our knowledge, no convergence of this type has been exhibited so far.

The full journal version of the paper will also include an efficient $O(\max(j, n/j))$ algorithm for $\theta(C_{n,j})$ computation. The algorithm, whose description is too involved to be dicussed here, is based on sophisticated geometric arguments. It strongly outperforms the known algorithms for theta function computation whose complexity is of the order of $O(n^4)$ [1].

3.3 Generalized circulant graphs

We define a *generalized circulant* $G(k \times C_{n,J}, E)$ to be a graph which consists of k copies $C^i_{n,J}$, $i = 1, 2, \ldots, k$, of certain circulant graph $C_{n,J}$, and a set E of edges which interconnect vertices from different replicas of $C_{n,J}$. $J = \{j_1, j_2, \ldots, j_s\}$, $1 \leq s \leq n$, is a multi-index that describes the adjacencies in $C_{n,J}$. To define E, for every vertex x of $C^1_{n,J}$ one specifies all existing edges (if any) between x and vertices from $C^2_{n,J}, C^3_{n,J}, \ldots, C^k_{n,J}$. The same pattern of edges cyclically propagates between the corresponding vertices of $C^2_{n,J}$ and ones of $C^3_{n,J}, C^4_{n,J}, \ldots, C^k_{n,J}, C^1_{n,J}$, etc. See Fig. 2 for illustration.

[2] The upper bound 3.3177... is reached for $n = 7, j = 3$.

Varying the parameters defining a generalized circulant, one can obtain diverse classes of graphs with interesting properties. Some of them can be used in the design of communication networks with a high degree of fault-tolerance.

Generalized circulants are also interesting from theoretical perspective. For instance, for some their subclasses the gap between the clique and chromatic numbers can be arbitrarily large. As a simple example, consider a generalized circulant $G(k \times C_5, E)$, where C_5 is the pentagon and E contains all possible edges connecting vertices of different pentagons. It is not hard to see that $G(k \times C_5, E)$ has $5k$ vertices, $\frac{25k(k-1)}{2}$ edges, and $\omega(G(k \times C_5, E)) = 2k$, $\chi(G(k \times C_5, E)) = 3k$.

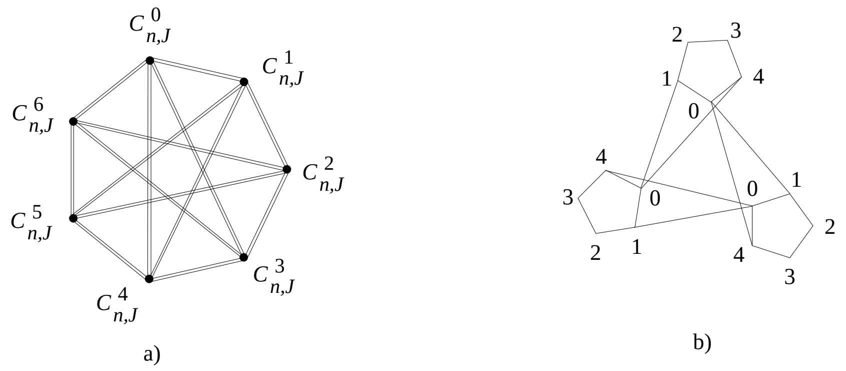


Figure 2. a) Exemplary structure of a generalized circulant $G(7 \times C_{n,J}, E)$. The double lines represent multiple edges between vertices from different copies of $C_{n,J}$. b) A generalized circulant $G(3 \times C_5, E)$. E contains 6 edges.

References

[1] Alizadeh, F. et al., SDPPACK user's guide, http://www.cs.nyu.edu/faculty/overton/sdppack/sdppack.html

[2] Beivide, R., E. Herrada, J.L. Balcázar and A. Arruabarrena, Optimal distance networks of low degree for parallel computers, *IEEE Trans. on Computers* **C-30**(10) (1991) 1109-1124

[3] Bermond, J.-C., F. Comellas and D.F. Hsu, Distributed loop computer networks: A survey, *J. of Parallel and Distributed Computing* **24** (1995) 2-10

[4] Bouknight, W.J., S.A. Denenberg, D.E. McIntyre, J.M. Randall, A.H. Samel and D.L. Slotnick, The Illiac IV System, Proc. IEEE **60**(4) (1972) 369-378

[5] Brimkov, V.E., B. Codenotti, V. Crespi and M. Leoncini, On the Lovász number of certain circulant graphs, In: *Algorithms and Complexity*, Bongiovanni, G., G. Gambosi, R. Petreschi (Eds.), LNCS No 1767 (2000) 291-305

[6] Dorst, L. and R.P.W. Duin, Spirograph theory: a framework for calculations on digitized straight lines, *IEEE Trans. Pattern Analysis and Machine Intelligence*, **6** (1984) 632-639

[7] Edwards, C.S., C.H. Elphik, Lower bounds for the clique and the chromatic numbers of a graph, *Discrete Applied Mathematics* **5** (1983) 51-64

[8] Feige U., Randomized graph products, chromatic numbers, and the Lovász θ-function, *Proc of the 27th STOC* (1995) 635-640

[9] Huber, K., Codes over tori, *IEEE Trans. on Information Theory* **43**(2) (1997) 740-744

[10] Knuth, D.E., The sandwich theorem, *Electronic J. Combinatorics*, **1** (1994) 1-48

[11] Lovász, L., On the Shannon capacity of a graph, *IEEE Trans. on Inf. Theory*, **25** (1979) 1-7

[12] Shannon, C.E., The zero-error capacity of a noisy channel, *IRE Trans. Inform. Theory* **IT-2** (1956) 8-19

[13] Wilkov, R.S., Analysis and design of reliable computer networks, *IEEE Trans. on Communications* **20** (1972) 660-678

[14] Wong, C.K. and D. Coppersmith, A combinatorial problem related to multimodule memory organization, *Journal of the ACM* **21**(3) (1974) 392-402

[15] Yang, Y., A. Funashashi, A. Jouraku, H. Nishi, H. Amano and T. Sueyoshi, Recursive diagonal torus: an interconnection network for massively parallel computers, *IEEE Trans. on Parallel and Distributed Systems* **12**(7) (2001) 701-715

An improved local search algorithm for 3-SAT

Tobias Brueggemann[1] Walter Kern[2]

Department of Applied Mathematics, University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract

We slightly improve the pruning technique presented in Dantsin et. al. (2002) to obtain an $\mathcal{O}^*(1.473^n)$ algorithm for 3-SAT.

Key words: exact algorithm, local search, 3-SAT
MSC2000: 68Q25

1 Introduction

An instance of 3-SAT is a boolean formula φ in n variables $x_1, \ldots, x_n$, defined as the conjunction of a set $\mathcal{C}$ of disjunctive clauses of length at most 3. Satisfiability of φ can be tested in a straightforward manner in time

$$\mathcal{O}\left(2^n \cdot n^3\right) = \mathcal{O}^*\left(2^n\right).$$

Here, as usual, we use the $\mathcal{O}^*$-notation to indicate that polynomial factors are suppressed.

During the last years so-called *exact algorithms* have been designed solving 3-SAT in time $\mathcal{O}^*(\alpha^n)$ with $\alpha < 2$, see Schoening [3] for an overview. The currently fastest randomized algorithms run in time $\mathcal{O}^*(1.3302^n)$ (see Hofmeister, Schoening, Schuler and Watanabe [2]) and the fastest deterministic algorithm (see Dantsin et. al. [1]) takes $\mathcal{O}^*(1.481^n)$. We slightly improve the pruning technique used in Dantsin et. al. [1] to obtain a running time of $\mathcal{O}^*(1.473^n)$.

2 Local search

Let φ be an instance of 3-SAT given by a set $\mathcal{C}$ of clauses in variables $x_1, \ldots, x_n$. For $a \in \{0,1\}^n$ let $B_r(a) \subseteq \{0,1\}^n$ denote the set of 0-1 vectors with Hamming distance at most r

[1] E-mail: `t.brueggemann@math.utwente.nl`. Supported by the Netherlands Organization for Scientific Research (NWO) grant 613.000.225 (Local Search with Exponential Neighborhoods).
[2] E-mail: `kern@math.utwente.nl`. Corresponding author.

from a. The currently fastest algorithms for 3-SAT are based on *local search*: First, a *covering code* of suitable *radius* $r \leq n$ is constructed, i.e. a set $A \subseteq \{0,1\}^n$ such that

$$\{0,1\}^n = \bigcup_{a \in A} B_r(a)$$

holds. Next we search for a truth assignment for φ in each $B_r(a)$, $a \in A$, separately. To make our paper self-contained, we briefly describe the basic idea for constructing a covering code and (to some extent) the local search within a given $B_r(a)$ as presented in Dantsin et. al. [1].

Covering codes. As $B_r := B_r(0)$ contains exactly

$$V(n,r) = \sum_{i=1}^{r} \binom{n}{i}$$

elements, a covering code $A \subseteq \{0,1\}^n$ of radius $r \leq n$ must necessarily satisfy

$$|A| \geq \frac{2^n}{V(n,r)}.$$

Covering codes of approximately this size indeed exist and can be constructed randomly: Choose

$$t = \frac{n2^n}{V(n,r)}$$

elements from $\{0,1\}^n$ uniformly at random, resulting in a set $A \subseteq \{0,1\}^n$ of size $|A| \leq t$. The probability that a particular $a^* \in \{0,1\}^n$ is *not* covered by any $B_r(a)$, $a \in A$ is at most

$$P[a^* \text{ not covered}] = \left(1 - \frac{V(n,r)}{2^n}\right)^t \leq \mathrm{e}^{-n},$$

using $1 + x \leq \mathrm{e}^x$ for $x \in \mathbb{R}$. So the probability that A is *not* a covering code is at most $2^n \mathrm{e}^{-n}$, which tends to 0 as $n \to \infty$.

This procedure can be de-randomized by taking in each step a new code word $a \in \{0,1\}^n$ that is best possible in the sense that it covers as many as possible of the yet uncovered elements in $\{0,1\}^n$. Note, however, that this *greedy construction* takes $\mathcal{O}^*(2^n)$ per step and thus almost $\mathcal{O}(2^{2n}) = \mathcal{O}^*(4^n)$ in total (which is far too slow). Dantsin et. al. [1] therefore propose the following. Let $K \in \mathbb{N}$ be a constant and assume w.l.o.g. that $n = Kn_0$ and $r = Kr$. Then construct a covering code $A_0 \subseteq \{0,1\}^{n_0}$ in time $\mathcal{O}(4^{n_0}) = \mathcal{O}^*\left(\sqrt[K]{4}^n\right)$ and take

$$A = \underbrace{A_0 \times \ldots \times A_0}_{K \text{ times}}$$

as a covering code for $\{0,1\}^n$. Proceeding this way, the time needed for constructing the covering code becomes negligible.

Local search. Assume we want to search for a truth assignment for φ in $B_r(a) \subseteq \{0,1\}^n$. We may assume w.l.o.g. that $a = 0$, i.e., we search in $B_r = B_r(0)$. (Interchange x_i with $\overline{x}_i$ if necessary.) If $a = 0$ is not a truth assignment for φ, there must exist a *false clause*, i.e. a clause $C \in \mathcal{C}$ that is false under $a = 0$, say $C = (x_i \vee x_{i'} \vee x_{i''})$. It then suffices to search for a truth assignment in $B_{r-1} \subseteq \{0,1\}^{n-1}$ w.r.t. each of the formulae

$$\varphi_1 = \varphi[x_i = 1], \; \varphi_2 = \varphi[x_{i'} = 1] \text{ and } \varphi_3 = \varphi[x_{i''} = 1],$$

obtained by fixing a variable as indicated in brackets. If necessary, we may even fix in addition some variables to zero, e.g., define $\varphi_1 := \varphi[x_i = 1], \varphi_2 := \varphi[x_{i'} = 1, x_i = 0]$ and $\varphi_3 := \varphi[x_{i''} = 1, x_i = 0, x_{i'} = 0]$.

Continuing this way, our search can be described by a *search tree* T_r, constructed by *branching on false clauses* (one false clause per node), as indicated in figure 1.

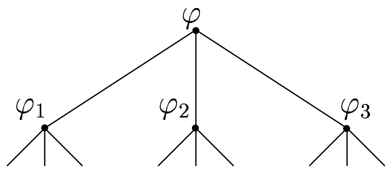


Figure 1. The search tree T_r

Needless to say that we never branch to formulas $\varphi' = \varphi[x_i = 1, \ldots]$ that are obviously non-satisfiable because they contain an empty (non-satisfiable) clause. (For example, if $(\overline{x}_i) \in \mathcal{C}$, we would only branch to φ_2 and φ_3 in figure 1.) We denote the number of leaves of T_r by $|T_r|$ and refer to it as the *size* of T_r. Clearly,

$$|T_r| \leq 3^r \tag{1}$$

holds, an immediate consequence of the recursion $|T_r| \leq 3|T_{r-1}|$ (see figure 1). In case φ contains a false 2-clause $C \in \mathcal{C}$, then branching on C would yield $|T_r| \leq 2|T_{r-1}|$.

As pointed out in Dantsin et. al. [1], this simple argument already gives an $\mathcal{O}^*\left(\sqrt[2]{3}^n\right) \approx \mathcal{O}^*(1.7321^n)$ algorithm: Take $r = \frac{n}{2}$ and search $B_r(0)$ and $B_r(1)$ separately in time $\mathcal{O}^*(3^r) = \mathcal{O}^*\left(\sqrt[2]{3}^n\right)$ each.

Smaller search trees. The trivial bound (1) on the size of the search tree can be improved by a clever branching technique, as shown in Dantsin et. al. [1]: Assume that φ contains three pairwise disjoint false clauses $C = (x_i \vee x_{i'} \vee x_{i''}), C_1 = (x_j \vee x_{j'} \vee x_{j''})$ and $C_1' = (x_k \vee x_{k'} \vee x_{k''})$ and a (true) clause $(\overline{x}_i \vee \overline{x}_j \vee \overline{x}_k)$. We may then *branch along* $(\overline{x}_i \vee \overline{x}_j \vee \overline{x}_k)$, i.e. first branch on C at the root node φ, then branch on C_1 at $\varphi_1 = \varphi[x_i = 1]$ and finally branch on C_1' at $\varphi_1' = \varphi_1[x_j = 1] = \varphi[x_i = 1, x_j = 1]$. The resulting search tree is indicated in figure 2.

Note that the node corresponding to φ_1' has only two descendants because $\varphi[x_i = 1, x_j = 1, x_k = 1]$ is ruled out by the clause $(\overline{x}_i \vee \overline{x}_j \vee \overline{x}_k)$.

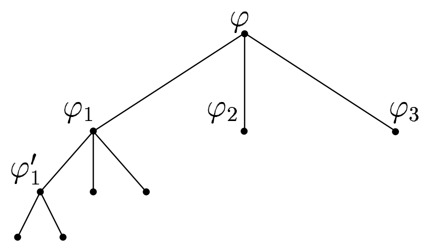


Figure 2. Branching along $(\overline{x}_i \vee \overline{x}_j \vee \overline{x}_k)$

If a similar branching was possible also at φ_2 and φ_3, we would get a search tree satisfying a recursion

$$|T_r| \leq 6|T_{r-2}| + 6|T_{r-3}|. \tag{2}$$

Indeed, this is what Dantsin et. al. [1] show. Assuming inductively that $|T_k| \leq c\alpha^k$ holds for some constant $c > 0$, (2) implies that

$$|T_r| \leq \mathcal{O}\left(\alpha^r\right), \tag{3}$$

where $\alpha = \sqrt[3]{4} + \sqrt[3]{2} \approx 2.848$ is the largest root of $\alpha^3 - 6\alpha - 6 = 0$.

The main result of our paper slightly improves this bound as follows.

Theorem 1 *By branching on false clauses we can ensure that*

$$|T_r| \leq c\beta^r,$$

where $\beta = \frac{1+\sqrt{21}}{2} \approx 2.792$ *is the largest root of* $\beta^3 - 6\beta - 5 = 0$.

Running time. Let $\varrho < \frac{1}{2}$ and $r = \varrho n$. By Stirling's formula, the size of a covering code we construct is (up to a polynomial factor) bounded by

$$|A| = \mathcal{O}^*\left(\left[2\varrho^\varrho\left(1-\varrho\right)^{1-\varrho}\right]^n\right).$$

According to (3), the number of nodes in T_r is bounded by $n|T_r| = \mathcal{O}^*\left(|T_r|\right)$ and hence the total running time is thus bounded by

$$\mathcal{O}^*\left(|A||T_r|\right) = \mathcal{O}^*\left(\left[2(\alpha\varrho)^\varrho\left(1-\varrho\right)^{1-\varrho}\right]^n\right).$$

This expression is minimal for $\varrho \approx 0.26$, yielding the bound of $\mathcal{O}^*\left(1.481^n\right)$ in Dantsin et. al. [1].

Similarly, replacing α by β from Theorem 1, we obtain for $\varrho \approx 0.264$ an exact algorithm that runs in $\mathcal{O}^*\left(1.473^n\right)$.

References

[1] E. Dantsin, A. Goerdt, E.A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, O. Raghavan, U. Schoening [2002]: *A deterministic* $(2-2/(k+1))^n$ *algorithm for k-SAT*

based on local search. In: Theoretical Computer Science 289 (2002), 69-83. Elsevier Science B.V.

[2] T. Hofmeister, U. Schoening, R. Schuler, O. Watanabe [2002]: *A Probabilistic 3-SAT Algorithm Further Improved.* In: H. Alt, A. Ferreira (Eds.): STACS 2002, LNCS 2285, 192-202. Springer-Verlag Berlin Heidelberg.

[3] U. Schoening [2002]: *A Probabilistic Algorithm for k-SAT Based on Limited Local Search and Restart.* In: Algorithmica 32 (2002), 615-623. Springer-Verlag New York Inc.

Clique algorithms for classifying substructures in generalized quadrangles

M. Cajkova [1], V. Fack [2]

Research Group on Combinatorial Algorithms and Algorithmic Graph Theory, Department of Applied Mathematics and Computer Science, Ghent University, Krijgslaan 281-S9, B-9000 Ghent, Belgium.

1 Introduction

We denote an undirected graph by $G = (V, E)$, where V is the set of vertices and E is the set of edges. Two vertices are said to be *adjacent* if they are connected by an edge. A *clique* of a graph is a set of vertices, all of which are pairwise adjacent. The *maximum clique problem* asks for cliques of maximum cardinality in a graph; this problem is well-known to be NP-complete [2].

A (finite) *generalized quadrangle* (GQ) is an incidence structure $S = (P, B, I)$ in which P and B are disjoint (nonempty) sets of objects called points and lines (respectively), and for which I is a symmetric point-line incidence relation satisfying the following axioms:

(i) each point is incident with $1 + t$ lines ($t \geq 1$) and two distinct points are incident with at most one line;
(ii) each line is incident with $1 + s$ points ($s \geq 1$) and two distinct lines are incident with at most one point;
(iii) if x is a point and L is a line not incident with x, then there is a unique pair $(y, M) \in P \times B$ for which $x \ I \ M \ I \ y \ I \ L$.

The integers s and t are the *parameters* of the GQ and S is said to have order (s, t); if $s = t$, S is said to have *order* s. Generalized quadrangles with $s > 1$ and $t > 1$ are called *thick* . Interchanging points and lines in S yields a GQ of order (t, s), which is called the *dual* S^D of S. Two points p and q of S are called *collinear* if there is a line L in B incident with both. Dually, two lines L and M are called *concurrent* if there is a point p in P incident with both. For the theory of generalized quadrangles we refer to [3].

With a generalized quadrangle S a so-called *collinearity graph* G_S can be associated as follows: the points of S correspond to the vertices of G_S, and two vertices are adjacent if and only if the corresponding points are collinear.

[1] E-mail: `Miroslava.Cajkova@UGent.be`.
[2] E-mail: `Veerle.Fack@UGent.be`.

An *ovoid* of a generalized quadrangle S is a set O of points of S such that each line of S is incident with a unique point of O. Dually, a *spread* of S is a set R of lines of S such that each point of S is incident with a unique line of R.

An ovoid of S corresponds to a maximum coclique of size $st+1$ in G_S (or equivalently, a maximum clique in its complement $\overline{G_S}$). A spread of S corresponds to a maximum coclique of size $st+1$ in the collinearity graph of S^D. If no coclique of the required size exists, then the maximum coclique is said to be a maximal partial ovoid or spread in S.

Here we will consider the classical thick generalized quadrangles:

- The *generalized quadrangle* $Q(4,q)$ and $Q^-(5,q)$:
  Let Q resp. Q^- be a non-singular quadric of projective index 1 in the projective space $PG(4,q)$ resp. $PG(5,q)$. Then the points of quadric together with the lines of the quadric form form a GQ with parameters $(s,t)=(q,q)$ resp. (q,q^2).
- The *symplectic generalized quadrangle* $W(q)$:
  The points of the projective space $PG(3,q)$, together with the totally isotropic lines with respect to a symplectic polarity, form a GQ with parameters $(s,t)=(q,q)$. Note that $W(q)$ is isomorphic to the dual of $Q(4,q)$. Moreover, $Q(4,q)$ (or $W(q)$) is selfdual iff q is even.
- The *generalized quadrangle* $H(3,q^2)$ and $H(4,q^2)$:
  Let H be a nonsingular hermitian variety of the projective space $PG(3,q^2)$ resp. $PG(4,q^2)$. Then the points of H together with the lines on H form a GQ with parameters $(s,t)=(q^2,q)$ resp. (q^2,q^3). Note that $H(3,q^2)$ is isomorphic to the dual of $Q^-(5,q)$.

In this paper we use standard maximum clique algorithms and add pruning strategies based on specific properties of the collinearity graphs in order to determine (partial) ovoids or spreads of classical generalized quadrangles.

2 Algorithms

The basic form of most published algorithms (e.g. [1]) for the maximum clique problem is a backtracking search which tries to extend a partial clique by adding the vertices of a set A of allowed remaining vertices in a systematic way. Pruning strategies are used to avoid going through every single clique of the graph. Typical pruning strategies involve vertex colorings. In a vertex coloring, adjacent vertices must be assigned different colors, so if a graph or an induced subgraph can be colored with c colors, then the graph or subgraph cannot contain a clique of size $c+1$. Recently Östergård [5] presented a new maximum clique algorithm that allows to introduce a new pruning strategy.

Taking into account the special structure of the graphs considered here allows to add some specific pruning strategies to the standard algorithms.

A straightforward and effective way of reducing the search space consists in fixing two points of the ovoid, as follows. We call two ovoids equivalent if there is an automorphism of S that transforms one ovoid into the other. It can be proven that a classical generalized quadrangle is transitive on the pairs of noncollinear points. Hence, for every ovoid not containing a certain noncollinear pair, there is an equivalent ovoid that does contain that pair. It follows that we can restrict the search to ovoids that contain a certain fixed pair of noncollinear points without missing any of the essentially different ovoids. In terms of the clique finding algorithm in $\overline{G_S}$, this means that we can restrict the search to cliques containing a certain fixed edge. This reduces the search space with a factor of $O(vk)$, where $v = (s+1)(st+1)$ is the number of points of S and $k = s^2t$ is the number of points not collinear with an arbitrary point.

For some types of generalized quadrangles optimal vertex colorings of the collinearity graph can be constructed from theoretical arguments. For instance, classical constructions for ovoids in $Q(4,q)$ are known [3]. Since $Q(4,q)$ is isomorphic to the dual of $W(q)$, the points of an ovoid in $Q(4,q)$ correspond to lines in $W(q)$, hence to a partition of the vertices of $\overline{G_{W(q)}}$ into classes of mutually non-adjacent vertices. In other words, this is a partition of $\overline{G_{W(q)}}$ into color classes that can be used for pruning and/or vertex ordering in a maximum clique algorithm. It can be proven that the obtained vertex coloring is optimal, i.e., uses a minimum number of colors.

The following strategy is useful when checking whether a quadrangle has an ovoid or when classifying all ovoids in a quadrangle known to contain ovoids. Next to the clique finding algorithm that works in the collinearity graph of the quadrangle, we also maintain the incidence structure of points and lines of the quadrangle and use this information as follows. In the process of adding a vertex to the current clique, it can happen that there is a line with only a single point corresponding to a vertex that is still allowed. In that case this point must belong to the ovoid, which means that the corresponding vertex must be added to the current clique.

3 Results

It is known (see [3]) that $Q(4,q)$ always has ovoids of size q^2+1. Using the above mentioned strategies we obtained a full classification for $Q(4,q)$ for $q \leq 9$. This confirms earlier theoretical results, including an unpublished result of a computer search by T. Penttila classifying the two non-equivalent ovoids for $Q(4,9)$. The result for $q = 9$ is shown in Table 1.

Table 7
Quadrangles with ovoids

GQ	#points	ovoid size	#cliques	$\lvert Aut\rvert$
$Q(4,9)$	820	82	2	2125440, 5184

For $W(q)$ (q odd), $Q(5,q)$ and $H(4,q^2)$, it is known that no ovoids exist. Some theoretical upper bounds for the maximal partial ovoid or spread size are known. Table 2 gives some results on partial ovoids obtained by our programs. We have obtained the exact size of the partial ovoids (for some of them also complete classification), which improve on the best known theoretical bounds.

We also confirmed Brouwer's unpublished result that $H(4,4)$ has no spread (i.e., $H(4,4)^D$ has no ovoid).

Table 8
Quadrangles without ovoids

GQ	#points	bound	exact size	# maximum cliques	$\|Aut\|$
$W(5)$	156	21	18	2	12, 72
$Q(5,3)$	112	21	16	1	11520
$Q(5,4)$	325	37	25		
$H(4,4)$	165	25	21	1	648
$H(4,4)^D$	297	32	29	6	

References

[1] R. Carraghan and P.M. Pardalos, *An exact algorithm for the maximum clique problem*, Oper. Res. Lett. **9** (1990) 375-382.

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

[3] S.E. Payne and J. Thas, *Finite Generalized Quadrangles*, Pitman Res. Notes Math. Ser. **110**, Longman, 1984.

[4] G.Hanssens and H. Van Maldeghem, *A new look at the classical generalized quadrangles*, Ars Combinatorica **24** (1987) 199-210.

[5] P.J. Östergård, *A fast algorithm for the maximum clique problem*, Discrete Applied Mathematics **120** (2002) 197-207.

Two-Levels-Greedy: a generalization of Dijkstra's shortest path algorithm

Domenico Cantone [1] Simone Faro [2]

Department of Mathematics and Computer Science, University of Catania, Viale A. Doria 6, 95125, Catania, Italy

The shortest path problem on weighted directed graphs is one of the basic network optimization problems. Its importance is mainly due to its applications in various areas, such as communication and transportation. Here we are interested in the single-source case. When the graph is not required to satisfy any particular restriction and negative weight edges can occur, the problem is solved by the Bellman-Ford-Moore algorithm [Bel58, For56, Moo59], whose complexity is $\mathcal{O}(|V||E|)$, with V and E denoting the sets of nodes and of edges, respectively. A more efficient solution due to Dijkstra [Dij59] is available when weights are restricted to non-negative values. Depending on the implementation used for maintaining a service priority queue, Dijkstra's algorithm has complexity $\mathcal{O}(|V|^2)$ (simple list), or $\mathcal{O}(|E|\log|V|)$ (standard binary heap), or $\mathcal{O}(|V|\log|V|+|E|)$ (Fibonacci heap [FT87]). Another case which can be solved very efficiently occurs when the underlying graph is acyclic. In such a case, by scanning the nodes in topological ordering, one can achieve a $\mathcal{O}(|V|+|E|)$ complexity.

In this note we present a natural generalization of Dijkstra's algorithm to the case in which negative weight edges are allowed, but only outside of any cycle. The resulting algorithm turns out to have the same asymptotic complexity of Dijkstra's algorithm and shows a linear behavior in the case of acyclic graphs. In fact, we will also see that our proposed algorithm compares very well in practice with the most efficient shortest path algorithms available, such as the ones due to Dial [Dia69], Pape [Pap74], Pallottino [Pal84], Glover *et. al.* [GGK84], and to Goldberg and Radzik [GR93].

1 Preliminaries

We begin by reviewing the relevant notations and terminology. A *directed graph* is represented as a pair $G = (V, E)$, where V is a finite set of nodes and $E \subseteq V \times V$ is a set of edges such that E does not contain any self-loop of the form (v, v). A *weight function* l on $G = (V, E)$ is any real function $l : E \to \mathbb{R}$. A *path* in $G = (V, E)$ is any finite sequence $(v_1, v_2, \dots, v_k)$ of nodes such that (v_i, v_{i+1}) is an edge of G, for $i = 1, \dots, k-1$. The weight function can be naturally extended over paths by putting $l(v_1, v_2, \dots, v_k) = \sum_{i=1}^{k-1} l(v_i, v_{i+1})$. A *minimum*

[1] E-mail: `cantone@dmi.unict.it, http://www.dmi.unict.it/~cantone`.
[2] E-mail: `faro@dmi.unict.it, http://www.dmi.unict.it/~faro`.

weight path (or shortest path) from u to v is a path in $G = (V, E)$ whose weight is minimum among all paths from u to v. Provided that v is reachable from u and that no path from u to v goes through a negative weight cycle, a minimum weight path from u to v exists; in such a case we denote by $\delta(u, v)$ the minimum weight of a path from u to v. If v is not reachable from u, we put $\delta(u, v) = +\infty$. Finally, if there is a path from u to v through a negative weight cycle, we put $\delta(u, v) = -\infty$. The function $\delta : V \times V \to \mathbb{R} \cup \{+\infty, -\infty\}$ is called the *distance function* on (G, l). Given a source node s in a graph G, the *single-source shortest path problem* from s is the problem of finding the minimum weight paths from s to all other nodes of G or to ascertain the existence of a negative weight cycle in G reachable from s.

Most single-source shortest path algorithms are based on the *labeling* method, which maintain a *potential* function $d : V \to \mathbb{R} \cup \{+\infty\}$, a *predecessor* function $\pi : V \to V \cup \{\text{NIL}\}$, and a *status* function $S : V \to \{\text{UNREACHED, LABELED, SCANNED}\}$. Initially, one puts $d(s) := 0$, $\pi(s) := \text{NIL}$, $S(s) := \text{LABELED}$, where s is the source node, and also puts $d(v) := +\infty$, $\pi(v) = \text{NIL}$, and $S(v) := \text{UNREACHED}$, for $v \in V \setminus \{s\}$ (procedure INITIALIZE). Subsequently, the potential function d is updated only by assignments of the form $d(v) := d(u) + l(u, v)$, provided that $d(v) > d(u) + l(u, v)$, in which case one also puts $\pi(v) := u$ and $S(v) :=$ LABELED. It turns out that $d(v) \geq \delta(s, v)$ always holds, for $v \in V$. Additionally, if $d(v) = \delta(s, v)$, then the predecessor function π can be used to reconstruct a shortest path from s to v backwards. The values $d(v)$ are updated within SCAN operations (see below).

```
PROCEDURE INITIALIZE($G$, $s$)
    for all $v \in V$ do
        $d(v) := +\infty$
        $\pi(v) :=$ NIL
        $S(v) :=$ UNREACHED
    $d(s) := 0$
    $S(s) :=$ LABELED
```

```
PROCEDURE SCAN($G, u$)
    for all $v \in V$ such that $(u, v) \in E$ do
        if $d(v) > d(u) + l(u, v)$ then
            $d(v) := d(u) + l(u, v)$
            $S(v) :=$ LABELED
            $\pi(v) := u$
    $S(u) :=$ SCANNED
```

Procedure SCAN is called on LABELED nodes until all nodes are marked either UNREACHED or SCANNED. Notice that after a SCAN operation is called on a LABELED node u, some UNREACHED or SCANNED node may become LABELED, whereas the node u becomes SCANNED.

Shortest path algorithms based on the labeling method are mainly characterized by the strategy they adopt to select the next node to be scanned from the set Q of all LABELED nodes. For instance, the Bellman-Ford-Moore algorithm maintains the set of LABELED nodes into a FIFO queue. Thus, the next node to be scanned is removed from the head of the queue, whereas a node that becomes LABELED is added to the tail. As another example, Dijkstra's algorithm applies a greedy strategy, which consists in selecting at each iteration a LABELED node $v \in Q$ with the *minimum* potential value $d(v)$. It turns out that Dijkstra's algorithm is very efficient for graphs containing no negative weight edges, but it may run in exponential time if they are present.

```
TWO-LEVELS-GREEDY($G$, $s$)
    compute the component graph $G^{SCC} = (V^{SCC}, E^{SCC})$
    compute a topological ordering $<$ of $G^{SCC}$
    INITIALIZE($G$,$s$)
    while $G$ contains some node marked LABELED do
        let $\mathbf{C} \in E^{SCC}$ be the $<$-smallest s.c.c. containing a LABELED node
        let $v$ be a $d$-smallest LABELED node in $\mathbf{C}$
        SCAN($G, v$)
```

Figure 1. The TWO-LEVELS-GREEDY algorithm

2 The TWO-LEVELS-GREEDY algorithm

The algorithm which we propose is a natural generalization of Dijkstra's algorithm to the case in which negative weight edges are allowed, but only outside of any cycle (*negative weight edge restriction*). It consists in a *preliminary phase* and a *scanning phase.*
Given a directed graph $G = (V, E)$, with weight function l and satisfying the above negative weight edge restriction, as a first preliminary step we compute the graph $G^{SCC} = (V^{SCC}, E^{SCC})$ of the strongly connected components (s.c.c.) of G. We recall that V^{SCC} is the partition of V with respect to the relation $\sim$ over V, where $u \sim v$ holds if and only if u and v lie on a same cycle, and E^{SCC} is the collection of pairs $(\mathbf{C}_1, \mathbf{C}_2)$ such that there exists an edge $(v_1, v_2) \in E$, with $v_1 \in \mathbf{C}_1$ and $v_2 \in \mathbf{C}_2$. It turns out that the graph G^{SCC} can be computed in $\mathcal{O}(|V| + |E|)$-time (cf. [CLR90]); moreover, G^{SCC} is acyclic and, because of the negative weight edge restriction, negative weight edges can connect only nodes belonging to different components. Next, again in $\mathcal{O}(|V| + |E|)$-time (cf. [CLR90]), we compute a *topological ordering* of G^{SCC}. This is any linear ordering $<$ of V^{SCC} such that if $(\mathbf{C}_1, \mathbf{C}_2) \in E^{SCC}$ then $\mathbf{C}_1 < \mathbf{C}_2$. Finally, we complete the preliminary phase by executing the procedure INITIALIZE on the graph G with a given source s. In the scanning phase, nodes are selected to be processed by procedure SCAN according to the following "two-levels" greedy strategy, until there is no node which is marked LABELED (for convenience, a s.c.c. $\mathbf{C}$ containing a LABELED node is said to be LABELED as well):

- firstly, the LABELED s.c.c. $\mathbf{C} \in E^{SCC}$ which is smallest with respect to the topological ordering $<$ is selected;
- secondly, a LABELED node v in $\mathbf{C}$ with minimal potential $d(v)$ is selected to be scanned.

We name the resulting algorithm TWO-LEVELS-GREEDY (TLG, for short). Its pseudo-code is shown in Fig. 2. It may be shown that under the above restriction on negative weight edges the TLG algorithm computes correctly all shortest-paths from a given source s in G. Moreover, it turns out that the TLG algorithm retains the same asymptotic time complexity as Dijkstra's algorithm. In addition, when the input graph is acyclic, its complexity is linear in the size of the graph.

3 Experimental results

A significant amount of experimental testing on shortest path algorithms has been carried out over the years, see for instance [Ber93], [CG99], [CGR96], and [MCN91]. To this purpose, several classes of graphs have been introduced. We present below experimental results relative to two extreme classes of graphs for the TLG algorithm: RAND-LEN, a class of strongly connected and dense random graphs, and ACYC-P2N, a class of acyclic random graphs with a variable fraction of negative edges (see [CGR96] for more details). All algorithms have been implemented in the **C** programming language and have been tested on a PC with AMD Athlon processor of 1.19 GHz. For each test, we computed the running time in CPU milliseconds (in bold) and the average number of scan operations per node. Maintaining the same style of [CGR96], each entry is the average of five runs of the code on problem instances produced with the same generator parameters, except for the pseudo-random generator seed. For each problem, the two best results have been underlined.

The TLG algorithm has been tested against the following algorithms: the Bellman-Ford-Moore algorithm (BFM) and one of its variants (BFP), which implements the *parent-checking* heuristic introduced in [CGR96]; Dijkstra's algorithm implemented with bucket heaps (DIKB), as proposed by Dial [Dia69]; two incremental algorithms due to Pape (PAPE) [Pap74] and Pallottino TWOQ) [Pal84]; the threshold algorithm (THRESH) due to Glover *et. al.* [GGK84]; two topological sorting algorithms (GOR and GOR1) due to Goldberg and Radzik [GR93]. The priority queue of the TLG algorithm has been implemented with bucket heaps.

The first table presents experimental results on the RAND-LEN class of graphs, where each graph is constructed by first creating a Hamiltonian cycle and then adding edges with distinct end points. In our experiments we set to 1 the weight of the edges on the Hamiltonian cycle whereas the weights of the remaining edges have been randomly selected in the interval $[L, U]$, using a uniform distribution.

$[L,U]$	BFM	BFP	DIKB	PAPE	TWOQ	THRESH	GOR	GOR1	TLG
$[1,1]$	**215**	**216**	**218**	**214**	**224**	**346**	**347**	**917**	**330**
	1.00	1.00	1.00	1.00	1.00	1.00	1.61	4.48	1.00
$[0,10]$	**671**	**601**	**265**	**506**	**509**	**367**	**779**	**1124**	**360**
	3.17	2.67	1.00	2.85	2.84	1.01	4.36	5.94	1.00
$[0,10^2]$	**1534**	**1354**	**293**	**1171**	**1229**	**705**	**1376**	**1400**	**390**
	7.40	6.14	1.00	8.07	8.05	1.78	8.93	8.21	1.00
$[0,10^4]$	**3867**	**3463**	**339**	**3360**	**3693**	**2738**	**2573**	**2101**	**468**
	19.04	16.74	1.00	27.31	26.07	9.16	19.48	14.15	1.00
$[0,10^8]$	**5564**	**5219**	**352**	**5500**	**5584**	**4096**	**3527**	**1520**	**478**
	29.21	26.76	1.00	47.60	41.89	16.03	27.96	12.26	1.00

The second table shows experimental results obtained on the class ACYC-P2N of acyclic random graphs. In a graph of this class all edge weights are selected uniformly from the interval $[L, U]$, where the values of $L < 0$ and $U > 0$ determine the expected fraction $f = -L/(U - L)$ of negative weight edges.

f (%)	BFM	BFP	DIKB	PAPE	TWOQ	THRESH	GOR	GOR1	TLG
0	**52**	**52**	**44**	**42**	**44**	**38**	**76**	**40**	**80**
	1.76	1.66	1.00	1.83	1.83	1.01	2.61	2.00	1.00
10	**72**	**64**	**44**	**56**	**62**	**42**	**84**	**40**	**240**
	2.43	2.21	1.14	2.64	2.64	1.29	3.01	2.00	1.00
20	**167**	**146**	**134**	**134**	**140**	**123**	**125**	**40**	**250**
	8.41	6.78	7.28	10.92	10.53	7.06	6.29	2.00	1.00
30	**731**	**542**	**771**	**516**	**488**	**619**	**195**	**40**	**270**
	40.9	28.75	41.31	69.27	59.69	40.28	12.49	2.00	1.00
40	**4702**	**3402**	**5666**	**5161**	**3304**	**4707**	**351**	**40**	**285**
	288.7	179.65	325.83	940.89	641.19	339.58	22.81	2.00	1.00
50	**32692**	**19496**	**43372**	**67320**	**29774**	**37842**	**485**	**40**	**300**
	2171.6	1056.25	2405.31	12088.96	9510.17	2862.66	30.81	2.00	1.00
60	-	**39761**	-	-	**53357**	**83537**	**587**	**38**	**320**
	-	2149.88	-	-	20124.87	5918.05	33.52	2.00	1.00
100	-	**44185**	-	-	**25967**	-	**40**	**35**	**386**
	-	2349.75	-	-	10819.75	-	2.00	2.00	1.00

Concerning the running time, it turns out that the TLG algorithm achieves in both cases very good results and often it is very close to the best performances. Concerning the number of scan operations performed by the algorithms, it turns out that the TLG algorithm always obtains the best results.

References

[Bel58] BELLMAN, R. E.: On a routing problem. In: *Quarterly Applied Mathematics* (1958), Nr. 16, S. 87–90

[Ber93] BERTSEKAS, D. P.: A simple and fast label correcting algorithm for shortest paths. In: *Networks* 23 (1993), S. 703–709

[CG99] CHERKASSKY, B. V. ; GOLDBERG, A. V.: Negative-cycle detection algorithms. In: *Mathematical Programming* 85 (1999), Nr. series A, S. 277–311

[CGR96] CHERKASSKY, B. V. ; GOLDBERG, A. V. ; RADZIK, T.: Shortest paths algorithms: theory and experimental evaluation. In: *Mathematical Programming* 73 (1996), Nr. series A, S. 129–174

[CLR90] CORMEN, T. H. ; LEISERSON, C. E. ; RIVEST, R. L.: Introduction to Algorithms. In: *MIT Press, Cambridge, MA* (1990)

[Dia69] DIAL, R. B.: Algorithm 360: Shortest Path Forest with Topological Ordering. In: *Comm. ACM* 12 (1969), S. 632–633

[Dij59] DIJKSTRA, E. W.: A note on two problems in connexion with graphs. In: *Numerische Matematik* 1 (1959), S. 269–271

[For56] FORD, L. R.: Network flow theory. In: *Rand Corporation Report* (1956), S. P–293

[FT87] FREDMAN, M. L. ; TARJAN, R. E.: Fibonacci heaps and Their Uses In Improved Network Optimization Algorithms. In: *J. Assoc. Comput. Mach.* 34 (1987), S. 596–615

[GGK84] GLOVER, F. ; GLOVER, R. ; KLINGMAN, D.: Computational Study of an Improved Shortest Path Algorithm. In: *Networks* 14 (1984), S. 25–37

[GR93] GOLDBERG, A. V. ; RADZIK, T.: A Heuristic Improvement of the Bellman-Ford Algorithm. In: *Applied Math. Let.* 6 (1993), S. 3–6

[MCN91] MONDOU, J.-F. ; CRAINIC, T. G. ; NGUYEN, S.: Shortest path algorithms: a computational study with the C programming language. In: *Computers and Operations Research* 18 (1991), S. 767–786

[Moo59] MOORE, E. F.: The shortest path through a maze. In: *In Proceedings of the International Symposium on the Theory of Switching, Harvard University Press* (1959), S. 285–292

[Pal84] PALLOTTINO, S.: Shortest-path methods: complexity, interrelations and new propositions. In: *Networks* 14 (1984), S. 257–267

[Pap74] PAPE, U.: Implementation and efficiency of Moore algorithms for the shortest root problem. In: *Mathematical Programming* 7 (1974), S. 212–222

Minimum Weighted Feedback Vertex Set on Diamonds

F.Carrabs [a 1] , R.Cerulli [a 2] , M.Gentili [a 3] , G.Parlato [b 4]

[a] *Dipartimento di Matematica ed Informatica Università di Salerno, Via Ponte Don Melillo, 84084, Fisciano (SA), Italy*

[b] *Dipartimento di Informatica ed Applicazioni Università di Salerno, Via S. Allende, 84081, Baronissi (SA), Italy*

Abstract

Given a vertex weighted graph G, a minimum Weighted Feedback Vertex Set (MWFVS) is a subset $F \subseteq V$ of vertices of minimum weight such that each cycle in G contains at least one vertex in F. The MWFVS on general graph is known to be NP-hard. In this paper we introduce a new class of graphs, namely the *diamond* graphs, and give a linear time algorithm to solve MWFVS on it. We will discuss, moreover, how this result could be used to effectively improve the approximated solution of any known heuristic to solve MWFVS on a general graph.

Key words: Feedback Vertex Set, Dynamic Programming, Diamond Graphs

1 Introduction

Given a vertex weighted graph G, a minimum Weighted Feedback Vertex Set (MWFVS) is a subset $F \subseteq V$ of vertices of minimum weight such that each cycle in G contains at least one vertex in F. The MWFVS on general graph is known to be NP-hard. However, a large literature shows that it becomes polynomial when addressed on specific classes of graph: interval graph [1], cocomparability graph [2], AT-free graph, among others. In this paper we introduce a new class of graphs, namely the *diamond* graphs, and give a linear time algorithm to solve MWFVS on it. We will discuss, moreover, how this result could be used to effectively improve an approximated solution on a general graph, that is the object of our further research. Section 2 introduces the needed notation and the diamond graph class. Section 3 contains the description of our linear algorithm based on dynamic programming to optimally solve MWFVS on diamonds. Further research is discussed in Section 4.

[1] E-mail: `carrabs@dia.unisa.it`.
[2] E-mail: `raffaele@unisa.it`.
[3] E-mail: `mgentili@unisa.it`.
[4] E-mail: `parlato@dia.unisa.it`.

2 The class of diamond graphs

In this section we give some basic concepts used in this paper. Let $G = (V, E, w)$ be a weighted graph, where V is the set of vertices, $E \subseteq V \times V$ is the set of edges and $w : V \to \Re^+$ is a weight function which associates a positive real number to each vertex of G. Given a weighted graph $G = (V, E, w)$ and a subset X of V, we define $G - X = (V \setminus X, ((V \setminus X) \times (V \setminus X)) \cap E, w')$, where w' is the restriction of w on the domain $V \setminus X$.

A *tree* is an acyclic, connected and undirected graph. Let T be a tree rooted in r. Given a vertex u in T, we denote as C_u the set of children of u in T. The *height* of a vertex u in T, denoted by $h(u)$, is recursively defined as follows. If u is a leaf then $h(u) = 0$, otherwise $h(u) = \max_{v \in C_u}\{h(v)\} + 1$. We define the height of a tree T as the height of its root (i.e $h(T) = h(r)$). Given a vertex $u \in T$, the *subtree* of T rooted in u is the subgraph of T induced by the set of vertices constituted by u and its descendents in T. In the following we denote by T_u the subtree of T rooted in u.

Now we introduce a new class of graphs, namely the *Diamond* graphs.

Definition 1 *A* ***diamond*** $D = (V, E, w)$ *with apices* r *and* z *(with* $r, z \in V$*), is a weighted graph where (i) each* $v \in V$ *is included in at least a simple path between* r *and* z *and (ii)* $D - \{z\}$ *is a tree.*

We call the two vertices r and z of D the *upper* and *lower* apex of D, respectively. Given a diamond D with apices r and z, we refer to the tree $D - \{z\}$ rooted in r as T. Let us denote as D_u the subdiamond of D, with apices u and z, induced by vertices of T_u and z. The *height* of diamond D_u is given by the height of T_u (see Figure 1(a)).

Given a diamond D with apices r and z, we can see that it is formed by the subdiamonds D_{r_i}, with $r_i \in C_r$, having the common lower apex z and upper apex r_i.

3 Our resolution algorithm

In this section, we propose a linear algorithm, based on dynamic programming, to solve the MWFVS problem on diamonds. In the rest of this section we consider a diamond D having r and z as its upper and lower apices. Note that by definition the set $F = \{z\}$ is an FVS of D. Hence, an optimum solution F^* of MWFVS on D is such that either $F^* = \{z\}$ or, $z \notin F^*$ and $W(F^*) \leq w(z)$. Hence, the MWFVS problem on a diamond can be solved first finding a minimum FVS $\hat{F}$ that does not contain vertex z and then comparing $W(\hat{F})$ with $w(z)$. Let us notice that in $D - \hat{F}$ either there exists a simple path between r and z or it does not exist (see Figure 1(b)). We prove the above observation by the following proposition.

Proposition 1 *Given a diamond* D *with apices* r *and* z*, if* $F \subseteq V$ *is a (minimum) FVS of* D*, then there exists at most one path between* r *and* z *in* $D - F$*.*

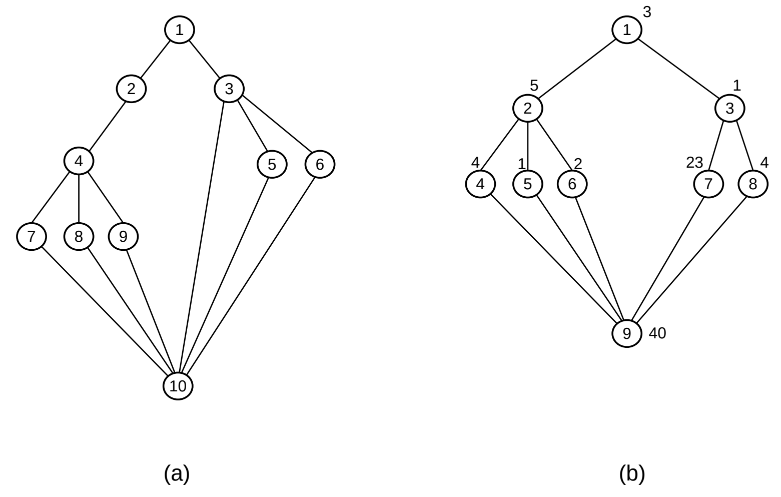


Figure 1. *(a) A diamond with apices* $r = 1$ *and* $z = 10$*. The height of this diamond is* 3 *and the associated tree* T *is* $D - \{z\}$*. Note that, the subgraph* D_2 *(with* $V_2 = \{2, 4, 7, 8, 9, 10\}$*) is a subdiamond of height 2 with* $r = 2$ *and* $z = 10$*.(b) A diamond with apices* $r = 1$ *and* $z = 9$*. A minimum FVS (that does not contain vertex* z*) is* $\hat{F} = \{3, 5, 6\}$ *and since* $W(\hat{F}) = 4 < w(z) = 40$ *we have that* $F^* = \hat{F}$*. Note that in* $D - \hat{F}$ *there is a simple path between* r *and* z*. Setting* $w(2) = 2$*, the set* $\hat{F} = \{2, 3\}$ *with* $W(\hat{F}) = 3$ *is the optimum FVS. In this case* $D - \hat{F}$ *does not contain a path between* r *and* z

Let us define now two new problems on a diamond D, related to MWFVS that will be useful to solve MWFVS on D.

Path Problem: find a subset $F^+ \subseteq V \setminus \{z\}$ of minimum weight such that (i) $D - F$ does not contain cycles, and, (ii) there exist exactly a path in $D - F$ between r and z.

NoPath Problem: find a subset $F^- \subseteq V \setminus \{z\}$ of minimum weight such that (i) $D - F$ does not contain cycles, and, (ii) there does not exist a path in $D - F$ between r and z.

From the above observations the minimum $\hat{F}$ on D is either F^+ or F^-. Therefore, the optimum solution F^* on D is such that:

$$W(F^*) = \min\{w(z), W(F^+), W(F^-)\}$$

and, then we have either $F^* = \{z\}$ or $F^* = F^+$ or $F^* = F^-$.

3.1 Optimal Substructure and Recursion Rules

In this section, we conjunctly characterize the structure of an optimal solution for both *Path* and *NoPath* problems. We recall that a problem has an optimal substructure if an optimal solution to the problem contains within it optimal solutions to subproblems. Both *Path* and *NoPath* problems have an optimal substructure property. We pick as our subproblems the problems of determining the value of solution to *Path* and *NoPath* problems on the subdiamonds D_u with $u \in V \setminus \{z\}$. We will denote by F_u^+ and F_u^- the optimal solutions of the *Path* and *NoPath* problem, respectively, on D_u. Now, we prove that the optimal solution F_u^+ (F_u^-) contains, for each $u_i \in C_u$, either $F_{u_i}^+$ or $F_{u_i}^-$. .

Proposition 2 *Given the optimum solution F_u^+ on D_u, then each set $F_{u_i} = F_u^+ \cap V_{u_i}$ is an optimal solution to either Path problem or NoPath problem on D_{u_i}.*

Proposition 3 *Given the optimum solution F_u^- on D_u, then each set $F_{u_i} = F_u^- \cap V_{u_i}$ is an optimal solution to either Path problem or NoPath problem on D_{u_i}.*

Now, we describe the recursion rules to obtain the values of the optimum sets F_u^+ and F_u^-. We define the $W(F_u^+)$ and $W(F_u^-)$ recursively as follows.

Optimum Solution to Path Problem: $W(F_u^+)$

We distinguish two cases according to the height $h(u)$:

- $h(u) = 0$
  Trivially $W(F_u^+) = 0$.

- $h(u) > 0$
  If $(u, z) \in E_u$, then since $u \notin F_u^+$, $G_u - F_u^+$ has surely a path between u and z composed by the edge (u, z). Thus, to avoid cycles in $G_u - F_u^+$, the set F_u^+ is obtained by the union of F_x^-, for each $x \in C_u$ and $W(F_u^+) = \sum_{x \in C_u} W(F_x^-)$. Instead, when $(u, z) \notin E_u$, in order to have a path from u to z in $G_u - F_u^+$, the optimal set is obtained by the minimum weight union of exactly one set F_x^+, for some $x \in C_u$, and by $\bigcup_{y \in C_u - \{x\}} F_y^-$. Therefore, we have the following:
$$W(F_u^+) = \min_{x \in C_u} \{W(F_x^+) + \sum_{y \in C_u - \{x\}} W(F_y^-)\}$$

By applying a similar reasoning we can derive the following recursion rules for the *NoPath* problem.

Optimum Solution to NoPath Problem: $W(F_u^-)$

- $h(u) = 0$
  $F_u^- = \{u\}$ and $W(F_u^-) = w(u)$.
- $h(u) > 0$

$$W(F_u^-) = \min \left\{ w(u) + \sum_{x \in C_u} \min \left\{ W(F_x^-), W(F_x^+) \right\}, \sum_{x \in C_u} W(F_x^-) \right\}$$

4 Conclusion and Further Research

We studied the Weighted Feedback Vertex Set on a special class of graph: the diamonds graph. We showed a linear time algorithm to solve the problem on this graph class. Further research is focused on the study of the larger class of multidiamond graphs (diamonds with multi-upper and lower apices). In addition, our purpose is to use the linearity of the MWFVS on this class of graph to try to improve an approximated solution of MWFVS on general graph.

References

[1] C.L.Lu, C.Y.Tang, *A linear-time algorithm for the weighted feedback vertex set problem on interval graphs.* Information Processing Letters 61 (1997) 107-111.

[2] M.S. M.S. Chang, Y.D. Liang: *Minimum feedback vertex sets in cocomparability graphs and convex bipartite graphs*, em Acta Informatica Vol.34 (1997), 337-346 Thomas Erlebach, Stamatis Stematis Stefanakos *Wavelength Conversion in Network of Bounded Treewidth* Tik Report Nr. 132, April 2002

Linear Time Algorithms to the Minimum All-Ones Problem for Unicyclic and Bicyclic Graphs

William Y.C. Chen, Xueliang Li [1], Chao Wang, Xiaoyan Zhang

Center for Combinatorics and LPMC, Nankai University, Tianjin 300071, P.R. China

Abstract

In this paper [2], we give graph-theoretic algorithms of linear time to the Minimum All-Ones Problem for unicyclic and bicyclic graphs. These algorithms are based on a graph-theoretic algorithm of linear time to the Minimum All-Ones Problem with Restrictions for trees.

Keywords: (Minimum) All-Ones Problem; graph-theoretic algorithm; linear time

AMS subject classification(2000): 05C85, 05C70, 90C27, 68Q25, 68R10

1 Introduction

The term *All-Ones Problem* was introduced by Sutner, see [10]. It has applications in linear cellular automata, see [11] and the references therein. The problem is cited as follows: suppose each of the square of an $n \times n$ chessboard is equipped with an indicator light and a button. If the button of a square is pressed, the light of that square will change from off to on and vice versa; the same happens to the lights of all the edge-adjacent squares. Initially all lights are off. Now, consider the following questions: is it possible to press a sequence of buttons in such a way that in the end all lights are on ? This is referred as the *All-Ones Problem*. If there is such a solution, how to find a such way ? And finally, how to find such a way that presses as few buttons as possible ? This is referred as the *Minimum All-Ones Problem*. All the above questions can be asked for arbitrary graphs. Here and in what follows, we consider connected simple undirected graphs only. One can deal with disconnected graphs component by component. For all terminology and notations on graphs, we refer to [7]. An equivalent version of the All-Ones Problem was proposed by Peled in [8], where it was called the *Lamp Lighting Problem*. The rule of the All-Ones Problem is called σ^+-rule on graphs, which means that a button lights not only its neighbors but also its own light. If a button lights only its neighbors but not its own light, this rule on graphs is called σ-rule.

In graph-theoretic terminology, a solution to the All-Ones Problem with σ^+-rule can be stated as follows: given a graph $G = (V, E)$, where V and E denotes the vertex-set and the

[1] E-mail: `x.li@eyou.com`.

[2] Supported by NSFC.

edge-set of G, respectively. A subset X of V is a solution if and only if for every vertex v of G the number of vertices in X adjacent to or equal to v is odd. Such a subset X is called an *odd parity cover* in [11]. So, the All-Ones Problem can be formulated as follows: given a graph $G = (V, E)$, does a subset X of V exist such that for all vertex $v \in V - X$, the number of vertices in X adjacent to v is odd, while for all vertex $v \in X$, the number of vertices in X adjacent to v is even ? If there exists a solution, how to find a one with minimum cardinality ?

There have been many publications on the All-Ones Problem, see Sutner [12,13], Barua and Ramakrishnan [1] and Dodis and Winkler [3]. Using linear algebra, Sutner [11] proved that it is always possible to light every lamp in any graphs by σ^+-rule. Lossers [6] gave another beautiful proof also by using linear algebra. A graph-theoretic proof was given by Erikisson et al [4]. So, the existence of solutions of the All-Ones Problem for general graphs was solved already. Sutner [10] proposed the question whether there is a graph-theoretic method to find a solution to the All-Ones Problem for trees. Galvin [5] solved this question by giving a graph-theoretic algorithm of linear time. In [9], Sutner proved that the Minimum All-Ones Problem is NP-complete for arbitrary graphs. So, it becomes an interesting problem to find graph-theoretic algorithms of polynomial time to the Minimum All-Ones Problem for some special classes of graphs. In [2] we gave a linear time algorithm for trees, which is based on the idea of Galvin algorithm [5] to the All-Ones Problem for trees. In his algorithm, the nodes of a rooted tree, drawn like a family tree with the root at the top, will be divided into three classes: outcasts, oddballs and rebels. The classification is defined inductively, from the bottom up, as follows:

• All of the childless nodes or leaves are rebels.

• A node, other than a leaf, is called an *rebel* if it has no oddball children and an even number of its children are rebels.

• A node is called an *oddball* if it has no oddball children and an odd number of its children are rebels.

• A node is called an *outcast* if at least one of its children is an oddball.

We sometimes simply call a node *r-type, b-type or o-type* if it belongs to the rebel class, the oddball class or the outcast class.

These notations and terminology will be used in the sequel.

2 Algorithm to the Minimum All-Ones Problem with Restrictions for Trees

In order to solve the Minimum All-Ones Problems for unicyclic and bicyclic graphs, we need to introduce and solve the Minimum All-Ones Problem with Restrictions for trees, which is an interesting problem on its own. First we need to solve the following problem.

For a matrix $M_{2\times n} = (m_{ij})_{2\times n}$, $i \in \{0, 1\}$, $j \in \{1, 2, \cdots, n\}$, $m_{ij} \in Z^+ \cup \{\infty\}$, the *Minimum*

Odd Sum Problem with Restrictions is defined as

$$\min \sum_{j=1}^{n} m_{0j}x_{0j} + m_{1j}x_{1j}$$

$$\begin{cases} \sum_{j=1}^{n} x_{1j} = 1 \mod 2 \\ x_{0j} + x_{1j} = 1, j = 1, 2, \cdots, n \\ x_{ij} \in \{0, 1\}, i \in \{0, 1\} \end{cases}$$

Here we suppose that "∞" is bigger than any $k \in Z^+$, and $\forall k \in Z^+$, $k + \infty = \infty$.

We omit the description and the proof of the algorithm for the Minimum Odd Sum Problem with Restrictions. The time complexity of the algorithm is linear.

Replacing $\sum_{j=1}^{n} x_{1j} = 1 \mod 2$ in the Minimum Odd Sum Problem with Restrictions by $\sum_{j=1}^{n} x_{1j} = 0 \mod 2$, we then get a new problem, called the *Minimum Even Sum Problem with Restrictions.* It can be solved in the same way as above. The details are omitted.

For convenience, we say that the truth value of a node u in G, denoted by $tv(u)$, is 1, if it belongs to the solution to the All-Ones Problem for G; and 0, otherwise.

The so-called "the Minimum All-Ones Problem with Restrictions for trees" is to find a solution to the Minimum All-Ones Problem for trees under the condition that the truth values of some nodes have been assigned. For this problem, our algorithm uses induction on the number of layers of a tree and the algorithms for the Minimum Odd or Even Sum with Restrictions as subprocess. The details are omitted.

3 Algorithm for Unicyclic Graphs

First, we recall that a unicyclic graph is a connected graph with a unique cycle. So, we can regard a unicyclic graph as a cycle attached with each node a rooted tree, called a *suspended tree.* Note that the depth of a suspended tree can be 0. For simplicity, we say that a node t in the cycle has the same type as the type of the root t of the suspended tree. Based on the algorithm with restrictions for trees in Section 2, we give a graph-theoretic algorithm of linear time to the Minimum All-Ones Problem for unicyclic graphs.

Algorithm for Unicyclic Graphs

Case 1. If none of the nodes on the cycle with length q is an outcast then we use the following way to get all the possible truth values of all nodes on the cycle.

(1) Fix an order on the cycle. Assume that the truth value of the 1st node is x, the truth value of the 2nd node is y. x and y will be completely determined in the end.

(2) Suppose that the truth value of the i-th node is a_i, a function of x and y. Next, determine

the truth value of the $(i+1)$-th node in two cases: If the i-th node is a rebel, then the truth value of the $(i+1)$-th node is $a_{i+1} = (1 - a_i - a_{i-1}) \bmod 2$; else, the truth value is $a_{i+1} = a_{i-1}$. Repeat this step until $i = q$.

(3) After we get the truth value of a_q for the q-th node, the following equalities hold.

$$\begin{cases} a_{q-1} = x & \text{if the } q\text{-th node is an oddball} \\ a_{q-1} + a_q + x = 1 \bmod 2 & \text{if the } q\text{-th node is a rebel} \end{cases}$$

$$\begin{cases} a_q = y & \text{if the 1-st node is an oddball} \\ a_q + x + y = 1 \bmod 2 & \text{if the 1-st node is a rebel} \end{cases}$$

By solving these equalities, we get at most 4 possible values of x and y, which determine all possible truth values of each node on the cycle. Suppose that all the possible $k (1 \leq k \leq 4)$ group of truth values of the nodes on the cycle are z_{j1}, z_{j2}, ..., z_{jq}, $(1 \leq j \leq k)$. According to each group of z_{j1}, z_{j2}, ..., z_{jq}, the unicyclic graph G has a solution to the All-Ones Problem. Conversely, any solution to the All-Ones Problem for the unicyclic graph G has a restriction on the nodes on the cycle, which must coincide with one of the t groups z_{j1}, z_{j2}, ..., z_{jq}, $(1 \leq j \leq k)$.

Subcase 1.1 v_i is a rebel. If $z_{ji} = 1$. Let $C_j(T_i)$ be the minimum solution to the All-Ones Problem for T_i. If $z_{ji} = 0$. Let $C_j(T_i)$ be the minimum solution to the Quasi-All-Ones Problem for T_i.

Subcase 1.2 v_i is an oddball. If $z_{ji} = 1$, then let $C_j(T_i)$ be the minimum solution to the All-Ones Problem for T_i with the restriction that the truth value of the root is 1. If $z_{ji} = 0$, then let $C_j(T_i)$ be the minimum solution to the All-Ones Problem for T_i with the restriction that the truth value of the root is 0.

After getting all the $C_j(T_i)$, $1 \leq i \leq q$, we can see that

$$C_j(G) = \bigcup_{1 \leq i \leq q} C_j(T_i), \quad 1 \leq j \leq k$$

are all the possible minimum solutions to the All-Ones Problem for the unicyclic graph G. So the minimum one in the k solutions is the minimum solution to the All-Ones Problem for G.

Case 2. There is at least one of the nodes on the cycle is an outcast. Suppose this outcast node is u. We fix an order to the nodes on the cycle. Suppose the node before u on the cycle is v, the node after u on the cycle is w. Then we cut the edges between u and v and u and w. The unicyclic graph G will be changed into two trees. One has the root u, denoted by T_u, the other is the remaining part of G excluding T_u, denoted by T'_u. We can easily verify that the minimum solution to the All-Ones Problem for G, denoted by $C(G)$, must be one of the following four possible solutions:

$$C(T'_u|tv(v) = i, tv(w) = j) \bigcup C_{[(i+j+1) \bmod 2]}(T_u), \quad i, j \in \{0, 1\},$$

where $C_{[1]}(T_u)$ means the minimum solution to the All-Ones Problem for the suspended tree T_u with the root u, $C_{[0]}(T_u)$ means the minimum solution to the Quasi-All-Ones Problem for T_u.

Then the minimum one of the four possible solutions will be the minimum solution to the All-Ones Problem for unicyclic graphs.

Summing up the above, we get that the algorithm outputs a solution to the Minimum All-Ones Problem for unicyclic graphs, and the time complexity is linear.

For bicyclic graphs, the analysis of the Minimum All-Ones Problem would be similar but more detailed and complicated. As an abstract, we have to omit its details. We can use the same technique for tricyclic graphs, quadrucyclic graphs, etc. But, unfortunately, we cannot employ it efficiently for graphs with more and more cycles. We have point out that in [9], Sutner proved that the Minimum All-Ones Problem is NP-complete for general graphs.

References

[1] R. Barua and S. Ramakrishnan, σ-game, σ^+-game and two-dimensional additive cellular automata, Theoret. Comput. Sci., 154(1996), 349-366.

[2] William Y.C. Chen, X. Li, C. Wang and X. Zhang, The minimum all-ones problem for trees, SIAM J. Computing 33(2)(2004).

[3] Y. Dodis and P. Winkler, Universal configurations in light-flipping games, Proceedings of 12-th Annuaql ACM/SIAM Symposium on Discrete Algorithms (SODA), Jannuary 2001, 926-927.

[4] H. Eriksson, K. Eriksson and J. Sjöstrand, Note on the lamp lighting problem, Advances in Applied Mathematics, 27(2001), 357-366.

[5] Fred Galvin, Solution to problem 88-8, Math. Intelligencer, Vol.11, No.2(1989), 31-32.

[6] O.P. Lossers, Solution to problem 10197, Amer. Math. Monthly, Vol.100, No.8(1993), 806-807.

[7] Christos H. Papadimitriou and K. Steiglitz, Combinatorial Optimizations: Algorithms and Complexity, Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1982.

[8] Uri Peled, Problem 10197, The American Mathematical Monthly, Vol.99, No.2(1992), 162.

[9] K. Sutner, Additive automata on graphs, Complex Systems, Vol.2, No.1(1988), 1-28.

[10] K. Sutner, Problem 88-8, Math. Intelligencer, Vol.10, No.3(1988).

[11] K. Sutner, Linear cellular automata and the Garden-of-Eden, Math. Intelligencer, 11(2)(1989), 49-53.

[12] K. Sutner, The σ-game and cellular automata, Amer. Math. Monthly, 97(1990), 24-34.

[13] K. Sutner, σ-automata and Chebyshev-polynomials, Theoret. Comput. Sci., 230(2000), 49-73.

The demand-dependent optimization of regular train timetables

Alessandro Chierici [1] Roberto Cordone [2]

DTI - Università degli Studi di Milano, Via Bramante 65, 26013 - Crema, Italy

Roberto Maja [3]

INDACO, Politecnico di Milano, Via Durando 10, 20158 - Milano, Italy

Abstract

Regular timetables, in which the trains arrive and depart at constant intervals, have been adopted in various European countries, because of the simpler and fairer service they allow. The design of such a timetable has recently received a certain attention in the literature. This paper extends the commonly adopted model to take into account the reciprocal influence between the quality of a timetable and the transport demand captured by the railway with respect to alternative means of transport. The resulting mixed-integer non linear model remains non convex even after relaxing the integrality constraints. We solve it by a branch-and-bound algorithm based on *Outer Approximation* and a heuristic algorithm exploiting the decomposition and reciprocal update of two submodels. Preliminary computational results concern a regional network in North-western Italy.

Key words: Regular timetabling, Global optimization, Outer approximation

1 Introduction

In a *regular timetable*, such as the ones adopted by the Dutch and Swiss companies [1], the trains servicing a line follow each other at fixed time intervals. The passengers enjoy an easily memorizable timetable and a good level of service over the whole day and for every origin-destination (O/D) pair. The railway company enjoys a simpler way to manage its resources. To tune the balance between transport demand and offer along the day, some special trains can be superposed to the regular schedule during the peak hours and some regular ones can be dropped during low-traffic hours.

[1] E-mail: `alexchr@tiscali.it`.
[2] E-mail: `cordone@dti.unimi.it`, URL: `http://www.dti.unimi.it/ cordone`. Corresponding author.
[3] E-mail: `roberto.maja@polimi.it`.

The design of regular timetables has been modelled as a *Periodic Event Scheduling Problem* (*PESP*) [2]. Nachtigall proposed to minimize the passenger waiting time and to admit lines with different periods [3], Odijk proposed a branch-and-cut algorithm [4]. Recent developments take into account a formulation stated in terms of cycles in an auxiliary graph [5]. This model has a large number of constraints and integer variables, but only a linear subset are relevant. Moreover, strong inequalities derive from suitable cycle classes.

While all previous studies assume a given transport demand, this paper accounts for the strong feed-back influence exerted by the timetable on the demand itself. The actual demand attracted by the train is estimated by a *discrete-choice models* [6], whose integration into the regular timetable model provides an interesting mixed-integer non linear problem. Its objective is to maximize the total demand captured by the train. Since in Italy several local networks exhibit a critical relation between timetable and demand, including such a feed-back mechanism in a model is of paramount practical interest. Since the model is non-convex, even if the integrality constraints are relaxed, we solve it through a branch-and-bound algorithm, which allows to evaluate upper and lower bounds on the optimum. We also introduce a heuristic based on the decomposition and reciprocal update of the timetable design and demand estimate submodels. Experience proves that the method quickly converges to a fairly good solution. The preliminary computational results refer to a regional network in Northwestern Italy.

2 The model

A *line* is a sequence of trains following the same track and stopping at the same stations every P minutes. The trains moving in the opposite direction form a different line. The *constraint graph* $G(N, A)$ is a directed graph whose nodes model the entrance and the exit of each line in each stop. The arcs correspond to time constraints between the events modelled by the end nodes and are associated to time variables x_a. The *travel arcs* go from the exit of a line from a station to its entrance in the following one and are associated to travel times. The *backward arcs* go from the entrance of a line into a station to the exit of the opposite line; they are associated to the waiting time before getting back. The *stopping arcs* go from the entrance to the exit of a line in a station, the *transfer arcs* from the entrance of a line to the exit of another one; they are associated, respectively, to stopping times and line transfer times.

The x_a variables identify a timetable, except for an irrelevant additive uniform shift. To define a regular timetable, they must satisfy

$$\sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a = P q_C \qquad C \text{ cycle of } G \tag{1}$$

$$l_a \le x_a \le u_a \qquad a \in A \tag{2}$$

$$h_C \le q_C \le k_C \qquad a \in A \tag{3}$$

$$x_a \in \mathbb{R} \qquad a \in A \tag{4}$$

$$q_C \in \mathbb{Z} \qquad C \text{ cycle of } G \tag{5}$$

where the integer variables q_C are associated to the cycles of the undirected graph underlying G. If C^+ and C^- denote the arcs consistent and opposed to an arbitrary orientation of cycle C, Equations (1) express regularity: any passenger following a close circuit should return to his origin after an integer number of periods. Equations (2) bound the time variables, modelling technical limitations on the travel times and quality guarantees on the length of the stops and of the transfers from train to train. They also imply Equations (3), which limit the integer variables q_C to 2 or 3 feasible values. Equations (1) and variables q_C are exponentially many, but the formulation is practical, as only a linear number of them are necessary [5]. Further constraints model practical limitations, such as the safety delay between two trains travelling on the same track in the same direction or in opposite ones (for single-track lines).

We integrate the previous constraints with a *modal choice* model, which relates the features of a transport mode (such as the length of the trips, deriving from the timetable) to the fraction of potential travellers captured. The *Logit model* [6] assumes the travellers to be rational decisors, maximizing their own utility, which is not completely known. We consider three alternative modes: the bus, the car, and the train ($j = b, c, t$) and characterize them for each O/D pair by such attributes as travel time, monetary cost, walking time, average waiting time and comfort. The utility associated to each mode is a random variable. Its sistematic component $V^{(j)}$ linearly combines the attributes with coefficients estimated by a *revealed preference* (*RP*) analysis [7]. Its random residual ϵ is assumed to be independently and identically distributed according to the Weibull-Gumble distribution. The Logit model estimates the probability with which a traveller prefers each mode: if Λ_{od} denotes the daily number of individuals travelling from o to d, the number of train passengers is

$$\lambda_{od} = \Lambda_{od} \frac{\exp\left[V_{od}^{(t)}\left(T_{od}\left(x\right)\right)\right]}{\exp\left[V_{od}^{(b)}\right] + \exp\left[V_{od}^{(c)}\right] + \exp\left[V_{od}^{(t)}\left(T_{od}\left(x\right)\right)\right]} \tag{6}$$

where $V^{(t)}$ depends on the total travel time T_{od} from station o to station d, which is the sum of all travel and waiting times along the path P_{od}

$$T_{od} = \sum_{a \in P_{od}} x_a \tag{7}$$

Path P_{od} is assigned *a priori*, since most of the time the shortest path is the same for all feasible timetables. Our model combines Equations (1-5), (6) and (7) with the objective function

$$\max z = \sum_{od} \lambda_{od} \tag{8}$$

The resulting formulation is a *Mixed Integer Non Linear Problem* (*MINLP*) and it is non convex, both because of the integrality constraints and because of the non linear equality constraints (6).

3 The algorithm

The proposed algorithm is a simple branch-and-bound approach based on the formulation previously introduced. We replace Equations (6) by

$$\lambda_{od} \leq \Lambda_{od} \frac{\exp\left[V_{od}^{(t)}\left(T_{od}\left(x\right)\right)\right]}{\exp\left[V_{od}^{(b)}\right] + \exp\left[V_{od}^{(c)}\right] + \exp\left[V_{od}^{(t)}\left(T_{od}\left(x\right)\right)\right]} \tag{9}$$

which are equivalent, as each variable λ_{od} appears in a single constraint and in the objective function. Both T_{od} and $V_{od}^{(t)}$ vary in given intervals; λ_{od} is a decreasing, first convex then concave, function of T_{od}. To obtain an upper bound, we replace Equations (9) by their linear envelope between the extreme values of T_{od}. The resulting *MIP* is solved by a general-purpose solver (presently *CPLEX*, through the *AMPL* interface): the sum of the resulting $\tilde{\lambda}_{od}$ bounds from above the number of individuals which could prefer to move by train.

Moreover, the (x, q) variables yield a heuristic timetable. Equations (6) provide heuristic values λ_{od}^{H}, yielding a lower bound on the optimum. When T_{od} hits the upper or lower bound for all O/D pairs, the relaxed solution is feasible and optimal. When λ_{od} is convex for all values of T_{od} and for all O/D pairs, the relaxed feasible region is convex: the *Outer Approximation* or the *Generalized Benders' Decomposition* allow to update the linear envelope, gradually converging towards a global optimum. However, in most O/D pairs λ_{od} is concave, at least for large values of T_{od}. Therefore, the gap between the linear envelope and the actual feasible region is unremovable. Given one O/D pair in which T_{od} falls between its lower and upper bounds, the feasible interval can be split in two subproblems with linear envelopes stricter than the original one. The most promising O/D pair is the one with the largest gap between λ_{od}^{H} and $\tilde{\lambda}_{od}$.

An iterative approach provides another heuristic solution. Assumed a given number λ_{od}^{0} of passengers for each O/D, an auxiliary *MIP* determines the schedule which maximizes their total utility. Conversely, the optimal schedule updates the demand, reinitializing the problem. Though it is not formally guarantee, the heuristic experimentally converges.

4 Computational results

The experimental campaign employs a regional rail network in North-western Italy, around the station of Mortara (11 regional lines, 2 direct lines and 3 intercity lines). The network includes 72 stations and several cycles. The travel times derive from the official 2002-2003 timetable, the O/D matrix from the ISTAT 1991 census, updated by more recent studies [7]. Preliminary computational results show that the method provides useful information on a regional railway network and is not far from an exact solution. A 2.2 GHz Pentium III computer proves, in approximately one hour, that at most 4.03% of the individuals could choose the train. The corresponding heuristic timetable captures only 2.98% of the travellers, but the iterated heuristic finds in a few hours a solution with $z = 3.94\%$. Thus, the optimality gap is only $\varepsilon = 2.21\%$.

References

[1] J. Hooghiemstra, L. Kroon, M. Odijk, M. Salomon, P. Zwaneveld, Decision support systems support the search for win-win solutions in railway network design, Interfaces 29 (2) (1999) 15–32.

[2] P. Serafini, W. Ukovich, A mathematical model for periodic scheduling problems, SIAM Journal on Discrete Mathematics 2 (4) (1989) 550–581.

[3] K. Nachtigall, Periodic network optimization with different arc frequencies, Discrete Applied Mathematics 69 (1996) 1–17.

[4] M. Odijk, A constraint generation algorithm for the construction of periodic railway timetables, Transportation Science 30 (6) (1996) 455–464.

[5] L. Kroon, L. Peeters, A variable trip time model for cyclic railway timetabling, Transportation Science 37 (2) (2003) 198–212.

[6] M. E. Ben-Akiva, S. R. Lerman, Discrete Choice Analysis: Theory and Application to Travel Demand, MIT Press, Cambridge, Ma, 1985.

[7] R. Maja, Accordo di programma per la riqualificazione e il potenziamento della linea ferroviaria Milano-Mortara, Technical report, Politecnico di Milano (1999).

Multiway cut and integer flow problems in trees

Marie-Christine Costa[1] , Alain Billionnet

CEDRIC, CNAM, 292 rue St-Martin 75141 Paris cedex 03, France.

Abstract

Let $T = (V, E)$ be a n-vertices undirected tree and $X \subset V$ a set of terminal vertices. We consider the well-known multway cut problem to which we associate the problem of finding an integral multiway flow maximizing the flow routed between all pairs of terminals. These problems are special cases of multicut and integral multiflow problems which are known to be NP-hard in trees. We propose a generic procedure to explore and reduce a tree, which allows to devise an $O(n)$ procedure for the multiway cut and an $O(n^2)$ procedure for the integral multiway flow in trees. Efficient procedures are also proposed to solve the problems in directed trees.

Key words: multiway cut, multiway flow, multiterminal cut, tree.

1 Introduction

Consider a connected (simple) graph $G = (V, E)$ with a positive integral weight (or capacity) p_e on each edge e of E and a set $X \subset V$ of terminal vertices. The multiway (or multiterminal) cut problem is to find a minimum weight set of edges $E_1 \subset E$ whose removal separates each pair of terminals in X. The graph $G_1 = (V, E - E_1)$ contains $|X|$ components with no two vertices from X in the same component. Associate a commodity with each pair of terminals; there are $\frac{1}{2}|X|(|X| - 1)$ different commodities. The integer multiway flow problem consists in maximizing the sum over all commodities of the integral flow corresponding to a commodity, subject to capacity and flow conservation requirements. Each commodity has an integral flow through each edge.

Generalizations of these problems are the minimum multicut and maximum integral multicommodity flow problems where instead of a given set X of terminals we are given a list of K pairs of terminals. Both problems, can be formulated as integer linear programs whose the continuous relaxations are two dual linear programs ([4],[8]).

The multiway cut problem is known to be NP-hard and MAX SNP-hard, in directed graphs and in undirected graphs, [7], [6]. It is NP-hard in undirected planar graphs, too. These results holds for the multicut even in stars, and integral multiflow problems even in undirected trees [8]. However, both problems are solvable in polynomial time in directed trees [3].

[1] E-mail: `costa@cnam.fr`.

The minimum multiway cut problem has been extensively studied [1, 2, 5, 6, 7, 9], contrary to the maximum integral multiway flow problem. One of many applications of the multiway cut problem is relative to parallel query optimization in databases: Hasan and Motwani [10] gave an algorithm for the problem of choosing the partitioning attribute in a query tree so as to minimize the total cost. In fact, they solved efficiently the multiway cut problem in trees.

In this paper, we prove that their algorithm can be implemented in $O(n)$ (that improves the algorithms in $O(n^2)$ proposed in [2] and [9]) and we devise an $O(n^2)$ algorithm solving the integer multiway flow problem in trees.

In Section 2, we give a generic algorithm to explore and treat a tree with a complexity time bounded by $O(n)$ or $O(n^2)$. In Section 3 we show that Hasan and Motwani's algorithm follows the generic algorithm and is in $O(n)$. In Section 4 we solve the integer multiway flow problem. In Section 5 we solve efficiently the problems in directed and rooted trees. In Section 6 we compare multiway cut and integral multiway flow optimal values in trees before concluding.

We set that $p(A) = \sum\limits_{e \in A} p_e$, for all $A \subseteq E$.

For the rest of the paper we will consider trees in which there is a bijection between leaves and terminals, there is no path without junction and all the edge weights are positive. We will show that these restrictions are made without loss of generality.

2 A generic procedure in trees

In this section, we describe a general scheme for exploring and treating a tree. Let $T = (V, E)$ be a tree with $n = |V|$. Our algorithm starts by rooting T at an arbitrary vertex, denoted by x_0. We then number the other vertices and all the edges from 1 to $n-1$ following a breadth-first ordering; doing so, x_j is the lowest endpoint of the edge u_j.

If T is not a star, let a 1-star $S \subset E$ be a maximum sub-set of edges adjacent to a leaf and admitting a common endpoint, noted x_S; let u_S be the edge connecting S to the rest of the tree. We say that we contract an edge $u = (x_i, x_j)$, $j > i$, if the lowest endpoint of u, x_j, is collapsed into its father, x_i. Let $u_k = (x_j, x_k)$; after contracting (x_i, x_j), the new edge (x_i, x_k) represents the edge (x_j, x_k) and takes its name, u_k. The sketch is to treat the tree from the leaves up to the root. First we select a 1-star, S, from the lowest level of the rooted tree. Second, we treat this 1-star. Third, we reduce it either by contracting all its edges (x_S becomes a leaf) or by contracting some of its edges plus u_S (x_S disappears). We denote by m_S the number of contracted edges: m_S depends on the type of problem to solve. We get a new tree and we reiterate the process. Let S_i be the 1-star selected at step i and m_{S_i} be its cardinality. We prove that the procedure, called $generic - trees$, is in $O(n)$ if $m_{S_i} = |S_i|$ for all i, or in $O\left(n^2\right)$ if $m_{S_i} \leq |S_i|$ for all i.

3 Multiway cuts in trees

Consider a star with n edges, n terminals and a midpoint x: a minimum cut contains all the edges but one, the maximum weighted edge. Therefore, whereas the general minimum multicut problem is NP-hard in a star [8], the multiway cut problem is trivially solved in $O(n)$.

Then, we can show that the algorithm *DLC* proposed in [10] to solve the problem in trees follows the scheme given by procedure $generic - trees$, with $m_{S_i} = |S_i|$ for all i: it can be implemented in $O(n)$ (without sorting the edges).

4 Integer multiway flows in trees

First we solve the problem in stars. In stars, the general maximum integral multiflow problem, and so, the multiway flow problem too, can be solved by an algorithm in $O(n^{\frac{5}{2}})$ [8]. Here, we propose an algorithm in $O(n)$ to solve the multiway flow problem in stars; that is the best possible complexity time.

Lemma 1 *Let $T = (V, S)$ be a star and let w be the maximum capacited edge of S. The value of a maximum integer multiway flow ϕ^* in T is*

(i) $v(\phi^*) = \sum_{e \in S, e \neq w} p_e \quad \textit{if} \quad p_w \geq \sum_{e \in S, e \neq w} p_e$

(i.i) $v(\phi^*) = \left\lfloor \frac{\sum_{e \in S} p_e}{2} \right\rfloor$ *otherwise*

Proof. We explain how routing a flow ϕ whose the value is $v(\phi^*)$ and we prove that it is the best possible value.

Now we solve the integer multiway flow problem in trees. The algorithm is based on the previous result in stars and uses the scheme of procedure $generic - trees$. Nevertheless the reductions must be adapted and are much more complicated than the reductions used for the multiway cut. At each step, we route a sub-flow inside a 1-star S_i: this sub-flow must be bounded in order to keep a residual capacity inside S_i. This residual capacity is equal to $p_{u_{S_i}}$ or $p_{u_{S_i}} - 1$. Then, we contract m_{S_i} edges with $m_{S_i} \leq |S_i|$. To prove that the obtained flow is optimal, we need to study several cases, corresponding to different relationships between $p(S_i)$ and $p_{u_{S_i}}$. Finally, we prove that the complexity time of procedure $multiway-flow-in-trees$ is $O(n^2)$.

5 Multiway problems in directed trees

In a directed tree, we show that the multiway cut and integral flow problems can be reduced to a simple max flow-min cut problem and can be solved in $O(n^{\frac{5}{2}})$. In a rooted tree multicut and integer multiflow problems can be solved in $O(Kn)$ by using the algorithm proposed in [3]. We show that the multiway cut problem can be solved in $O(n)$ and the integral multiway flow problem can be solved in $O(Kh)$ in a rooted tree of height h.

6 Conclusion

The minimum multiway cut, minimum multicut and maximum integer multiflow problems are generally NP-hard. But, while both last problems are NP-hard in trees, the multiway cut and integer multiway flow problems are polynomially solvable. In this paper, we proposed a general scheme which allows to solve the minimum multiway cut problem in $O(n)$, and the maximum integer multiway flow problem in $O(n^2)$, when the graph is a tree. Note that there is often a gap between the multiway cut and integer flow optimal values. We have also shown that the two problems can be reduced to a simple max flow-min cut problem when the graph is a directed tree: to our knowledge, no algorithm have been proposed to solve the multicut and integral multiflow problems in directed trees although they are known to be polynomial [3]. We have also seen how to solve efficiently the integer multiway flow problem in a rooted tree. Finally, in a rooted tree, we used our generic algorithm to solve the multiway cut in $O(n)$, and one can imagine that the generic algorithm could be used for solving some other optimization problems on trees.

References

[1] D.Bertsimas, C-P. Teo and R. Vohra. 1999. Analysis of LP relaxations for Multiway and multicut problems. Wiley. 102-114.

[2] S. Chopra and M.R. Rao. 1991. On the multiway cut polyhedron. Networks 21. 51-89.

[3] M.-C. Costa, L. Létocart and F. Roupin. 2002. A greedy algorithm for multicut and integral multiflow in rooted trees. Operations Research Letters. Vol 31 (1),2003.

[4] M.-C. Costa, L. Létocart and F. Roupin. Minimal multicut and maximal integer multiflow: a survey. To appear in EJOR.

[5] W. H. Cunningham. 1991. The optimal multiterminal cut problem. DIMACS Series in Disc. Math. and Theor. Comput. Sci., 5. 105-120.

[6] E. Dalhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. 1994. The complexity of multiterminal cuts. SIAM J. Comput 23, 864-894.

[7] N. Garg, V.V. Vazirani and M. Yannakakis. 1994. Multiway cuts in directed and node weighted graphs. In proceedings ICALP. 487-498.

[8] N. Garg, V.V. Vazirani and M. Yannakakis. 1997. Primal-Dual approximation algorithms for integral flow and multicut in trees. Algorithmica 18, 3-20.

[9] P.L. Erdos and L.A. Székely. 1994. On weighted multiway cuts in trees. Mathematical Programming 65, 93-105.

[10] W. Hasan and R. Motwani. 1995. Coloring away communication in parallel query optimization. Proc. 21st VLDB Conference. Swizerland. 239-250.

New exponential neighbourhood for polynomially solvable TSPs

V.Deineko[1]

Warwick Business School, Coventry, CV4 7AL, UK

Abstract

Many of the well known solvable cases of the symmetric travelling salesman problem (STSP) have been characterized by describing special conditions for the underlying distance matrices. For all these special cases an optimal tour can either be given in advance and without regarding the precise numerical values of the data, or can efficiently be found in the set of so-called pyramidal tours. We introduce a new polynomially solvable case of the STSP where an optimal tour can be found in an exponential neighbourhood which is different from the set of pyramidal tours. Our new case is the first example of "multi-peak" optimization for polynomially solvable STSPs.

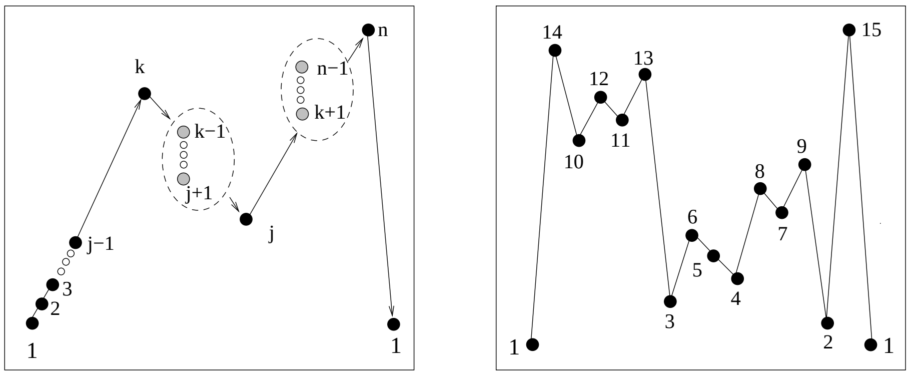
Key words: Traveling salesman problem, specially structured matrices, exponential neighbourhood, recognition algorithm

1 Introduction

The travelling salesman problem (TSP) is a well known problem of combinatorial optimization. In the symmetric TSP (STSP), given a symmetric $n \times n$ distance matrix $C = (c_{ij})$, one looks for a cyclic permutation τ of the set $\{1, 2, \ldots, n\}$ that minimizes the function $c(\tau) = \sum_{i=1}^{n} c_{i\tau(i)}$. Items in the permutation is usually referred as points or cities. A pair $[i, j]$ with $j = \tau(i)$ is referred as an *arc* of the tour τ.

Although the STSP is NP-hard, there are quite a few special cases when the problem can be solved in polynomial time [4, 2, 6]. Many of the well known solvable cases of the STSP have been characterized by describing special conditions for the underlying distance matrix. For all such special cases an optimal tour can either be identified in advance, or can be found in the set of so-called pyramidal tours. We introduce a new polynomially solvable case of the STSP where an optimal tour can be found in an exponential neighbourhood which is different from the set of pyramidal tours. Our new case is the first example of "multi-peak" optimization for polynomially solvable STSPs.

[1] E-mail: `V.Deineko@warwick.ac.uk`.



(a) (b)

Figure 1. Illustration to $\mathcal{N}$-permutations: (a) – definition of $\mathcal{N}$-structure; (b) – example of an $\mathcal{N}$-permutation.

2 TSP on Relaxed Kalmanson Matrices

New exponential neighbourhood. A symmetric $n \times n$ matrix $C = (c_{ij})$ will be called a *relaxed Kalmanson* [7] matrix (RK-matrix), if it satisfies the condition $c_{ik} + c_{jl} \geq c_{il} + c_{jk}$ for all indices $i < j < k < l$. It can be shown that these conditions can be checked in $O(n^2)$ time. Cyclic permutation π will be called $\mathcal{N}$-permutation if it does not contain pairs of arcs $[i, \pi(i)]$, $[j, \pi(j)]$ such that $i < \pi(i)$ and $j > \pi(i) > \pi(j) > i$, or $i < \pi(i)$ and $\pi(i) > j > i > \pi(j)$. Properties of $\mathcal{N}$-permutations are characterized in the next lemmas.

Lemma 1 *An optimal tour for the STSP with an* RK-*matrix can be found among $\mathcal{N}$-permutations.*

Lemma 2 *Every $\mathcal{N}$-permutation contains arc* $[1, n]$.

Lemma 3 *A structure of the path from* 1 *to* n *in an $\mathcal{N}$-permutation, to which we will refer as the $\mathcal{N}$-structure, can recursively be defined as follows: If there is no valley on the path from* 1 *to* n*, then this is the path* $\langle \mathbf{1}, 2, \ldots, n-1, \mathbf{n} \rangle$*. Otherwise let* j *be the minimal valley in the path from* 1 *to* n*. In this case the path has the structure*

$$\langle \mathbf{1}, 2, \ldots, j-1, k, \{j+1, j+2, \ldots, k-1\}, j, \{k+1, k+2, \ldots, n-2, n-1\}, \mathbf{n} \rangle$$

where k *is a peak and the two paths – from* j *to* k *through the set* $\{j+1, j+2, \ldots, k-2, k-1\}$*, and from* j *to* n *through the set* $\{k+1, k+2, \ldots, n-2, n-1\}$ *have the $\mathcal{N}$-structure.*

In the Lemma above the sets are meant to be empty if the first index in the set is bigger than the last one. Figure 1(a) illustrates the definition of the $\mathcal{N}$-structure. Figure 1(b) shows a permutation that satisfies the definition: path $\langle 1, \ldots, 15 \rangle$ has 2 as the minimal valley, so $j = 2$ and $k = 14$; for the path $\langle 2, \ldots, 14 \rangle$ the corresponding pair (j, k) is $(3, 9)$, and so on.

It can be seen from the definition of the $\mathcal{N}$-structure, that $\mathcal{N}$-permutations belong to the set of so-called twisted permutations ([1]). It means that the optimal TSP tour can be found in $O(n^7)$ time using the algorithm for finding an optimal twisted permutation ([3]). The special structure of $\mathcal{N}$-permutations allows us to find an optimal solution much faster, as shown in the theorem below.

Theorem 1 *The STSP with an* RK-*matrix as a distance matrix can be solved in $O(n^4)$ time.*

Proof. Let $L[p,q]$ be the length of the shortest path with the N-structure from index p to index q through the set of indices $\{p+1, p+2, ..., q-2, q-1\}$, $p < q$, and $V[j,p,q]$ be the length of the shortest path with N-structure from index j to index q through the set of indices $\{p, p+1, ..., q-2, q-1\}$, $j < p < q$. It follows from the definition of the $\mathcal{N}$-structure that the values L and V satisfy the following recursions:

$$L[p,q] = min \begin{cases} \sum_{t=p}^{q-1} c_{t,t+1} \\ min_{j<k}\Big\{ \sum_{t=p}^{j-2} c_{t,t+1} + c_{j-1,k} + L[j,k] + V[j,k+1,q] \Big\} \end{cases}$$

$$V[j,p,q] = \begin{cases} c_{jp}, & \text{if } p > q, \\ min \begin{cases} c_{jp} + L[p,q] \\ min_k \Big\{ c_{jk} + \sum_{t=p}^{k-1} c_{t,t+1} + V[p,k+1,q] \Big\} \end{cases} & \text{otherwise.} \end{cases}$$

If in the formulae above the upper limit in a sum is smaller than the lower limit, then the sum is zero. The length of the optimal tour can be calculated as $L[1,n] + c_{n1}$. Each of the values L can be calculated in $O(n^2)$ time, each of the values V can be calculated in liner time. It gives $O(n^4)$ overall complexity. The lemma is proved.

Recognizing permuted RK**-matrices.** Unfortunately the problem of recognizing permuted RK-matrices remains open. For the STSP with the Euclidean distance matrices, however, the problem can be solved in polynomial time.

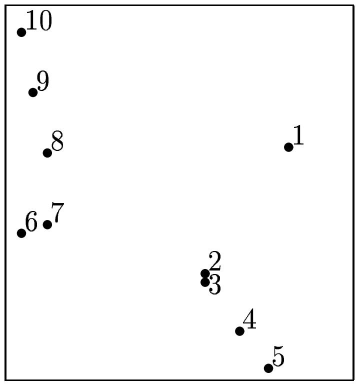
Theorem 2 *It can be decided in $O(n^4 log\ n)$ time whether an $n \times n$ matrix $C = (c_{ij})$ is a permuted Euclidean* RK-*matrix. If it is, a permutation σ such that $(c_{\sigma(i),\sigma(j)})$ is an* RK-*matrix is explicitly determined within this time bounds.*

Examples of Euclidean sets of points, for which the corresponding distance matrices are RK-matrices, are shown on Figure 2.

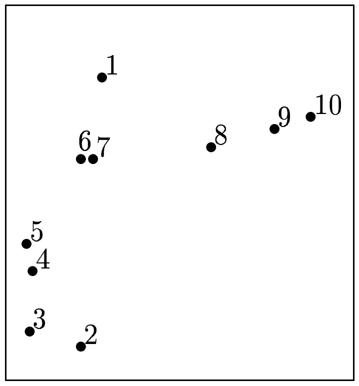
References

[1] F. Aurenhammer, On-line sorting of twisted sequences in linear time. *BIT*, **28**, 1988, 194–204.

	1	2	3	4	5	6	7	8	9	10
(a)	(96,81)	(67,37)	(67,34)	(79,17)	(89,4)	(3,51)	(12,54)	(12,79)	(7,100)	(3,121)
(b)	(28,114)	(21,13)	(1,18)	(2,41)	(0,51)	(21,83)	(25,83)	(70,88)	(94,95)	(107,99)



(a)



(b)

Figure 2. Euclidean RK-sequences of points with coordinates: (a) optimal tour is $\langle 1,5,4,3,2,7,6,8,9,10,1\rangle$; (b) optimal tour is $\langle 1,7,6,5,4,3,2,8,9,10,1\rangle$.

[2] R.E. Burkard, V.G. Deineko, R. van Dal, J.A.A. van der Veen, G.J. Woeginger. Well-solvable special cases of the TSP: A survey. *SIAM Review* **40**, 3, 1998, 496–546.

[3] V.G. Deineko, G.J. Woeginger. A study of exponential neigborhoods for the travelling salesman problem and for the quadratic assignment problem. *Mathematical Programming*, **87**, 2000, 519–542.

[4] P.C. Gilmore, E.L. Lawler and D.B. Shmoys. Well-solved special cases. Chapter 4 in [8], 87–143.

[5] G. Gutin and A.P. Punnen, *The travelling salesman problem and its variations*. Kluwer Academic Publishers, 2002.

[6] S.N. Kabadi, Polynomially solvable cases of the TSP, Chapter 11 in [5], 489–583.

[7] K. Kalmanson, Edgeconvex circuits and the traveling salesman problem, *Canadian Journal of Mathematics* **27**, 1975, 1000–1010.

[8] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys. *The Traveling Salesman Problem*. Wiley, Chichester, 1985.

Minimum Dominating Trail Set for Two-Terminal Series Parallel Graphs

Paolo Detti [a 1], Carlo Meloni [b 2], Marco Pranzo [c 3]

[a] *Dipartimento di Ingegneria dell'Informazione - Università di Siena, Italy.*

[b] *Dipartimento di Elettrotecnica ed Elettronica - Politecnico di Bari, Italy.*

[c] *Dipartimento di Informatica e Automazione - Università Roma Tre, Italy.*

Abstract

Given a graph G, the Minimum Dominating Trail Set ($MDTS$) problem consists in finding a minimum cardinality collection of pairwise edge-disjoint trails such that each edge of G has at least one endvertex on some trail. The $MDTS$ problem is NP–hard for general graphs. In this paper an algorithmic approach for the $MDTS$ problem on (two-terminal) series parallel graphs is presented.

Key words: graph algorithms, edge domination, series parallel graphs.

1 Introduction

Given a graph $G = (V, E)$, a *trail* is a sequence $\tau := (v_0, e_0, v_1, e_1, \dots, e_{k-1}, v_k)$, where $(v_0, v_1, v_2, \dots, v_k)$ are nodes of G, $(e_0, e_1, e_2, \dots, e_{k-1})$ are distinct edges of G, and v_i and v_{i+1} are the endpoints of e_i for $0 \leq i \leq k-1$. The trail is a *path* if its nodes $(v_0, v_1, v_2, \dots, v_k)$ are distinct. In other words, a trail is a path that can pass more times through the same node. A path or a trail may consist of a single node.
A *dominating trail* D_τ in G is a trail such that each edge of G has at least one endpoint belonging to it (i.e., a dominating trail *covers* all the edges of G). Note that a dominating trail may not exist on G. A *dominating trail set* Σ is a collection of edge-disjoint trails that altogether cover all the edges of G. A *minimum dominating trail set* ($MDTS$) is a dominating trail set of minimum cardinality.
The problem of finding a $MDTS$ is closely related to the Hamiltonicity of a graph. A graph G is called Hamiltonian if it has a Hamiltonian path. The problem of finding the minimum number of edges which need to be added to G to make it Hamiltonian is known in literature as the problem of finding the *Hamiltonian completion number* of a graph and it is usually denoted as $HCN(G)$. In particular, $MDTS$ is related to $HCN(G)$ restricted to a particular

[1] E-mail: `detti@dii.unisi.it`.
[2] E-mail: `meloni@deemail.poliba.it`.
[3] E-mail: `mpranzo@dia.uniroma3.it`.

class of graphs, called *line graphs*. The line graph $L(G)$ of $G = (V, E)$ is a graph having $|E|$ nodes, each node of $L(G)$ being associated to an edge of G. There is an edge between two nodes of $L(G)$ if the corresponding edges of G are adjacent. Harary and Nash-Williams [5] link the problem of finding $HCN(L(G))$ and $MDTS$ showing that the line graph $L(G)$ of a graph G has a Hamiltonian path if and only if G has a dominating trail. As a consequence, if $HCN(L(G)) = k$ then the cardinality of $MDTS$ of G is $k + 1$.
The problem of finding $HCN(L(G))$ is well known to be NP–hard [2], even when G is bipartite [1]. The motivation for studying the $MDTS$ problem originates from applications in scheduling [1] and in other domains [4], such as data structures updating, genetics and combinatorial chemistry.
In this paper we study the $MDTS$ problem on the well known class of series parallel graphs. Series parallel graphs appear in several applications, e.g., electrical and electronics circuit analisys and design, scheduling problems, dynamic programming algorithm design. A well-studied problem is the recognition of series parallel graphs and there are linear time algorithms for this problem [6]. Since a *binary decomposition tree* (BDT) for a series parallel graph can be obtained in linear time [6], then many problems can be polynomially solved, including many problems that are NP-hard for general graphs. These facts motivate the study of the $MDTS$ problem on that particular graph class.
The paper is organized as follows. In Section 2, some special subgraphs are first introduced, then their series parallel composition properties are discussed. On the basis of some subgraphs composition rules, Section 3 addresses an algorithmic approach for the $MDTS$ problem on series parallel graphs.

2 Subgraphs definitions and composition rules

In this section, two terminal series parallel ($TTSP$) graph families with special dominating properties are introduced.
The following, notation is used. Given a $TTSP$ graph $G = (V, E, s, t)$, we say that a node i is an internal node of G if $i \in V \setminus \{s, t\}$. Two trails on G are distinct if they have no edges in common.
Let $H = (V_H, E_H, s_H, t_H)$ be a $TTSP$ subgraph of G connected with G only by the two terminal nodes s_H and t_H, and let τ be a trail in G. We say that τ *crosses* H if (i) dominates all edges of H, (ii) contains a trail $(v_x, e_x, s_H, \dots, t_H, e_y, v_y)$, where $(s_H, \dots, t_H)$ is all contained in H and $v_x, e_x, v_y, e_y \in G \setminus H$. In other words, τ *crosses* H if enters in s_H, dominates all edges of H and exits from t_H.
We say that τ *loops* in H from s_H if τ (i) dominates all edges of H, (ii) contains a trail $(v_x, e_x, s_H, \dots, s_H, e_y, v_y)$, where $(s_H, \dots, s_H)$ is a trail in H and $v_x, e_x, v_y, e_y \in G \setminus H$. In other words, τ loops in H from s_H if enters and exits from s_H dominating all edges of H. Similarly, τ *loops* in H from t_H if τ (i) dominates all edges of H, (ii) contains a trail $(v_x, e_x, t_H, \dots, t_H, e_y, v_y)$.
We say that τ *cycles* H if: τ (i) dominates all edges of H, (ii) contains a trail $(v_x, e_x, s_H, \dots, t_H, \dots, s_H, e_y, v_y)$, where $(s_H, \dots, t_H, \dots, s_H)$ is a trail in H and $v_x, e_x, v_y, e_y \in G \setminus H$. In other words, τ cycles in H if enters in s_H passes in t_H and exits from s_H dominating all edges of H. Note that if τ cycles H then it loops in H both from s_H and from t_H.
Finally, we say that H can be *externally dominated* if a trail that loops in H (both from s_H

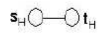
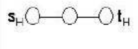
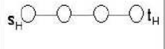
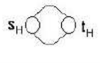
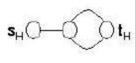
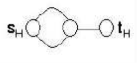
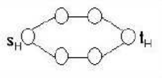
Class	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$
Graph							

Figure 1. The smallest graphs belonging to families $\mathcal{A} - \mathcal{G}$

and t_H) does not exist in G, and if H can be dominated by one or two trails in G that pass both in s_H and in t_H. Note that such trails must not necessarily contain internal nodes of H.
In Definition 1, six graph families are introduced.

Definition 1 *Let $H = (V_H, E_H, s_H, t_H)$ be a TTSP subgraph of G connected with G only by the two terminal nodes s_H and t_H. Let $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, $\mathcal{E}$, $\mathcal{F}$ and $\mathcal{G}$ be disjoint graph families containing graphs with at least one edge, and such that H belongs to family:*

$\mathcal{A}$, if trails that cycle H do not exist;
$\mathcal{B}$, if trails that loop in H or cycle H do not exist;
$\mathcal{C}$, if trails that loop in H or cycle H do not exist and if H cannot be externally dominated;
$\mathcal{D}$, if trails that cross H and cycle in H exist in G and if H can be externally dominated;
$\mathcal{E}$, if trails that cycle H or loop in H from s_H do not exist in G;
$\mathcal{F}$, if trails that cycle H or loop in H from t_H do not exist in G;
$\mathcal{G}$, if trails that cross H do not exist in G.

Note that, graphs in family $\mathcal{G}$ are the only graphs in which trails crossing H do not exist. Moreover, note that two distinct trails from s_H to t_H exist in H only if trails cycling in H exist. In Figure 1, the smallest graphs belonging to families $\mathcal{A} - \mathcal{G}$ are reported. It is easy to see that each of them can be dominated by a single dominating trail. In the following, two lemmas are reported, without proofs, in which some domination properties of the graph families are given.

Lemma 1 *Let $H = (V_H, E_H, s_H, t_H)$ be a TTSP subgraph of G connected with G only by the two terminal nodes s_H and t_H. If H belongs to: i) families $\mathcal{A}$ or $\mathcal{D}$ then there exists a $MDTS$ on G containing a trail passing in s_H or in t_H; ii) family $\mathcal{B}$ then there exists a $MDTS$ on G containing a trail(s) passing in s_H and in t_H; iii) family $\mathcal{C}$ then there exists a $MDTS$ on G containing a trail crossing H.*

Lemma 2 *Let $H = (V_H, E_H, s_H, t_H)$ be a TTSP subgraph of G connected with G only by the two terminal nodes s_H and t_H. If H belongs to family $\mathcal{E}$ (family $\mathcal{F}$) then there exists a $MDTS$ on G containing either a trail crossing H or one looping from t_H (from s_H).*

In Table 9 (10) the series (parallel) composition of graph families (Definition 1) is introduced, respectively. By Definition 1, it is easy to prove all the series and parallel compositions of the different graph families. Consider, for example, the series composition between a graph $G_D = (V_D, E_D, s_D, t_D)$ of family $\mathcal{D}$ and a graph $G_E = (V_E, E_E, s_E, t_E)$ of family $\mathcal{E}$ obtained by setting $t_D = s_E$. By definition, G_D admits both a crossing, a looping and a cycling trail, whereas G_E has a crossing trail and a looping trail from t_E. Therefore, the resulting graph

$*$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$
$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{E}$	$\mathcal{B}$	$\mathcal{B}$	
$\mathcal{B}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{C}$	
$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	
$\mathcal{D}$	$\mathcal{F}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{B}$	$\mathcal{B}$	$\mathcal{G}$
$\mathcal{E}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{C}$	
$\mathcal{F}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{B}$	$\mathcal{B}$	$\mathcal{C}$	
$\mathcal{G}$				$\mathcal{G}$			$\mathcal{G}$

Table 9
Series composition rules.

$\|\|$	$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{E}$	$\mathcal{F}$	$\mathcal{G}$
$\mathcal{A}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$
$\mathcal{B}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$
$\mathcal{C}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{G}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{C}$
$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$
$\mathcal{E}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$
$\mathcal{F}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$
$\mathcal{G}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{C}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{D}$	$\mathcal{G}$

Table 10
Parallel composition rules.

admits a crossing trail (the crossing trail in G_D plus the crossing trail in G_E), but does not admit looping or cycling trails. Moreover, it can be externally dominated, for example, by two trails that respectively loop in G_D from s_D and loop in G_E from s_E. As a consequence the resulting graph of the series composition belongs to family $\mathcal{B}$.
Regarding Table 9 we point out that graph family $\mathcal{C}$ always generates $\mathcal{C}$ whatever the other series graph family is considered. We note that Table 10 is symmetrical and a graph of family $\mathcal{D}$ is generated in all the parallel compositions but four. Moreover $\mathcal{A}$ is never generated in any composition rule. We note that there are some *open cases* in Table 9 corresponding to the empty entries. In these particular open cases the composed subgraph does not fall in any previously introduced graph family. As it will be pointed out in the next section, the analysis of such open cases is important for finding a $MDTS$ on $TTSP$ graphs.

3 Algorithmic results

The connection properties of the graph may play an important role in the solution of the $MDTS$ problem on $TTSP$ graphs. Colbourn and Stewart [3] proposed an algorithm to find the dominating trail in biconnected $TTSP$ graphs. Note that, if a $TTSP$ graph is not biconnected, then its blocks form a path, i.e., every cut vertex of the graph is in exactly two blocks, all blocks have at most two cut vertices, and there are exactly two blocks that contain only one cut vertex. The subgraph analysis presented in the previous sections allows to construct an algorithm for the $MDTS$ problem starting from the BDT (binary decomposition tree) representation. Given a $TTSP$ graph G, applying the series-parallel compositions described by the BDT, two situations may occur: i) no *open cases* are formed; ii) some *open case* is obtained.

The first situation is the most favorable, indicating that the graph G can be dominated by a single trail or cycle. The characteristics of subgraphs families and the composition rules presented in the previous sections give a way to easily find the (unique) dominating trail. Note that the case addressed in [3] falls in this situation.
In the second situation the solution of $MDTS$ may contain more than one trail. In fact, it can be shown that when an *open case* subgraph is involved in series parallel compositions a single dominating trail may not exist on the resulting graph. An optimal solution for the whole graph can be constructed introducing a procedure able to determine the best way to dominate the *open case* subgraphs.

References

[1] Agnetis, A., Detti, P., Meloni, C., Pacciarelli, D., 2001, Set-up coordination between two stages of a supply chain, *Annals of Operations Research*, 107, 15–32,
[2] Bertossi, A.A., 1981, The edge Hamiltonian problem is NP-hard, *Information Processing Letters*, 13, 157–159.
[3] Colbourn, C.J., Stewart, L.K., 1985, Dominating cycles in series-parallel graphs, *Ars Combinatoria*, 19, 107–112.
[4] Golumbic, M.C., 1980, Algorithmic graph theory and perfect graphs, Academic Press, New York.
[5] Harary, F., Nash-Williams, C.St.J.A., 1965, On Eulerian and Hamiltonian graphs and line-graphs, *Canadian Mathematics Bulletin*, 8, 701–709.
[6] Valdes, J., Tarjan, R.E., Lawler, E.L., 1982, The recognition of series parallel digraphs, *SIAM Journal on Computing*, 11, 298-313.

On the generation of bicliques of a graph

Vânia M. F. Dias [1]

COPPE, Universidade Federal do Rio de Janeiro, Brazil.

Celina M. H. de Figueiredo [2]

IM and COPPE, Universidade Federal do Rio de Janeiro, Brazil.

Jayme L. Szwarcfiter [3]

COPPE, IM and NCE, Universidade Federal do Rio de Janeiro, Brazil.

1 Introduction

Generating all configurations that satisfy a given specification is a well-studied problem in Combinatorics and in Graph theory suggesting many interesting problems. Among them, generating all maximal independent sets of a given graph is one that has attracted considerable attention [3, 4, 7]. A *maximal independent set* of a graph $G = (V, E)$ is a subset $V' \subseteq V$ such that no two vertices in V' are adjacent by an edge in E, and such that each vertex in $V - V'$ is adjacent to some vertex in V'.

Let $s_1, \ldots, s_{|S|}$ and $t_1, \ldots, t_{|T|}$ be two distinct sequences S and T, respectively and with $|S| \leq |T|$, of elements of an ordered set. Say that S is *lexicographically smaller* than T, if the first position i with respect to which S and T disagree satisfies: s_i is smaller than t_i or $i > |S|$. Johnson, Yannakakis and Papadimitriou [3] showed that there is no polynomial-delay algorithm for generating all maximal independent sets of a given graph in reverse lexicographic order, unless $P = NP$.

We examine the generation of bicliques in a graph. A *biclique* of a graph $G = (V, E)$ is an inclusion maximal induced complete bipartite subgraph of a graph, i.e. a pair (U, W) of subsets of V, such that both U and W are independent sets of G, every vertex in U is adjacent to every vertex in W, and such that for each vertex v in $V - (U \cup W)$, if $\{v\} \cup U$ is an independent set, then v is nonadjacent to some vertex in W, and if $\{v\} \cup W$ is an independent set, then v is nonadjacent to some vertex in U. When the requirement that U and W are independent sets of G is dropped, we have a *non-induced biclique*.

[1] E-mail: `vaniad@cos.ufrj.br`.
[2] E-mail: `celina@cos.ufrj.br`.
[3] E-mail: `jayme@nce.ufrj.br`.

Prisner [6] gives upper bounds on the number of bicliques in bipartite graphs and general graphs, exhibits examples of classes of graphs where the number of bicliques is exponential, and characterizes classes of graphs where the number of bicliques is polynomial in the number of vertices of the graph. The NP-completenes of the weighted maximum edge biclique problem for bipartite graphs is established by Dawanke et al. [2], and more recently for the non-weighted version by Peeters [5]. Alexe et al. [1] describe an algorithm for generating all non-induced bicliques with polynomial delay. The generation of all bicliques of a general graph with polynomial delay is an open problem.

We show it is NP-complete to test whether a subset of the vertices of a graph is part of a biclique. In light of this result, it might not be obvious how to obtain a polynomial-delay algorithm for generating all the (induced) bicliques. We show there is no polynomial-delay algorithm for generating all bicliques in reverse lexicographic order, unless $P = NP$. Nevertheless, we describe a greedy algorithm that generates the lexicographically first biclique of a graph $G = (V, E)$ in time $O(|V|^2)$. In a more complete version of this paper, we describe an algorithm for generating both all the (induced) bicliques and maximal independent sets with polynomial delay. In addition, we also propose specialized efficient algorithms for generating the bicliques of special classes of graphs.

2 NP-completeness results

Given a graph $G = (V, E)$, we say that a subset $S \subset V$ is *part* of a biclique (U, W) of G, if $S = U$ or $S = W$. In this section we establish the NP-completeness of two biclique decision problems defined next, by reducing the NP-complete problem SATISFIABILITY to each of them.

SATISFIABILITY
Instance: Set $X = \{x_1, \ldots, x_k\}$ of k Boolean variables, collection $C = \{c_1, \ldots, c_n\}$ of $n > 1$ clauses over literals of X.
Question: Is there a truth assignment satisfying C?

PART OF BICLIQUE
Instance: Graph $G = (V, E)$, subset $S \subset V$.
Question: Is S part of a biclique?

LEXICOGRAPHICALLY LAST BICLIQUE
Instance: Graph $G = (V, E)$, biclique B, order on V.
Question: Is there a biclique B' of G such that B' is lexicographically larger than B?

Theorem 1 *Given a graph $G = (V, E)$ and a subset $S \subset V$, it is NP-complete to decide whether S is part of a biclique.*

Proof. Problem PART OF BICLIQUE is in NP since a short certificate is a biclique (U, S) having subset S as a part. We can verify in polynomial time that (U, S) induces a complete bipartite subgraph, and in order to verify its maximality, we can verify in polynomial time for each vertex v in $V-(U\cup S)$ such that $\{v\}\cup U$ is an independent set that v is nonadjacent to some vertex in S, and for each vertex v in $V-(U\cup S)$ such that $\{v\}\cup S$ is an independent set that v is nonadjacent to some vertex in U.

To show completeness, we sketch a polynomial reduction from SATISFIABILITY. Given an instance (X,C) of SATISFIABILITY we construct in polynomial time a graph $G=(V,E)$ and a subset $S\subset V$, such that there is a truth assignment satisfying C if and only if S is part of a biclique of G.

Let ℓ_{ij} be the j-th literal of clause c_i. Vertex set V is the union $V=W\cup S\cup Z$, where there is a $w_{ij}\in W$ corresponding to each ℓ_{ij}; $S=\{v_1,\ldots,v_n\}$; $Z=\{z_1,\ldots,z_n\}$. Edge set E is the union $E=W'\cup S'\cup Z'$, where $W'=\{(w_{ij},w_{p\ell}):\ell_{ij}=\neg\ell_{p\ell}\}$; $S'=\{(v_i,w_{p\ell}):v_i\in S, w_{p\ell}\in W\}$; $Z'=\{(z_i,w_{p\ell}):z_i\in Z, w_{p\ell}\in W, i\neq p\}$.

Let T be a truth assignment satisfying C. We define an independent set $U\subseteq W$ such that (U,S) is a biclique of G. Define first U_1 as the set of vertices of W that correspond to literals with value true in T. By definition, U_1 is an independent set of G, containing at least one vertex w_{ij}, for each $i=1,\ldots,n$. Now let U be a maximal independent set of G containing U_1. Note $U\subseteq W$. Clearly (U,S) induces a complete bipartite subgraph. In order to verify that (U,S) is a biclique, we recall that $|C|>1$ and note that every vertex $z_i\in Z$ in nonadjacent to a vertex $w_{ij}\in U_1\subseteq U$.

Conversely, let (U,S) be a biclique of G. Since $Z\cup S$ is an independent set of G, we have $U\subseteq W$. The maximality of (U,S) says that for each $z_i\in Z$ there exists $w_{ij}\in U$. Since U is a stable set, each $w_{ij}\in U$ corresponds to a literal whose complementary literal does not belong to U. So, for each clause c_i, there exists at least one corresponding vertex $w_{ij}\in U$, which gives the desired truth assignment satisfying C. ∎ □

Corollary 1 *Given a graph $G=(V,E)$ and a vertex $v\in V$, it is NP-complete to decide whether $\{v\}$ is part of a biclique.*

Theorem 2 *Given a graph $G=(V,E)$, a biclique B, and an order on V, it is coNP-complete to decide whether B is the lexicographically last biclique.*

Proof. Problem LEXICOGRAPHICALLY LAST BICLIQUE is in *coNP* since a short certificate is a biclique B' that is lexicographically larger than B.

To show completeness, we sketch a polynomial transformation from SATISFIABILITY. Given an instance (X,C) of SATISFIABILITY we construct in polynomial time a graph $G=(V,E)$, a biclique B, and an order on V, such that there is a truth assignment satisfying C if and only if B is the lexicographically last biclique.

Let ℓ_{ij} be the j-th literal of clause c_i. Vertex set V is the union $V=Y\cup Z\cup W$, where $Y=\{y_1,\ldots,y_n,y'_1,\ldots,y'_n\}$, set Y contains a pair of vertices y_i,y'_i corresponding to each clause c_i; $Z=\{z,z'\}$; set W contains a pair of vertices w_{ij},w'_{ij} corresponding to each ℓ_{ij}. Edge set E is the union $E=Y^*\cup Z^*\cup W^*$, where $Y^*=\{(y_i,y'_j): i=1,\ldots,n \text{ and } j=1,\ldots,n\}\cup\{(y_i,w'_{p\ell}): i=1,\ldots,n \text{ and } p\neq i\}\cup\{(y'_i,w_{p\ell}): i=1,\ldots,n \text{ and } p\neq i\}$; $Z^*=$

$\{(z,z')\} \cup \{(z',y_i) : i = 1,\ldots,n\}$; $W^* = \{(w_{ij}, w'_{ij}) : w_{ij}, w'_{ij} \in W\} \cup \{(w_{ij}, w'_{p\ell}), (w'_{ij}, w_{p\ell}) : i \neq p$ and $\ell_{ij} \neq \neg\ell_{p\ell}\}$.

Each clause c_i corresponds to an edge (y_i, y'_i). The graph G is bipartite, since $V = R \cup Q$, where $R = \{z\} \cup \{y_1,\ldots,y_n\} \cup \{w_{i,j}\}$ and $Q = V \setminus R$ are both independent sets. Note that $B = (z', z \cup \{(y_1,\ldots,y_n\})$ is a biclique, as $N(z) = \{z'\}$, and $N(z') = \{z\} \cup \{(y_1,\ldots,y_n\}$. In addition, B is the only biclique containing vertex z. Every biclique $B' \neq B$ such that $z' \in B'$ is such that $y'_i \in B'$, for some $i = 1,\ldots,n$. In addition, B is the only biclique containing y_i but not y'_i, for some $i = 1,\ldots,n$.

The structure of the graph G gives the following equivalence: there is a biclique of G contained in W if and only if there is a truth assignment satisfying C. Clearly, a complete bipartite contained in W that cannot be enlarged by addition of $y \in Y$ is a complete bipartite that meets all clauses and that consequently gives a truth assignment satisfying C.

Now, the order on V is defined as follows. Each vertex y'_i has label i, $i = 1,\ldots,n$, each vertex y_i has label $n+i$, $i = 1,\ldots,n$; vertices z and z' have labels $2n+1$ and $2n+2$, respectively; each vertex $w \in W$ has a label $j \geq 2n+3$.

Clearly, a biclique of G contained in W is lexicographically larger than B. Conversely, let $B' = (U, U')$ be a biclique lexicographically larger than B. Hence $y'_i \notin B'$, $i = 1,\ldots,n$. We conclude that $U \cup U' \subseteq W$. ∎ □

Corollary 2 *Given a graph $G = (V,E)$, a biclique B, and an order on V, it is coNP-complete to test if B is the lexicographically last non-induced biclique.*

3 Greedy generation of lexicographically first biclique

Let $G = (V,E)$ be a connected graph, with an order on $V = \{1,\ldots,n\}$. We describe an $O(n^2)$ greedy algorithm to generate the lexicographically first biclique.

Let $B = (U,W)$ be the lexicographically first biclique. Clearly, $1 \in B$, assume $1 \in U$. Hence, $W \subseteq N(1)$. Let j be the smallest element of $N(1)$. Either $j \in B$, which means that j is the smallest element of W; or $j \notin B$, which means that there exists a vertex i, with $1 < i < j$ such that $i \notin N(j)$ and $N(i) \cap N(1) \neq \emptyset$.

The proposed algorithm consists of two phases. Phase 1 finds $j' \geq j$ the smallest element of W, and $U' = U \cap \{1,\ldots,(j'-1)\}$. Phase 2 enlarges subset U' to U and subset $\{j'\}$ to W, by scanning vertices $j' < k \leq n$ in the specified order, and by accepting every vertex k that satisfies precisely one of the situations: vertex k is nonadjacent to every vertex already accepted to be in U and vertex k is adjacent to every vertex already accepted to be in W; or vertex k is nonadjacent to every vertex already accepted to be in W and vertex k is adjacent to every vertex already accepted to be in U.

References

[1] G. Alexe, S. Alexe, S. Foldes, P. L. Hammer, B. Simeone, *Consensus algorithms for the generation of all maximal bicliques*, Rutgers University, RUTCOR Research Report RRR 23-2000, and DIMACS Technical Report 52-2002.

[2] M. Dawanke, P. Keskinocak, J. Swaminathan, S. Tayur, *On the biclique and multipartite clique problems*, J. Algorithms 41 (2001), pp. 388–403.

[3] D. S. Johnson, M. Yannakakis, C. H. Papadimitriou, *On generating all maximal independent sets*, Inform. Process. Lett. 27 (1988), pp. 119–123.

[4] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, *Generating all maximal independent sets: NP-hardness and polynomial-time algorithms*, SIAM J. Comput. 9 (1980), pp. 558–565.

[5] R. Peeters, *The maximum edge biclique problem is NP-complete*, Discrete Appl. Math. 131 (2003), pp. 651–654.

[6] E. Prisner, *Bicliques in graphs. I. Bounds on their number*, Combinatorica 20 (2000), pp. 109–117.

[7] S. Tsukiyama, M. Ide, H. Ariyoshi, I. Shirakawa, *A new algorithm for generating all the maximal independent sets*, SIAM J. Comput. 6 (1977), pp. 505–517.

An Improved Discrepancy Approach to Declustering

Benjamin Doerr Nils Hebbinghaus[1] Sören Werth[1]

Mathematisches Seminar, Bereich II, Christian-Albrechts-Universität zu Kiel, Christian-Albrechts-Platz 4, 24118 Kiel, Germany.

1 Introduction

The last decade saw dramatic improvements in computer processing speed and storage capacities. Nowadays, the bottleneck in data-intensive applications is disk I/O, the time needed to retrieve typically large amount of data from storage devices. One idea to overcome this obstacle is to spread the data on disks of multi-disk systems so that they can be retrieved in parallel. The data allocation is determined by declustering schemes. Their aim is to allocate the data in such a manner that typical requests find their data evenly distributed on the disks.

The declustering problem is to assign data blocks from a multi-dimensional grid system to one of M storage devices in a balanced manner. More precisely, our grid is $V = [n_1] \times \cdots \times [n_d]$ for some positive integers $n_1, \ldots, n_d$.[2] A query Q requests the data assigned to a sub-grid $[x_1..y_1] \times \cdots \times [x_d..y_d]$ for some integers $1 \le x_i \le y_i \le n_i$. We assume that the time to process such a query is proportional to the maximum number of requested data blocks that are stored in a single device. If we represent the assignment of the data blocks to the devices through a mapping $\chi : V \to [M]$, then the query time of the query above is $\max_{i \in [M]} |\chi^{-1}(i) \cap Q|$, where we identify the query Q with its associated sub-grid. Clearly, no declustering scheme can do better than $|Q|/M$. Hence a natural performance measure is the additive deviation from this lower bound.

This makes the problem a combinatorial discrepancy problem in M colors. Denote by $\mathcal{E}$ the set of all sub-grids in V. Then $\mathcal{H} = (V, \mathcal{E})$ is a hypergraph. For a coloring $\chi : V \to [M]$, the positive discrepancy of $\mathcal{H}$ with respect to χ and the positive discrepancy of $\mathcal{H}$ in M colors are

$$\begin{aligned}\operatorname{disc}^+(\mathcal{H}, \chi) &:= \max_{i \in [M], E \in \mathcal{E}} \left(|\chi^{-1}(i) \cap E| - \tfrac{1}{M}|E| \right), \\ \operatorname{disc}^+(\mathcal{H}, M) &:= \min_{\chi : V \to [M]} \operatorname{disc}^+(\mathcal{H}, \chi)\end{aligned}$$

[1] E-mail: {bed,nhe,swe}@numerik.uni-kiel.de. The authors thank the DFG-Graduiertenkolleg 357 “Effiziente Algorithmen und Mehrskalenmethoden” for supporting this research.

[2] We use the notations $[n] := \{1, 2, \ldots, n\}$ and $[n..m] := \{n, n+1, \ldots, m\}$ for $n, m \in \mathbb{N}$, $n \le m$.

Apart from the fact that we only regard positive deviations, these notions were introduced by Srivastav and the first author in [DS03]. Independently, Anstee, Demetrovics, Katona and Sali [ADKS00] and Sinha, Bhatia and Chen [SBC03] proved a lower bound of $\Omega(\log M)$ for the additive error of any declustering scheme in dimension two. Sinha et al. [SBC03] also give the bound $\Omega(\log^{\frac{d-1}{2}} M)$ for arbitrary $d \geq 3$, but their proof contains a crucial error.

The current best upper bounds in arbitrary dimension for the declustering schemes are proposed by C.-M. Chen and C. Cheng [CC02]. They present two schemes for d–dimensional problems with an additive error $O(\log^{d-1} M)$. The first one works if $M = p^k$ for some $k \in \mathbb{N}$ and p a prime such that $p \geq d$, whereas the second works for arbitrary M, but the error increases with N

Our Results: For the upper bounds, we present an improved scheme that yields an additive error of $O(\log^{d-1} M)$ for a broader range of values of M and independent of the data size. Our requirement on M is that if $M = q_1 \ldots q_k$, $q_1 < \ldots < q_k$ is the canonical factorization of M into prime powers, $d \leq q_1 + 1$ is needed. Thus, in particular, our schemes work for M being a power of two (such that $M \geq d-1$), which is very useful from the view-point of application. We also show that the latin hypercube construction used by Chen et al. [CC02] is much better than claimed. Where they show that the latin hypercube coloring extended to the whole grid has an error of at most 2^d times the one of the latin hypercube, we show that both errors are the same.

For the lower bound, we present the first correct proof of the $\Omega(\log^{\frac{d-1}{2}} M)$ bound. Again, a more careful analysis shows that that the positive discrepancy is at least $\frac{1}{2d}$ times the normal discrepancy instead of 3^{-d} as used in [SBC03]. Note that in typical applications with M between 16 and 1024, these 2^d and 3^d factors are at least as important as finding the right exponent of the $\log M$ term.

Since a central result of this paper are discrepancy bounds that are independent of the size of the grid, we usually work with the hypergraph $\mathcal{H}_N^d = ([N]^d, \mathcal{E}_N^d)$, $\mathcal{E}_N^d = \{\prod_{i=1}^d [x_i..y_i] \mid 1 \leq x_i \leq y_i \leq N\}$ for some sufficiently large integer N. We prove the following result.

Theorem 1 *Let M, $d \geq 2$ be positive integers and q_1 the smallest prime power in the canonical factorization of M into prime powers. We have*

(i) $\operatorname{disc}^+(\mathcal{H}_N^d, M) = O(\log^{d-1} M)$ *for $d \leq q_1 + 1$, independently of $N \in \mathbb{N}$.*
(ii) $\operatorname{disc}^+(\mathcal{H}_N^d, M) = \Omega(\log^{\frac{d-1}{2}} M)$ *for $N \geq M$.*

The combinatorial discrepancy results are shown via strong results from geometric discrepancy theory. The problem of geometric discrepancy in the unit cube $[0,1[^d$ is to distribute $n \in \mathbb{N}$ points evenly with respect to axis-parallel boxes: In every box R should be approximately $n \operatorname{vol}(R)$ points, where $\operatorname{vol}(R)$ denotes the volume of R. Again, discrepancy quantifies the distance to a perfect distribution. The discrepancy of a point set $\mathcal{P}$ with respect to a box $R \subseteq [0,1[^d$ and the set of all axis-parallel boxes $\mathcal{R}_d$ are defined by

$$D(\mathcal{P},R) = ||\mathcal{P} \cap R| - n\,\mathrm{vol}(R)|\,,$$
$$D(\mathcal{P},\mathcal{R}_d) = \sup_{R\in\mathcal{R}_d} |D(\mathcal{P},R)|.$$

2 The lower bound

The general idea in the proofs of the lower bound in Sinha et al. [SBC03] and Anstee et al. [ADKS00] is the same, here described in two dimensions:

Starting with an arbitrary M–coloring of $[M]^2$, there is a monochromatic set $\hat{P}$ with M vertices. Based on this set, an M–point set $\mathcal{P}$ in $[0,1[^2$ is constructed. By discrepancy theory [Sch72], there is a rectangle R such that $D(\mathcal{P},R) = \Omega(\log M)$. Rounding R to the $[M]^2$ grid, they construct a hyperedge $\hat{R}$ that has almost the volume as R. Additionally $\hat{R}$ contains as many vertices of $\hat{P}$ as R points of $\mathcal{P}$. With the help of $\hat{R}$ and a short calculation the lower bound of the additive error $\Omega(\log M)$ is shown.

The small, but crucial mistake in the proof of Sinha et al. [SBC03] is in the transfer from the geometric discrepancy setting back to the combinatorial one. Recall that the authors started with a color class of exactly M^{d-1} points. They down-scaled it by a factor of M to a set in the unit cube (that, note this fact, is a subset of $\{0, \frac{1}{M}, \frac{2}{M}, \ldots, \frac{M-1}{M}\}^d$). Then their geometric discrepancy argument yields a rectangle of polylogarithmic discrepancy. However, the rectangle $[0, \frac{M-1}{M}]^d$ has a much larger discrepancy: It contains all M^{d-1} points, but has a volume of $(\frac{M-1}{M})^d$ only.

This yields a discrepancy of $M^{d-1}(1 - (\frac{M-1}{M})^d) = \Omega(M^{d-2})$. For dimension $d \geq 3$ this is larger than the upper bound, what also indicates an error in the proof of Sinha et al. [SBC03]. The last argument also shows that rounding an arbitrary box to a box in the grid can cause a roundoff error, which is of magnitude larger than the discrepancy. For this reason, a direct generalization using the lower bound of Roth [Rot64] is not possible. A more careful analysis is needed. In particular, we have to ensure the existence of a *small* box having large discrepancy. Using ideas of Beck [BC87], we show

Theorem 2 *For any n–point set $\mathcal{P}$ in the unit cube $[0,1[^d$, there is an axis-parallel cube Q with side at most $n^{-\frac{(2d-3)d}{(d-1)^2(2d+1)}}$ fully contained in $[0,1[^d$ with*

$$D(\mathcal{P},Q) = \Omega(\log^{\frac{d-1}{2}} n).$$

Now Theorem 1 (ii) follows from Theorem 2 using the roundoff reduction of Anstee et al. [ADKS00] and Sinha et al. [SBC03].

3 The upper bound

We use geometric discrepancies to construct a declustering scheme for the proof of our upper bound. The notation of Niederreiter [Nie87] is used in the following. For an integer $b \geq 2$, an elementary interval in base b is an interval of the form $E = \prod_{i=1}^{d} \left[a_i b^{-d_i}, (a_i+1)b^{-d_i}\right[$, with integers $d_i \geq 0$ and $0 \leq a_i < b^{d_i}$ for $1 \leq i \leq d$. For integers t, m such that $0 \leq t \leq m$, a (t, m, d)–net in base b is a point set of b^m points in $[0, 1[^d$ such that all elementary intervals with volume b^{t-m} contain exactly b^t points. Note that any elementary interval with volume b^{t-m} has discrepancy zero in a (t, m, d)–net. Since any subset of an elementary interval of volume b^{t-m} has discrepancy at most b^t and any box can be packed with elementary intervals in a way that the uncovered part can be covered by $O(\log^{d-1} n)$ elementary intervals of volume b^{t-m}, the following is immediate:

Theorem 3 *A (t, m, d)–net $\mathcal{P}$ has discrepancy $D(\mathcal{P}, \mathcal{R}_d) = O(\log^{d-1} n)$.*

The central argument in our proof of the upper bound is the following result of Niederreiter [Nie87] on the existence of $(0, m, d)$–nets. From the view-point of application it is important that his proof is constructive.

Theorem 4 *Let $b \geq 2$ be an arbitrary base and $b = q_1 q_2 \ldots q_u$ be the canonical factorization of b into prime powers such that $q_1 < \cdots < q_u$. Then for any $m \geq 0$ and $d \leq q_1 + 1$ there exists a $(0, m, d)$–net in base b.*

We construct colorings of $\mathcal{H}_N^d$ from $(0, m, d)$–nets with small discrepancy. We start with colorings for $\mathcal{H}_M^d$ in Lemma 1.

Lemma 1 *Let $\mathcal{P}_{net}$ be a $(0, d-1, d)$–net in base M in $[0, 1[^d$. Then there is a M–coloring χ_M of $\mathcal{H}_M^d = ([M]^d, \mathcal{E}_M^d)$ such that all rows of $[M]^d$ contain every color exactly once and $\operatorname{disc}(\mathcal{H}_M^d, \chi_M) \leq 2D(\mathcal{P}_{net}, \mathcal{R}_d)$.*

In Lemma 2, we show that it is sufficient to consider the discrepancy of $\mathcal{H}_M^d$ with respect to these colorings for determining the upper bound of the discrepancy of $\mathcal{H}_N^d$. The Lemma 2 is a remarkable improvement of Theorem 4.2 in [CC02] , where $\operatorname{disc}(\mathcal{H}_N^d, \chi) \leq 2^d \operatorname{disc}(\mathcal{H}_M^d, \chi_M)$ is shown. Note that this reduces the implicit constant in the upper bound by factor of 2^d.

Lemma 2 *Let χ_M be a M–coloring of $\mathcal{H}_M^d$ such that all rows of $[M]^d$ contain every color exactly once and χ a coloring of $\mathcal{H}_N^d$ defined by $\chi(x_1, \ldots, x_d) = \chi_M(y_1, \ldots, y_d)$ such that $x_i \equiv y_i \mod M$ for $i \in [d]$, $x_i \in [N]$, $y_i \in [M]$. Then*

$$\operatorname{disc}(\mathcal{H}_N^d, \chi) = \operatorname{disc}(\mathcal{H}_M^d, \chi_M).$$

The upper bound in Theorem 1 follows from the above.

References

[ADKS00] R. Anstee, J. Demetrovics, G. O. H. Katona and A. Sali. Low discrepancy allocation of two-dimensional data. *Foundations of Information and Knowledge Systems, First International Symposium*, 1–12, 2000.

[BC87] J. Beck and W. L. Chen. *Irregularities of distribution*, volume 89 of *Cambridge Tracts in Mathematics.* Cambridge University Press, Cambridge, 1987.

[CC02] C.-M. Chen and C. Cheng. From discrepancy to declustering: near optimal multidimensional declustering strategies for range queries. *ACM Symp. on Database Principles*, 29–38, 2002.

[DS03] B. Doerr and A. Srivastav. Multicolour discrepancies. *Combinatorics, Probability and Computing*, 12:365–399, 2003.

[Nie87] H. Niederreiter. Point sets and sequences with small discrepancy. *Monatsh. Math.*, 104:273–337, 1987.

[Rot64] K. F. Roth. Remark Concerning Integer Sequences. *Acta Arithmetica*, 9:257–260, 1964.

[SBC03] R. K. Sinha, R. Bhatia and C.-M. Chen. Asymptotically optimal declustering schemes for 2-dim range queries. *Theoret. Comput. Sci.*, 296:511–534, 2003.

[Sch72] W. M. Schmidt. On irregularities of distribution VII. *Acta Arith.*, 21:45–50, 1972.

A Primal-Dual Method for Approximating Tree Cover with Two Weights

Takashi Doi, Toshihiro Fujito [1]

Graduate School of Information Science, Nagoya University, Furo, Chikusa, Nagoya 464-8603 Japan

1 Introduction

In an undirected graph $G = (V, E)$ a vertex is said to *cover* all the edges incident to it, and similarly, an edge *dominates* all the edges adjacent to it. A *vertex cover* is such a vertex set $C \subseteq V$ that collectively covers all the edges in G, whereas an *edge dominating set* is an edge set $D \subseteq E$ collectively dominating all the other edges in G. A *tree cover (tc)* for connected G is defined to be an edge set $T \subseteq E$ forming a "connected" edge dominating set. Or equivalently, it is a connected edge set s.t. its vertex set forms a vertex cover for G. The *vertex cover (VC)*, *edge dominating set (EDS)*, and *tree cover (TC)* problems are to compute a vertex cover, an edge dominating set, and a tree cover, respectively, of minimum weight in a graph, where either vertices or edges are associated with nonnegative weights. The problems VC and EDS are classic $\mathcal{NP}$-hard graph problems, and the TC problem is also $\mathcal{NP}$-hard even in the unweighted case (i.e., all the edge weights are equal) since it then becomes equivalent to the *connected vertex cover* problem, which in fact is known to be as hard (to approximate) as VC is [5]. Given the apparent intractability in exact computation, efficient approximation algorithms for these problems have been studied extensively in the literature. A factor 2 approximation of VC was found early; it suffices to compute any maximal matching M and output the set $V(M)$ of vertices matched by M. The best approximation algorithm known today for VC, weighted or unweighted, achieves a ratio of $2 - \frac{\log\log n}{2\log n}$ [2]. Likewise, weighted EDS was recently found 2-approximable as in the unweighted case [4]. The unweighted version of TC is also known to be approximable within a factor of 2 by simple algorithms [7, 1]. On the other hand, TC with general weights was first shown to be approximable within a factor of 3.55 [1], and currently the best bound known is $3 + \epsilon$ [6, 3]. Thus, general TC is not yet known to be approximable as good as the unweighted version is. Even worse, the algorithm of [6] or [3] is far from practical in its efficiency; it requires to solve optimally an LP of huge size (ref. (1)), and to do so, it inevitably resorts to calling the ellipsoid method as its subroutine. In this paper we consider a restricted version of the weighted TC problem, as a first step towards better approximation

[1] E-mail: `fujito@nuee.nagoya-u.ac.jp`. Corresponding author; Partially supported by a Grant in Aid of MESSC of Japan.

of general TC, where edge weights are limited to either w_1 or w_2 satisfying $w_2 \geq 2w_1$. It will be shown that a factor 2 approximation can be attained efficiently (in the complexity of max flow) in this case by the primal-dual method.

2 Preliminaries

For a designated vertex r called *root*, an r-tc is a tree cover touching r, and *r-TC* is the problem of computing a minimum weight r-tc. As it suffices to approximate r-TC well for our purpose, we focus on this variant instead of approximating TC directly. Given an undirected graph $G = (V, E)$ with edge weights $c : E \to \mathbb{Q}_+$, let $\vec{G} = (V, \vec{E})$ denote its directed version obtained by replacing each edge $\{u, v\}$ of G by two anti-parallel arcs, (u, v) and (v, u), each having a weight $c(\{u, v\})$. A non-empty set $S \subseteq V - \{r\}$ is called *dependent* if S induces an edge in G. Let $\mathcal{D}$ be the family of such dependent sets (i.e., $\mathcal{D} = \{S \subseteq V - \{r\} \mid S \text{ is dependent in } G\}$). Suppose $T \subseteq E$ is an r-tc, and let $\vec{T}$ denote the directed tree obtained from T by directing each edge of T away from r to a leaf. Clearly, $c(T) = c(\vec{T})$. Moreover, $x^{\vec{T}} \in \{0,1\}^{\vec{E}}$, the characteristic vector of $\vec{T}$, satisfies the linear inequality $x(\delta^-(S)) \geq 1$ for all dependent sets $S \subseteq V - \{r\}$, where $\delta^-(S) = \{\, (u, v) \in \vec{E} \mid u \notin S, v \in S\,\}$, and $x(\vec{F}) = \sum_{a \in \vec{F}} x_a$ for $x \in \mathbb{Q}^{\vec{E}}$ and $\vec{F} \subseteq \vec{E}$, because at least one arc of $\vec{T}$ must enter S when an edge exists inside it. Thus, the following LP is a relaxation of r-TC, and its integral solutions are the ones we will be actually seeking for:

$$\text{(LP)} \quad \text{subject to:} \quad \begin{array}{ll} \min \sum_{a \in \vec{E}} c(a) x_a & \\ x(\delta^-(S)) \geq 1, & \forall S \in \mathcal{D} \\ x_a \geq 0, & \forall a \in \vec{E} \end{array} \qquad (1)$$

Unlike the algorithms in [3, 6], our algorithm also requires the dual of (LP):

$$\text{(D)} \quad \text{subject to:} \quad \begin{array}{ll} \max \sum_{S \in \mathcal{D}} y_S & \\ \displaystyle\sum_{S \in \mathcal{D} : a \in \delta^-(S)} y_S \leq c(a), & \forall a \in \vec{E} \\ y_S \geq 0, & \forall S \in \mathcal{D} \end{array}$$

3 Basic Approximation Techniques

In what follows it is assumed that the arcs in a rooted tree are always directed away from the root to a leaf, and for $a = (u, v) \in \vec{E}$, y_a is an abbreviation for $y_{\{u,v\}}$. In our algorithm we use the following primal-dual techniques; two for approximating unweighted tree cover, and one for exactly computing minimum weight vertex cover in bipartite graphs. For the first two, let $\vec{G} = (V, \vec{E})$ and $r \in V$ be an instance of unweighted r-TC (thus $c(a) = 1, \forall a \in \vec{E}$), and $\vec{T}$ be a spanning tree of $\vec{G}$ rooted at r.

T1. Let $\vec{T}$ be a dfs tree. Remove all the leaves from $\vec{T}$, and the resulting tree $\vec{T}'$ is an r-tc for $\vec{G}$ since no edge exists in G between any two leaves of $\vec{T}$. This algorithm of Savage was shown to be a factor 2 approximation using the counting argument [7]. It is also possible to assert the 2-approximability by the primal-dual argument as follows. For each non-leaf u of $\vec{T}$ with $u \neq r$, choose one child v of u, and mark the arc (u, v) "chosen". Let P be the set of these chosen arcs. Then, P consists of disjoint dipaths P_i's, each starting in some non-leaf and ending at some leaf of $\vec{T}$. Assign 1/2 to each of y_a for $a \in P$. Then, while $y \in \mathbb{Q}^{\vec{E}}$ is dual feasible in (D), it can be seen that $|\vec{T}'| = 2\sum_{a\in\vec{E}} y_a$ since $|\vec{T}'| = |P|$, and the 2-approximability thus follows.

T2. For a not necessarily dfs tree $\vec{T}$, construct a set P of disjoint dipaths as above, and label all the arcs in each dipath P_i with "1" and "0" alternatively, starting with " 1" at the first arc, continuing with "0" at the second, and so on. Let M_T be the set of all the arcs labeled "1"; M_T is then a matching in $\vec{T}$. For the set L' of leaves of $\vec{T}$ left unmatched by M_T, consider the subgraph $G[L']$ of G induced by L'. Then, $G[L']$ may contain some edges of G since $\vec{T}$ may not be a dfs tree. For any "maximal" matching $M_{L'}$ in $G[L']$, remove all such leaves of $\vec{T}$ that are unmatched by both M_T and $M_{L'}$, and let $\vec{T}'$ be the resulting tree. Then, $\vec{T}'$ is an r-tc for $\vec{G}$ since any edge joining leaves of $\vec{T}$ is covered by some leaf of $\vec{T}$ matched by either M_T or $M_{L'}$.

Setting $y_a = 1$ for each $a \in M_T \cup M_{L'}$, y can be seen dual feasible since $M_T \cup M_{L'}$ is a matching in G. It is also easy to observe that $|\vec{T}'| = 2\sum_{a\in\vec{E}} y_a$; for each arc a in M_T associate two arcs, a itself and the one preceding a in $\vec{T}$, and for each a in $M_{L'}$ associate the two arcs of $\vec{T}$ adjacent to a.

Suppose that it is actually the case in the above that every arc not in $\vec{T}$ has a weight ≥ 1 while $c(a) = 1$ for $a \in \vec{T}$. It will be useful to notice that y given above remains dual feasible under either type of the dual assignments.

T3. Suppose that $G = (A \cup B, E)$ is a bipartite graph and each vertex is associated with a nonnegative weight $b : A \cup B \to \mathbb{Q}_+$. A *b-matching* for G is a function $z : E \to \mathbb{Q}_+$ such that $z(\delta(u)) \leq b(u)$ for each vertex u in G, where $\delta(u)$ is the set of edges incident with u. Call the sum of the entries in a b-matching z (i.e., $\sum_{e\in E} z(e)$) as its *size*. Then it is a well-known fact (cf. Egerváry's theorem) that the maximum size of a b-matching is equal to the minimum weight of a vertex cover in any bipartite graph. Moreover, such an optimal vertex cover and an optimal b-matching can be found by a max flow computation.

4 Algorithm

Assume in what follows w.l.o.g. that edge weights are either 1 or w with $w \geq 2$, and call an arc of weight 1 (an arc of weight w, resp.) as *1-arc* (*w-arc*, resp.). Our algorithm works in two stages. First it computes a (minimum) spanning tree $\vec{T}$ of given $\vec{G}$. Then, it prunes certain leaves of $\vec{T}$, and to determine which leaves are to be pruned, it computes a vertex cover in a graph induced by the leaves of $\vec{T}$. By removing all the leaves of $\vec{T}$ excluded from this vertex cover, the resulting tree $\vec{T}'$ is output as an r-tc for $\vec{G}$. The first stage starts with:

(1) Construct a maximal forest $\mathcal{F}_1$ in $\vec{G} = (V, \vec{E})$ consisting of 1-arcs only.

(2) Shrink each tree in $\mathcal{F}_1$ into a vertex (naming anew the vertex as r if original r was shrunken into it), and compute a dfs spanning tree $\vec{T}_w$ rooted at r in the resulting graph.
(3) For each tree $\vec{T}_{1(i)}$ shrunken in Step (2) (but not into r), let u be the (unique) vertex in $\vec{T}_{1(i)}$ having an incoming arc of $\vec{T}_w$, and redirect all the arcs of $\vec{T}_{1(i)}$ so that it becomes a directed tree rooted at u.
(4) Construct $\vec{T}$ spanning in $\vec{G}$ by gluing together $\vec{T}_w$ and all $\vec{T}_{1(i)}$'s in $\mathcal{F}_1$.

Each leaf u of $\vec{T}$ is adjacent to either 1-arc or w-arc of $\vec{T}$, and we call u either 1-*leaf* or w-*leaf* accordingly. Let L_1 and L_w denote the sets of 1-leaves and w-leaves, respectively. The second stage resumes with:

(5) By applying **T2** of the previous section to each tree $\vec{T}_{1(i)}$ in $\mathcal{F}_1$, obtain a matching $M_{T_{1(i)}}$ within $\vec{T}_{1(i)}$. Let L_1' be the set of all such 1-leaves of $\vec{T}$ that are unmatched by all of $M_{T_{1(i)}}$'s.
(6) Construct a maximal matching $M_{L_1'}$ in $G[L_1']$, the subgraph of G induced by L_1', and let L_1'' be the set of leaves in L_1' left unmatched by $M_{L_1'}$.
(7) Consider $G[L_1'' \cup L_w]$. Then, $G[L_1'' \cup L_w]$ is bipartite as 1) no edge of G can exist between any two leaves of L_1' which are both left unmatched by a maximal matching $M_{L_1'}$, and 2) L_w consists of leaves of a dfs spanning tree. Apply **T3** to $G[L_1'' \cup L_w]$ with $b(u) = 1, \forall u \in L_1''$ and $b(u) = w, \forall u \in L_w$, and compute a minimum weight vertex cover L_{vc} and a maximum b-matching $z \in \mathbb{Q}^{E[L_1'' \cup L_w]}$.
(8) Prune all such leaves of $\vec{T}$ that are excluded from L_{vc} in Step (7) (i.e., those in $(L_1'' \cup L_w) - L_{\mathrm{vc}}$), and output the resulting tree $\vec{T}'$.

5 Analysis

By applications of **T2** to $\vec{T}_1(i)$'s and **T3** to $G[L_1'' \cup L_w]$, the dual variables y_a are set as follows; 1) within each $\vec{T}_1(i)$, $y_a = 1$ if $a \in M_{T_{1(i)}}$, 2) within $G[L_1']$, $y_a = 1$ if $a \in M_{L_1'}$, and 3) within $G[L_1'' \cup L_w]$, $y = z/2$ for a maximum b-matching z with $b(u) = 1, \forall u \in L_1''$ and $b(u) = w, \forall u \in L_w$. Distinguish the arcs a of a tree as either *leaf* or *non-leaf* according to whether a is incident with a leaf or not. It remains to account for the weights of the non-leaf arcs of $\vec{T}_w$ as well as the leaf arcs of $\vec{T}_{1(i)}$'s glued directly with $\vec{T}_w$ (thus *not* leaf arcs in $\vec{T}$). We apply **T1** to $\vec{T}_w$ with certain modifications: For each arc $a = (u, v)$ in P of $\vec{T}_w$, set $y_a = w/2$ (as in **T1**) if neither u nor v is a shrunken node, and in case either u or v is shrunken, set $y_a = 0$ if u is matched by $M_{T_{1(i)}}$, but otherwise, set $y_a = 1$. In addition, we need to set $y_{V(\vec{T}_{1(i)})} = w - 1$ for any $\vec{T}_{1(i)} \in \mathcal{F}_1$ if $r \notin V(\vec{T}_{1(i)})$. By careful examination, it can be verified that

Theorem 1 *The vector y of dual variables determined as above is feasible in (D). Moreover, it satisfies that $\sum_{a \in \vec{T}'} c(a) \leq 2 \sum_{S \in \mathcal{D}} y_S$ for the tree cover $\vec{T}'$ output by the algorithm.*

Therefore, by virtue of the (weak) duality theorem of LP, our algorithm approximates r-TC (and hence, TC) within a factor of 2.

References

[1] E.M. Arkin, M.M. Halldórsson and R. Hassin. Approximating the tree and tour covers of a graph. *Inform. Process. Lett.*, 47:275–282, 1993.

[2] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. In *Annals of Discrete Mathematics*, volume 25, pages 27–46, 1985.

[3] T. Fujito. On approximability of the independent/connected edge dominating set problems. In *Proc. 20th FSTTCS*, pages 117–126, 2000.

[4] T. Fujito and H. Nagamochi. A 2-approximation algorithm for the minimum weight edge dominating set problem. *Discrete Appl. Math.*, 118:199–207, 2002.

[5] M.R. Garey and D.S. Johnson. The rectilinear Steiner-tree problem is NP-complete. *SIAM J. Appl. Math.*, 32(4):826–834, 1977.

[6] J. Könemann, G. Konjevod, O. Parekh and A. Sinha. Improved approximations for tour and tree covers. In *Proc. 3rd APPROX*, pages 184–193, 2000.

[7] C. Savage. Depth-first search and the vertex cover problem. *Inform. Process. Lett.*, 14(5):233–235, 1982.

The Cut Number of the n-Cube, Boolean Methods and a Geometric Connection to Threshold Logic

M. R. Emamy-K. [1]

Dept. of Mathematics, POBOX 23355, University of Puerto Rico, Rio Piedras, Puerto Rico, 00931

The cut number $S(n)$ of the n-cube is the minimum number of the hyperplanes in R^n that slice, i.e. avoiding vertices, all the edges of the n-cube. The cut number problem for the hypercube of dimensions $n = 4$ was posed by P. O' Neil more than thirty years ago. The identity $S(3) = 3$ is easy and that of $S(4) = 4$ is a well-known result obtained by the author in 1988. Recently, Sohler-Ziegler [13] have obtained a computational solution to the 5-cube problem. To find a short and computer-free proof for the 5-cube will remain a challenging open problem. In the first part of the paper, we present the latest improvements on this theoretical approach, applying terminologies of Boolean methods in threshold logic.

On the other hand, the connection to threshold logic provides a geometric interaction between convex polytopes and threshold logic. The description of the latter, requires a few notations and definitions. By the n-cube we mean the Boolean n-cube i.e. the n-th power of the $\{0, 1\}$, unless otherwise stated. A cubical complex, or simply a complex, C is a nonempty collection of faces, i.e., subcubes, of the n-cube such that for any face $F1$ of $F2 \in C$, then $F1$ is also in C. A cut-complex is a cubical complex whose vertices are strictly separable from the rest of the vertices of the n-cube by a hyperplane in the n-dimensional Euclidean space.

Geometric characterization of non-isomorphic cut-complexes over the n-cube for arbitrary $n \geq 4$ is a hard problem that has a root in the paper of Grunbaum [9] and has been studied by the author et. al. in [2, 3, 4, 5, 6, 7]. They were actually introduced in [2] as the main tool in a solution to the cut number problem over the 4-cube. A comprehensive exposition of the latter problem and its relatives can be found in Saks [12]. For recent developments see Klee [11] and Sohler-Ziegler [13]. For more on the algorithmic characterizations, generation and enumeration of cut-complexes for $n \leq 7$, see [4, 5]. In the second part of this exposition, we consider only the class of cut-complexes with 2 or 3 maximal faces for $n \geq 3$. Our main result provides simple necessary and sufficient conditions to recognize these cut-complexes.

Cut-complexes are geometric presentations of threshold Boolean functions that has been extensively studied in threshold logic. An analogous characterization for the positive threshold Boolean functions and purely by Boolean algebraic methods is given by Elgot [1]. See Hu [10], Winder [16], Muroga [15] and their references for this and more on threshold logic. Here, our proofs are independent of the polynomial expression of Boolean functions; instead,

[1] E-mail: `mreza@rrpac.upr.clu.edu`.

we have employed the geometric tools from convex polytopes. In fact, there is no trace of convex polytopes in the literature of threshold logic and thus our method of proofs extends the application domain of convex polytopes to threshold logic.

Considering first the easy case of only two maximal faces, we shall prove the following result in Theorem 1: To form a cut-complex by pasting two distinct proper faces of the n-cube, it is necessary and sufficient that they intersect in a facet of one of them. Then later, we extend the result to 3 maximal faces.

References

[1] Elgot, C. C.: Truth functions realizable by single threshold organs. IEEE Symp. On Foundations of Computer Science (1960), pp 225-245.

[2] Emamy-K., M. R.: On the cuts and cut-number of the 4-cube. J. Combin. Theory, Ser. A, vol. 41, 2 (1986), pp. 221-227.

[3] Emamy-K., M. R.: On the covering cuts of the d-cube, $d \leq 5$. Disc. Math., 68 (1998), pp. 191-196.

[4] Emamy-K., M. R.: On the cut-complexes of the 5-cube. Disc. Math., 78 (1989), pp. 221-227. (with L. Lazarte).

[5] Emamy-K., M. R. and Caiseda., C.: An efficient algorithm for characterization of cut-complexes. Congressus Numerantium 85 (1991), pp. 89-95.

[6] Emamy-K., M. R.: A new connection between convex geometry and threshold Math. Appl. Ser., 329, Combinatorics Advances, Kluwer Acad. Pub., C. J. Colbourn and E. Mahmoodian (eds.), (1995), pp. 121-128.

[7] Emamy-K., M. R.: Geometry of cut-complexes and threshold logic, J. Geom., 65 (1999), pp. 91-100.

[8] Grunbaum, B.: Convex Polytopes. Edited by Kaibel V., Klee V.,Ziegler., G. M. Springer-Verlag, 2003.

[9] Grunbaum, B.: Polytopal graph. MAA Studies in Math., vol. 12 (1975), pp. 201-224.

[10] Hu, S. T.: Threshold Logic. University of California Press, Berkeley, 1965, 332 pages.

[11] Klee, V.: Shapes of the future. Some unresolved problems in high-dimensional intuitive geometry, Proceedings of the 11th Canadian Conference on Computational Geometry(1999), pp. 17.

[12] Saks, M. E.: Slicing the hypercube. Surveys in Combinatorics, Cambridge University Press (1993), pp. 211-255.

[13] Sohler, C., Ziegler, M.: Computing Cut Numbers. 12th Annual Canadian Conference on Computational Geometry. CCCG 2000, pp 73-79.

[14] M C Mullen, P. and SHEPHARD., G. C.: Convex Polytopes and the Upper-Bound Conjecture. London Math. Soc. Lecture Note Ser., 3, 1971.

[15] Muroga, S.: Threshold Logic and its Applications. Wiley Interscience, Toronto, 1971.

[16] Winder, R. O.: Threshold Logic. Ph.D. Thesis, Mathematics Department, Princeton University, May 1962.

[17] Ziegler.,M.: http//www.uni-paderborn.de/cs/cubecuts

[18] Ziegler., G. M.: Lectures on Polytopes, Springer- Verlag,1994.

Covering Graphs by Colored Stable Sets

Ulrich Faigle[1], Bernhard Fuchs[2], Britta Wienand[3],

Zentrum für Angewandte Informatik (ZAIK), Universität zu Köln, Weyertal 80, D-50931 Köln, Germany.

Abstract

Let $G = (V, R \cup B)$ be a multigraph with red and blue edges. G is an R/B-split graph if V is the union of a red and a blue stable set. R/B-split graphs yield a common generalization of split graphs and König graphs. It is shown, for example, that R/B-split graphs can be recognized in polynomial time. On the other hand, finding a maximal R/B-subgraph is $\mathcal{NP}$-hard already for the class of comparability graphs of series-parallel orders. Moreover, there can be no approximation ratio better than 31/32 unless $\mathcal{P}$=$\mathcal{NP}$.

1 Introduction

In the present article, we consider multigraphs $G = (V, E)$ whose sets of edges consist of "red" and "blue" edges, say $E = R \cup B$. We are interested in covering the node set V of such a graph G by a red and a blue stable set to the best possible, *i.e.* we want to maximize the cardinality of the union of a red and a blue stable set, where a *red stable set* denotes a stable set in the *red graph* $G_R = (V, R)$ and a *blue stable set* refers to a stable set in the *blue graph* $G_B = (V, B)$.

If a graph $G = (V, R \cup B)$ has the property that the whole vertex set V can be covered by a red and a blue stable set we will call G an *R/B-split graph* as V can be split into a red and a blue stable set.

Section 2 shows that one can decide in polynomial time whether or not a given graph is an R/B-split graph. It turns out that the model of R/B-split graphs provides a natural common generalization of classical split graphs (see Földes and Hammer [2]) and graphs with the König Property (see Lovász and Plummer [3], p. 222). In our terminology, a (classical) *split graph* is a one-colored graph whose node set can be split into a stable set and a clique, while a graph with the *König Property* is a graph in which the size of a maximal matching equals the size of a minimal node cover.

[1] E-mail: `faigle@zpr.uni-koeln.de`.
[2] E-mail: `bfuchs@zpr.uni-koeln.de`.
[3] E-mail: `bwienand@zpr.uni-koeln.de`.

If G is not an R/B-split graph one might want to determine a maximal subgraph of G that is an R/B-split graph, *i.e.* one wishes to cover as many nodes as possible by the union of a red and a blue stable set. This optimization problem is easily seen to be $\mathcal{NP}$-hard in general as it already reduces to the original MAX STABLE SET PROBLEM if $R = \emptyset$ or $B = \emptyset$. Section 3 discusses some polynomially solvable instances of the problem. If, for example, $G_R = G_B$ is the comparability graph of a partial order, the problem amounts to determining a maximal subset that can be expressed as the union of two antichains in the partial order. This maximization problem is well-known to be solvable in polynomial time even for the union of k antichains (see, *e.g.*, Frank [1]).

Interestingly, the problem of determining a maximal union of a red and a blue antichain turns out to be $\mathcal{NP}$-hard already for the class of series-parallel orders.

2 Red/Blue-Split Graphs

Given a graph $G = (V, R \cup B)$ with sets R and B of red, resp. blue, edges, we want to decide whether there exists a partition $V = S_R \cup S_B$ of V into a stable set S_R in the red graph $G_R = (V, R)$ and a stable set S_B in the blue graph $G_B = (V, B)$. Note that G is a multigraph as two nodes may be linked by a red and by a blue edge. We call this decision problem the R/B-SPLIT PROBLEM and show that it is efficiently solvable as it is in some sense equivalent to the 2–SATISFIABILITY PROBLEM:

Theorem 1 *The* R/B-SPLIT PROBLEM *can be efficiently reduced to a* 2–SATISFIABILITY PROBLEM *and vice versa.*

One may wonder if the generalized R/B/G-SPLIT PROBLEM of splitting a graph with red, blue and green edges into stable sets is also polynomial. We note that the generalized R/B/G-SPLIT PROBLEM is $\mathcal{NP}$-complete, as it reduces to the well-known $\mathcal{NP}$-complete 3-COLORING PROBLEM for the special case where $G_R = G_B = G_G$.

We now turn to the discussion of some special instances of the R/B-SPLIT PROBLEM:

2.1 Split graphs and König graphs

If the red graph G_R of $G = (V, R \cup B)$ is the complement of the blue graph G_B, we find that G is an R/B-split graph if and only if the blue graph G_B itself can be split into a clique and a stable set (as each clique is a stable set in the complement graph and vice versa). Therefore G is an R/B-split graph if and only G_B is a *split graph.*

A graph G is said to have the *König Property* [3] if the size of a maximum matching equals the size of a minimum node cover in G. If we determine a maximum matching M of G and the *Gallai-Edmonds decomposition* of V into the set $D(G)$ of all nodes not covered by at least one maximum matching, the set $A(G)$ of all neighbors of $D(G)$ in $V \setminus D(G)$ and the set $C(G)$ of the remaining nodes (which can be done efficiently, for example, with Edmonds' cardinality matching algorithm [3]) we prove that we can decide whether G has the König Property by solving an R/B-split problem:

Theorem 2 *Let M be any maximum matching in $G = (V, E)$. Then G has the König Property if and only if $D(G)$ is stable in G and the 2-colored graph $G' = (V, R \cup B)$, where $G'_R = G$ and $G'_B = (V, M)$, is an R/B-split graph.*

2.2 Stable Matroid Bases

Let $\mathcal{B} \subseteq 2^V$ denote the nonempty family of bases of a *matroid* $M = (V, \mathcal{B})$ on the set V of nodes of a graph $G = (V, E)$. We are interested in the question whether M admits a *stable matroid basis*, *i.e.* a basis of $M = (V, \mathcal{B})$ that is a stable set in G as well. This Stable Basis Problem is easily seen to be $\mathcal{NP}$-hard in general, but polynomially solvable for special instances.

For example for the case where $G = (V, E)$ is any graph but M the dual of a partition matroid M^* (A *partition matroid* is a matroid whose ground set V is partitioned into sets $V = A_1 \cup A_2 \cup ... \cup A_m$ and the bases are the subsets of V that contain exactly one element of each of the sets A_i) we prove:

Theorem 3 *If the matroid M is the dual of a partition matroid the* Stable Basis Problem *can be efficiently reduced to an R/B-*Split Problem*.*

The example of Theorem 3 shows that the Stable Basis Problem becomes polynomial even for general graphs if we restrict the problem to a special class of matroids. The following example exhibits the problem to become polynomial for general matroids M when we restrict ourselves to the special class of graphs G where G is the cocomparability graph of a *tree-order* P, *i.e.* a partial order P whose Hasse diagram forms a rooted tree. (Note that a stable set in the cocomparability graph of a partial order is simply a chain in that order.)

Theorem 4 *If P is a tree-order, then a maximal independent chain can be calculated in polynomial time.*

The general problem of a maximal independent chain (or antichain) turns out to be $\mathcal{NP}$-complete even for a partition matroid and series-parallel orders as we prove

Theorem 5 *The* Stable Basis Problem *is $\mathcal{NP}$-complete for a partition matroid and the comparability graph of a series-parallel order.*

3 Maximal Covers by Stable Sets

If $G = (V, R \cup B)$ is not an R/B-split graph one might ask for the largest subset of V such that the induced subgraph is an R/B-split graph. We refer to this problem as the Max R/B-Split Problem. The general Max R/B-Split Problem is $\mathcal{NP}$-hard as it includes the MAX STABLE SET PROBLEM. Therefore, it would be interesting to identify polynomially solvable cases of the MAX R/B-SPLIT PROBLEM.

Example 1 *If G_R is the complement of a chordal graph $\bar{G}_R$ (*i.e. *every cycle in $\bar{G}_R$ of length at least 4 possesses a chord) and G_B is a comparability graph the* MAX R/B-SPLIT PROBLEM *is polynomial.*

If the red graph and the blue graph are identical, the MAX R/B-SPLIT PROBLEM becomes the problem to determine a maximal union of two stable sets. This problem is still $\mathcal{NP}$-hard for general graphs as it is a special instance of the known $\mathcal{NP}$-hard MAXIMUM INDUCED SUBGRAPH WITH PROPERTY Π PROBLEM But again, there are graphs for which this problem is solvable:

Example 2 *If G_R and G_B are comparability graphs and $G_R = G_B$ the* MAX R/B-SPLIT PROBLEM *is polynomial since there exist efficient algorithms for the maximal union of two antichains relative to the same partial order (see* e.g. *[1]).*

The last example raises the question whether the MAX R/B-SPLIT PROBLEM is generally polynomial in case G_R and G_B are comparability graphs. We show that this problem is already $\mathcal{NP}$-hard for comparability graphs of series-parallel orders:

Theorem 6 *Given two partial orders $P_R = (V, \leq_R)$ and $P_B = (V, \leq_B)$ on the same ground set V, it is $\mathcal{NP}$-hard to decide whether there exist maximal antichains A_R in P_R and A_B in P_B with $A_R \cap A_B = \varnothing$.*

The same construction also produces a reduction from MAX 3-SAT, which cannot be approximated better that $\frac{7}{8}$ (or $\mathcal{P}= \mathcal{NP}$) [4]. This directly gives some insight into the approximability of the MAX R/B-SPLIT PROBLEM.

Corollary 1 *For $\varepsilon > 0$, there cannot exist a $(\frac{31}{32} + \varepsilon)$-approximative algorithm for* MAX R/B-SPLIT*, unless $\mathcal{P}= \mathcal{NP}$.*

We finally remark that taking a maximal red and a maximal blue antichain yields a simple 2-approximation algorithm, which altogether places MAX R/B-SPLIT into the class of so-called APX-complete problems.

The proof of Theorem 5 implies

Corollary 2 *For $\varepsilon > 0$, there cannot exist a $(\frac{7}{8} + \varepsilon)$-approximative algorithm for finding a longest independent (anti-)chain, unless $\mathcal{P}= \mathcal{NP}$.*

4 Acknowledgment

The authors are grateful to Nguyen Tien Thanh for pointing out the relationship between R/B-split graphs and König graphs.

References

[1] A. Frank, *On Chain and Antichain Families of a Partially Ordered Set*, J. Combinatorial Theory Ser. B **29**, 1980, pp 176-184.

[2] S. Földes, P.L. Hammer, *Split Graphs*, Proc. 8th Southeastern Conf. on Combinatorics, Graph Theory and Computing (F. Hoffmann et al., eds.), Lousiana State Univ., Baton Rouge, Louisiana, 1977, pp 311-315.

[3] L. Lovász, M.D. Plummer, *Matching Theory*, Annals of Discrete Mathematics **29**, North-Holland, 1986, p 224.

[4] J. Håstad, *Some optimal Inapproximability Results*, Journal of ACM, Vol. 48, 2001, pp 798–859.

About the b-continuity of graphs

Taoufik Faik [1]

U.M.R. 86-23 L.R.I., Université Paris-Sud Bât. 490-91405 Orsay Cedex, France.

1 Introduction

In this paper we use the graph theory definitions and notations from [1]. A b-coloring of a graph G is a proper coloring π of the vertices of G such that for each color c, there exists a vertex v with $\pi(v) = c$ such that for any color $c' \neq c$, there exists a neighbor v' of v with $\pi(v') = c'$. Such a vertex v is called a *b-chromatic vertex for color c*. We denote $|\pi| = |\pi(V(G))|$ the *cardinalities* of the coloring. If $|\pi| = k$, then π is called a *(k)b-coloring.* Given a graph G, the b-chromatic number $b(G)$ is the greatest integer k such that there exists a (k)b-coloring of G. Given an integer k, to know whether $b(G) \geq k$ is a NP-complete problem [2], even if G is bipartite [3]. This problem has been shown to be in P for trees, and some lower bounds of the b-chromatic number were given for the cartesian product of two graphs [2, 4].
It is easy to see that any proper coloring with $\chi(G)$ colors (the chromatic number) is a b-coloring. One peculiar characteristic of b-colorings is that for some graphs G, there exist some integers k, $\chi(G) < k < b(G)$, for which there is no (k)b-coloring of G (see for example the hypercube $H(3)$ with $k = 3$ [2]). Thus, we will say that a graph G is *b-continuous* if and only if for any k, $\chi(G) \leq k \leq b(G)$, there exists a (k)b-coloring of G. Using the terminology of Harary [5], we say that $b(G)$ is an *interpolating invariant*, when G is b-continuous. We focus here on the b-continuity of graphs. In [6] they proved that trees are b-continuous and showed that apart from two exceptions 3-regular graphs are b-continuous. In [7] they showed the b-continuity of interval graphs. Note that the b-chromatic number problem is still open for interval graphs. The *b-continuity* problem consisting in deciding if a given graph G is b-continuous was shown to be NP-complete [7], even if a $(\chi(G)$b-coloring and $(b(G)$b-coloring are easy to compute. This shows that knowing a minimal (resp. maximal) b-coloring of G would not help to decide if a graph G is b-continuous. Recently, we have shown that this problem remains NP-complete even for bipartite graphs.

Our paper is organized as follows. In the next section we prove the b-continuity of a particular class of graphs called chordal graphs, this generalizes the b-continuity of trees and interval graphs, indeed trees and interval graphs are subclasses of chordal graphs. Then we give an upper bound of the b-chromatic number and a family of graphs for which the chromatic and the b-chromatic number are equals.

[1] E-mail: `Taoufik.Faik@lri.fr`.

2 The b-continuity of chordal graphs

An undirected graph $G(V, E)$ is *chordal* if every cycle of length at least four has a *chord*, i.e. an edge between two nonconsecutive vertices of the cycle. A vertex v of a graph G is called *simplicial* if its closed neighborhood $\Gamma[v]$ induces a clique. An *elimination ordering* σ of a graph G of order n is a bijection $\sigma : \{1, 2, \ldots, n\} \to V$. Accordingly, $\sigma(i) = v_i$ is the *ith* vertex in the elimination ordering and $\sigma^{-1}(v)$, $v \in V$ gives the position of v in σ. A *perfect elimination ordering* (peo) is an elimination ordering $\sigma = (v_1, v_2, \ldots, v_n)$ such that for any i, $(1 \leq i \leq n)$, v_i is a simplicial vertex in the subgraph induced by $\{v_i, v_{i+1}, \ldots, v_n\}$. It is well known that an undirected graph is chordal if and only if it has a perfect elimination ordering (Fulkerson and Gross 1965, Golumbic 1980).

Given a peo σ of a chordal graph G, $\Gamma^+[v, \sigma]$ denotes the set of vertices in $\Gamma[v]$ with position not smaller than $\sigma^{-1}(v)$. That is, $\Gamma^+[v, \sigma] = \{u \in \Gamma[v] : \sigma^{-1}(u) \geq \sigma^{-1}(v)\}$. A path $C = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$ is called an *increasing path* with respect to σ if $i_1 < i_2 < \ldots < i_k$ and for every j, $1 \leq j \leq k-1$, v_{i_j} is adjacent to $v_{i_{j+1}}$.

Lemma 1 *Let $C = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$ an increasing path. Let v_r be a vertex not in C but adjacent to a vertex v_{i_j} of C.*
(1) If $r > i_k$, then v_r is adjacent to v_{i_h} for each h, $j \leq h \leq k$.
(2) If $r < i_k$, then there exists an increasing path from v_r to v_{i_k}.

Proof : (1). For each h, $j \leq h \leq k$, the set $\Gamma^+[v_{i_h}, \sigma]$ is a clique. By hypothesis $v_r \in \Gamma^+[v_{i_j}, \sigma]$, Since C is an increasing path $v_{i_{j+1}} \in \Gamma^+[v_{i_j}, \sigma]$. Then $v_{i_{j+1}}$ is adjacent to v_r. By the same argument we can prove that, for every h, $j+1 \leq h \leq k$, v_h is adjacent to v_r.

(2). If $r < i_j$, then $C' = (v_r, v_{i_j}, \ldots, v_{i_k})$ is an increasing path from v_r to v_{i_k}. Otherwise, $(i_j < r < i_k)$, let h, $j \leq h \leq k$ be the greatest integer such that $r > i_h$. As $C' = (v_{i_j}, \ldots, v_{i_h})$ is an increasing path, by (1), v_r is adjacent to v_{i_h}. Since $\Gamma^+[v_{i_h}, \sigma]$ is a clique and $v_r, i_{h+1} \in \Gamma^+[v_{i_h}, \sigma]$, v_r is adjacent to $v_{i_{h+1}}$. Thus $(v_r, v_{i_{h+1}}, \ldots, v_{i_k})$ is an increasing path from v_r to v_{i_k}. ∎

Theorem 1 *Chordal graphs are b-continuous.*

Proof : Let G be a chordal graph and π a b-coloring of G using p colors, $p > \chi(G)$. We give an algorithm which reduces π to a b-coloring π' using only $p-1$ colors. Let σ be an arbitrary peo of G. Let $\mu(\pi)$ be the smallest integer such that $v_{\mu(\pi)}$ is b-chromatic. Set $\mu(\pi) = k$. Note that $\mu(\pi) > 1$ since the set $\Gamma^+[v_k, \sigma]$ is a clique, $|\Gamma^+[v_k, \sigma]| \leq \chi(G) < p$. Therefore, the set $\mathcal{P} = \{1, \ldots, p\} \setminus \pi(\Gamma^+[v_k, \sigma])$ is nonempty. Let $j \in \mathcal{P}$, and let $\mathcal{J} = \{v_i : 1 \leq i \leq k-1, \pi(v_i) = j$ *and there exists an increasing path from* v_i *to* $v_k\}$. Consider $S = \{v_i : 1 \leq i \leq k-1, \pi(v_i) = j$ *and* v_i *is adjacent to* $v_k\}$. As v_k is b-chromatic and $j \notin \pi(\Gamma^+[v_k, \sigma])$, $S \neq \emptyset$. It is clear that $S \subset \mathcal{J}$, so $\mathcal{J} \neq \emptyset$.

Claim 1 *If a vertex v_r, $1 \leq r \leq k-1$, has a neighbor in $\mathcal{J}$, then all its neighbors of color j are in $\mathcal{J}$.*

Let v_i be a neighbor of v_r in $\mathcal{J}$. As $v_i \in \mathcal{J}$, there exists an increasing path from v_i to v_k. Moreover, v_r is adjacent to v_i and $r < k$. By Property (2) of Lemma 1 there exists an increasing path from v_r to v_k. Assume that there exists another neighbor v_h of v_r such that $\pi(v_h) = j$. We show that $v_h \in \mathcal{J}$.

First, we prove that $h < k$. Assume that $h \geq k$. There exists an increasing path from v_r to v_k and v_r is adjacent to v_h. By Property (1) of Lemma 1 v_h is adjacent to v_k, a contradiction with the fact that $\pi(v_h) = j$ and $j \notin \pi(\Gamma^+[v_k, \sigma])$.

Hence, as $h < k$, v_r is adjacent to v_h and there exists an increasing path from v_r to v_k, by Property (1) of Lemma 1, there exists an increasing path from v_h to v_k. This concludes the proof of the claim.

Recolor each vertex in $\mathcal{J}$ by a color taken in the set $\pi(V) \setminus \{j\}$, in such a way that the recoloring remains a proper one. This recoloring is possible since there is no b-chromatic vertex in $\mathcal{J}$. Denote π' the resulting coloring. Consider any vertex in the set $\{v_1, v_2, \ldots, v_{k-1}\}$ having a neighbor in $\mathcal{J}$ before the recoloring. By Claim 1, all its neighbors of color j were in $\mathcal{J}$, so it has no more neighbor of color j after the recoloring and so cannot be b-chromatic. Furthermore at least the vertex v_k is no more b-chromatic, hence $\mu(\pi') > \mu(\pi)$. Since the value of μ cannot increase indefinitely, it turns out that after a finite number of iterations of this recoloring process, at least one color loses all its b-chromatic vertices. Denote by $\mathcal{H}$ the set of vertices which were the last b-chromatic ones of their colors before the last iteration of the recoloring process, and k the last value of $\mu(\pi)$. The vertices of the set $\mathcal{H}$ are no longer b-chromatic because their neighbors of color j were all in the set $\mathcal{J}$. Applying Property (1) of Lemma 1, we obtain $\mathcal{H} \subset \Gamma^+[v_k, \sigma]$. So $\mathcal{H}$ is a clique. Hence, we lose at most one b-chromatic vertex of each color. Let l be a color which lost all its b-chromatic vertices after the last iteration of the recoloring process and v_r, $r \geq k$ be the last b-chromatic vertex of the color l before this last iteration. Note that v_r may happen to be distinct from v_k.

Now recoloring v_r with color j makes all the vertices of $\mathcal{H}$, apart from v_r, candidates for b-chromaticity in a coloring of G without color l. Since v_r was the last b-chromatic vertex of the color l, we may recolor each remaining vertex of color l with another color, obtaining by this way a $(p-1)$b-coloring π' of G.■

3 Modified degree sequence and $k - partite$ graphs

In this section we give an upper bound of the b-chromatic number and a family of graphs for which the chromatic number and the b-chromatic number are equals. A graph $G = (V, E)$ is $k - partite$ if there is a partition of V in k nonempty stable sets $X_i, 1 \leq i \leq k$. Given such a partition, the sets X_i are called *classes* of the partition. In the case $k = 2$ we talk about of *bipartite*-graphs. Let G be a graph with n vertices, and $(x_1, x_2, \ldots, x_n)$ an ordering of V giving a nonincreasing sequence of degrees *(* i.e. if d_i is the degree of x_i, we have $d_1 \geq d_2 \geq \ldots \geq d_n$*)*. If we delete in this ordering every vertex x_i such that there exists $j < i$ with $\Gamma(x_i) \subset \Gamma(x_j)$, the nonincreasing sequence of remaining degrees is called the *modified degree sequence* of G.

Note that the set $(x_{i_1}, \dots, x_{i_k})$ obtained after deletion of subordinate vertices may depend on the initial ordering, but the sequence of degrees $d'_1 \geq \cdots \geq d'_k$ in which d'_j is the degree of x_{i_j}, is the same for any choice of the initial ordering. We derive from this modified degree sequence, a parameter giving a bound for $b(G)$ improving the known bound $m(G)$ (see [2]. Let $d'_1 \geq \cdots \geq d'_k$ be the modified degree sequence of a graph G. Then the parameter $m'(G)$ is defined as follows : $m'(G) = \max\{j | d'_j \geq j-1\}$.

Lemma 2 *Let $G = (V, E)$ be a graph, x, y be two vertices with $\Gamma(y) \subseteq \Gamma(x)$, and set $G' = G \setminus \{y\}$. Then we have : (a) If G' has a b-coloration with k colors, so has G.*
(b) $b(G') \leq b(G)$.
(c) In any b-coloration π of G for which y is b-chromatic, x also is, and $\pi(x) = \pi(y)$.

Proof : : For any b-coloration of G' we may assign to y in G the same color as x, and obtain in such a way a b-coloration of G with the same set of colors, so part (a) is true. Part (b) is an obvious consequence of part (a). Moreover, if y is b-chromatic for a b-coloration $\pi : G \to C$, then $\pi(\Gamma(y)) = C \setminus \{\pi(y)\}$. Since $\Gamma(x) \supseteq \Gamma(y)$, the color of x must be equal to $\pi(y)$ and x is obviously b-chromatic. □

Proposition 1 *$b(G) \leq m'(G)$.*

Proof : : Suppose that G has a b-coloration with p colors. Then, from Lemma 1, any set of vertices $\{x_{i_1}, \dots, x_{i_k}\}$ giving, as explained above, the modified degree sequence, contains a b-chromatic vertex for each color. Since a b-chromatic vertex has degree at least $p-1$, and the modified sequence of degrees is nonincreasing, we must have $d'_1 \geq p-1, \dots, d'_p \geq p-1$ implying by definition $p \leq m'(G)$. □ A vertex in a class X of a k-partite graph $G = (V, E)$ is called *charismatic* if its neighborhood is $V \setminus X$.

Theorem 2 *(a) If each class of a k-partite graph G contains at least one charismatic vertex, then $\chi(G) = b(G) = k$. (b) If G is a bipartite graph, ($k = 2$), then $\chi(G) = b(G) = 2$ iff each class of the bipartion contains at least one charismatic vertex*

Proof : : (a) It is easy to see that $\chi(G) = m'(G) = k$ and the coloration which assigns to class X_i the color c_i is b-chromatic. (b) If $k = 2$, the existence of a charismatic vertex in each class of the bipartion is an equivalent characterization to that given in [3]. □ The part (b) of the previous theorem is an equivalent characterization to that given in [3]. This characterization has no obvious generalization for graphs with $\chi(G) \geq 3$. For instance, the graph G with set of vertices $\{x, y, z, t\}$ and edges $\{(xy), (yz), (zx), (zt)\}$ has both chromatic and b-chromatic numbers equal to 3, but one class of any 3-partition of G lacks of charismatic vertex.

Acknowledgment. I wish to express my thanks to Professor Jean-François Saclé whose kind help made this paper possible.

References

[1] C. Berge. Graphs. North-Holland,1985.

[2] R. W. Irving and D. F, Manlove. The b-chromatic number of a graph. Discrete Applied Mathematics, 91 : 127-141,1999.

[3] J. Kratochvil, Z. Tuza, M. Voigt. On the b-chromatic number of a graphs, WG 2002, LNCS 2573, 2002 310-320.

[4] M. Kouider and M. Mahéo. Some bounds for the b-chromatic number of a graph. Discrete Mathematics, 256 : 267-277, 2002.

[5] F. Harary, S. Hedetniemi, G. Prins. An interpolation theorem for graphical homomorphisms, Portugal. Math 26 (1967) 453-462

[6] T. Faik and J.-F. Saclé. Some b-continuous classes of graph, Technical Report N. 1350, LRI, Universite de Paris Sud,(2003).(submitted)

[7] D. Barth, J. Cohen, T. Faik. Complexity of determining the b-continuity property of graphs, PR*i*SM Technical Report,(2003). (submitted) .

A 3-approximation for the pathwidth of Halin graphs

Fedor V. Fomin [1]

Department of Informatics, University of Bergen N-5020, Bergen, Norway

Dimitrios M. Thilikos [2]

Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, E-08034, Barcelona, Spain

Abstract

We prove that the pathwidth of Halin graphs can be approximated within a factor of 3.

All graphs in this paper are finite without loops or multiple edges. For a graph G we denote by $A(G)$ the set of degree one vertices of G. A *plane graph* is a particular drawing of a planar graph in the plane without crossings. A plane graph G is a *Halin graph* if it can be constructed by taking a plane tree T and connecting its leaves with edges so that all of them obtain degree 3 and are incident to the external face of the resulting plane graph. In other words, the edge set $E(G)$ can be partitioned as $E(T) \cup E(C)$, $E(T) \cap E(C) = \emptyset$, where T is a tree and C is a cycle on only and all of the pendant vertices of T. The tree T is called *skeleton* of G. Halin graphs were introduced by Halin in [6]. As proved in [1] any Halin graph has treewidth ≤ 3. Finding a polynomial algorithm for computing (or approximating) the pathwidth of graphs of treewidth bounded by some fixed constant k is an old problem mentioned first by Dean in [4]. A general answer to this problem was given by Bodlaender and Kloks in [3]. However, while the algorithm in [3] is polynomial, its exponent is heavily depending on k and this makes it impractical even for small values of k (already one step in this algorithm requires to work with sets of size $O(n^{11})$). Therefore, it is interesting to find a low-degree polynomial time algorithm for graphs of treewidth bounded by small values of k. In this direction, the only case where there exists an exact algorithm for pathwidth is the case $k = 1$, i.e. when the input graph is a forest. For this, $O(n \log n)$-time algorithms where given in [7, 5] and this time was recently improved by the linear time algorithm of Skodinis in [9]. Unfortunately, so far, no fast explicit algorithm is known for $k = 2$. The best progress in this direction is the 2-approximation algorithm of [2] for outerplanar graphs, a class of graphs with treewidth ≤ 2. So far, no fast exact or approximation algorithm is

[1] E-mail: `fomin@ii.uib.no`.

[2] E-mail: `sedthilk@lsi.upc.es`. The second author was supported by the EU within the 6th Framework Programme under contract 001907 (DELIS) and by the Spanish CICYT project TIC-2002-04498-C05-03 (TRACER).

known for any class of graphs with treewidth 3 or more. In this paper we give a linear time algorithm that approximates the pathwidth of Halin graphs within a factor of 3.

The notion of pathwidth was introduced by Robertson and Seymour [8]. A *path decomposition* of a graph G is a sequence $(X_1, X_2, \dots, X_r)$ of subsets of $V(G)$ (these subsets are called *bags*), such that **(1)** $\bigcup_{1 \le i \le r} X_i = V(G)$, **(2)** for all $\{v, w\} \in E(G)$, there is an i, $1 \le i \le r$, with $v, w \in X_i$, and **(3)** for all $1 \le i_0 < i_1 < i_2 \le r$, $X_{i_0} \cap X_{i_2} \subseteq X_{i_1}$. The *width* of path decomposition $(X_1, X_2, \dots, X_r)$ is $\max_{1 \le i \le r} |X_i| - 1$. The *pathwidth* of a graph is the minimum width over its path decompositions.

For $S \subseteq V(G)$ we define $\partial S = \{u \in S$ and there exists $w \in V(G) \setminus S$ such that $\{u, w\} \in E(G)\}$. Let $\sigma = (v_1, v_2, \dots, v_n)$ be an ordering of $V(G)$. For $j \in \{1, \dots, n\}$ we put $V_j = \cup_{i=1}^{j} v_i$. Setting $\mathrm{vs}(G, \sigma) = \max_{i \in \{1, \dots, n\}} |\partial V_i|$, we define the *vertex separation* of G as $\mathrm{vs}(G) = \min\{\mathrm{vs}(G, \sigma)\colon \sigma$ is an ordering of $V(G)\}$. It is well known [5] that for any graph G, $\mathrm{vs}(G) = \mathbf{pw}(G)$.

The linear width was introduced by Thomas [11]. For $X \subseteq E(G)$ let $\delta(X)$ be the set of all vertices incident to edges in X and $E(G) \setminus X$. Let $\sigma = (e_1, e_2, \dots, e_m)$ be an ordering of $E(G)$. For $i \in \{1, \dots, m\}$ we put $E_i = \cup_{j=1}^{i} e_j$. We define $\mathbf{lw}(G, \sigma) = \max_{i \in \{1, \dots, m\}} |\delta(E_i)|$, and the *linear width* of G as $\mathbf{lw}(G) = \min\{\mathbf{lw}(G, \sigma)\colon \sigma$ is an ordering of $E(G)\}$. We can prove that, for any graph G, $\mathbf{pw}(G) \le \mathbf{lw}(G) \le \mathbf{pw}(G) + 1$ (proof omitted).

Let T be a tree and let P be a path of T. We define $\mathcal{T}(T, P)$ as the collection of trees defined by the connected components of the graph taken after subdividing in T all edges not in P but with endpoints in P and then removing all the vertices in P. For reasons of simplicity, we will use the same notation as in T for all the vertices of the trees in $\mathcal{T}(T, P)$. We omit the proof of the following lemma due to space restrictions (the proofs based on results from [10].

Lemma 1 *Any tree T with* $\mathrm{ms}(G) \le k$ *contains a path (*spine path*) P where all the trees in $\mathcal{T}(T, P)$ have linear width at most $k - 1$.*

Let G be a graph embedded in the plane. For any vertex v of G, we define as $C_v = (e_1, \dots, e_\sigma, e_1)$ the clockwise cyclic order of the edges incident to v according to the way they are embedded around v in the plane. We say that two edges are *friends* if they share an endpoint and belong in the border of the same face of the embedding. If $L = (e_1, \dots, e_q)$ is an ordering of $E(G)$ and $v \in V(G)$ we define $L_v = L \cap \{\{v, u\} \mid u \in N_G(v)\}$ (i.e. L_v is the restriction of L to the edges incident to v in G). We say that $L_v \sim C_v$ if for any $j, 1 \le j \le \sigma$ the j first edges in L_v form an interval of C_v (or, in other words, when any two consecutive edges in L_v are friends). We also say that L *respects* the plane embedding of G if for any $v \in V(G)$, $L_v \sim C_v$. When the plane embedding is clear from the context we will simply call L *respectful.*

Lemma 2 *For any plane tree T with linear-width at most k, there exists a respectful edge ordering L of T that has linear-width at most k.*

Proof : We will apply induction on k. We assume that the result holds for any $i < k$ and we will prove that it also holds for k. Let $P = (v_1, \dots v_r)$ be the spine path of T given by Lemma 1 . For any $i, 1 < i < r$, we denote as $L_i = (e_i^1, \ldots, e_i^{\rho_i})$ the edges incident to v_i ordered in a way that $L_i \sim C_{v_i}$, $e_i^1 = \{v_{i-1}, v_i\}$ and $e_i^{\rho_i} = \{v_i, v_{i+1}\}$. We also define $L_1 = (e_1^1, \ldots, e_1^{\rho_1})$ and $L_r = (e_r^1, \ldots, e_r^{\rho_r})$ such that $L_1 \sim C_{v_1}$, $L_r \sim C_{v_r}$, $e_1^{\rho_1} = \{v_1, v_2\}$ and $e_r^1 = \{v_{r-1}, v_r\}$. Let $i, 1 \leq i \leq r$ and $j, 1 \leq j \leq \rho_i$. We denote T_i^j the connected component of $\mathcal{T}(T, v_i)$ that contains e_i^j as an edge.

Let $I = \{(i,j) \mid 1 \leq i \leq r, 1 \leq j \leq \rho_i \text{ and } e_i^j \text{ is not an edge of } P\}$ and notice that for each $(i,j) \in I$, e_i^j belongs to a different tree of $T_i^j \in \mathcal{T}(T, P)$. From Lemma 1, any such a T_i^j has linear-width$\leq k-1$ and from the induction hypothesis it has a respectful edge ordering L_i^j of linear-width$\leq k-1$. We now define $L = L_1^1 \oplus \cdots \oplus L_1^{\rho_1 - 1} \oplus \{v_1, v_2\} \oplus L_2^2 \oplus \cdots \oplus L_2^{\rho_2 - 1} \oplus \{v_2, v_3\} \oplus \cdots \oplus \{v_{r-1}, v_r\} \oplus L_r^2 \oplus \cdots \oplus L_r^{\rho_r}$ and we observe that L is a continuous respectful edge ordering of G with linear-width$\leq k$ (notice that the L_i^j's that are omitted in this concatenation are exactly those that correspond to pairs (i,j) missing from I).

Lemma 3 *Any plane tree T is the minor of a plane tree T' of maximum degree 3, where* $\mathbf{lw}(T) = \mathbf{lw}(T')$ *and* $A(T) = A(T')$.

Proof : Let L be an edge ordering of T with minimum linear-width. From Lemma 2 there exists a respectful ordering L of T with the same linear-width. We apply the following algorithm on T and observe that it stops only when T becomes a tree of maximum degree 3, as required.

(1) If T contains a vertex v of degree ≥ 4 then goto the next step, otherwise output T and stop. **(2)** Let e_i and e_j be the two first edges in E_v that appear in L. We construct a new tree T' as follows: first construct the tree U_1 by taking by the union of the two trees of $\mathcal{T}(T, v)$ connecting the edges e_i and e_J, then construct the tree U_2 by taking the union of the rest of the trees of $\mathcal{T}(T, v)$, then rename to v^1 the vertex v in $U_i, i = 1, 2$ and finally define T' as the disjoint union of U_1 and U_2 with $e_{\text{new}} = \{v^1, v^2\}$ as an additional edge. Notice that $L' = (e_1, \ldots, e_i, \ldots, e_j, e_{\text{new}}, e_{j+1}, \ldots, e_q)$ is a respectful edge ordering for T' with linear-width$\leq k$. Therefore $\mathbf{lw}(T') \leq \mathbf{lw}(T)$. As T is a minor of T', we have that $\mathbf{lw}(T) \leq \mathbf{lw}(T')$ and thus T' has the same linear-width as T. Moreover, as the vertex splitting operation of this step is not applied to a leave, we get hat $A(T) = A(T')$ **(3)** Set $T := T', L := L'$ and goto Step **1**.

Lemma 4 *Let H be a Halin graph with skeleton T. Then* $\mathbf{lw}(T) \leq \mathbf{lw}(H) \leq 3 \cdot \mathbf{lw}(T)$.

Proof : The first inequality is obvious as T is a subgraph of H. In what remains we will prove that $\mathbf{lw}(H) \leq 3 \cdot \mathbf{lw}(T)$. From Lemma 3, T is the minor of some simple tree T' where $\mathbf{lw}(T') = \mathbf{lw}(T)$ and $A(T) = A(T')$. Let H' be the Halin graph having T' as a skeleton. Because $A(T) = A(T')$, we have that H is a minor of H' and therefore $\mathbf{lw}(H) \leq \mathbf{lw}(H')$. Clearly, the lemma follows if we prove that $\mathbf{lw}(H') \leq 3 \cdot \mathbf{lw}(T')$.

We construct a new graph J by modifying the graph H'. For every internal face F of H' (i.e. a face that is not the external face of H') we do the following. Let $(v_1, \ldots, v_r, v_1)$ be the cycle bordering F, where $r \geq 3$ and $\{v_1, v_r\}$ is the edge of $E(H') - E(T')$. We replace the edge $\{v_1, v_r\}$ by a path $(v_1 = a_1, a_2, \ldots, a_r = v_r)$ of length $r-1$. There is a natural one-to-one correspondence between edges of paths $(v_1, \ldots, v_r)$ and $(a_1, a_2, \ldots, a_r)$. For every $i \in \{1, \ldots, r-1\}$, we call the edge $\{a_i, a_{i+1}\}$ by the *shadow* of the edge $\{v_i, v_{i+1}\}$ and vertex a_i by shadow of v_i. Since every edge $e \in E(T')$ is adjacent to two faces, it has two shadows in

J. Because every vertex of T' is of degree ≤ 3, it has at most 3 shadows. For $e \in E(T')$ and $v \in V(T')$, let $S(e)$ and $S(v)$ be the sets of shadows of e and v. Notice that J is Halin graph with skeleton T'. Also J contains H' as a minor and therefore $\mathbf{lw}(H') \leq \mathbf{lw}(J)$. Therefore, it is enough to prove that $\mathbf{lw}(J) \leq 3 \cdot \mathbf{lw}(T')$.

Let $E \subset E(T')$. For every vertex $v \in \bigcup_{e\in E} \delta_J(S(e) \cup \{e\})$ either $v \in \delta_{T'}(E)$, or v is a shadow ($v \in S(u)$) of some $u \in \delta_{T'}(E)$. Moreover, since vertex degree of T' is at most 3, for every vertex $v \in \delta_{T'}(E)$, at most 2 of its shadows are in $\bigcup_{e\in E} \delta_J(S(e) \cup \{e\})$. Therefore, $|S(v) \cap \bigcup_{e\in E} \delta_J(S(e))| \leq 2$. Thus for any edge subset, $E \subset E(T')$ $|\delta_{T'}(E)| \leq 3|\bigcup_{e\in E} \delta_J(S(e) \cup \{e\})|$ (*). Let now $L = (e_1, \dots, e_q)$ be an edge ordering of T' of linear-width $\leq k$. This ordering induces an 'ordering of shadows' L_J in J, i.e. for $i < j$ edge e_i and its shadows are in L_J before e_j and its shadows. By (*), for any $\ell \in \{1, \dots, q\}$, $|\bigcup_{i=1}^{\ell} \delta_J(S(e_i) \cup e_i)| \leq 3|\bigcup_{i=1}^{\ell} \delta_{T'}(e_i)|$.

To finish the proof we need to show how to order shadows. Each edge e of T' has two shadows in $E(J) - E(T')$. We call these two shadows e^1 and e^2 always choosing indices 1 or 2 arbitrarily except the following case: For $i = 1, 2$, each endpoint of $e = \{v, v'\}$ belongs to exactly one edge e_i appearing before e in L and each such edge has a shadow sharing an endpoint with a shadow e^* of e. In this case we set $e^1 = e^*$ and let e^2 be the other shadow of e It is now easy to observe that $L^* = (e_1^1, e_1, e_1^2, \dots, e_q^1, e_q, e_q^2)$ is an edge layout of J with linear-width $\leq 3k$ as required.

Theorem 1 *There exists an linear time algorithm that for any Halin graph H returns an integer δ such that $\delta - 1 \leq \mathbf{pw}(H) \leq \mathbf{lw}(H) \leq 3 \cdot \delta$.*

Proof : There exists an algorithm that in $O(n)$ steps checks if it is a halin graph G and, in case of a positive answer, returns its skeleton T (we omit the proof because of space restrictions). After computing T, we use the linear algorithm of Skodinis in of [9] to compute the pathwidth of T in $O(n)$ time and output $\delta = \mathbf{pw}(T) + 1$. From Lemma 4, $\mathbf{lw}(H) \leq 3 \cdot \mathbf{lw}(T)$. Recall that $\mathbf{lw}(T) \leq \delta$ and therefore $\mathbf{lw}(H) \leq 3 \cdot \delta$. Also recall that $\mathbf{pw}(H) \leq \mathbf{lw}(H)$ and as T is a subgraph of H we also have $\delta - 1 = \mathbf{pw}(T) \leq \mathbf{pw}(H)$.

References

[1] Hans L. Bodlaender. Planar graphs with bounded treewidth. Technical Report RUU-CS-88-14, Dept. of Computer Science, Utrecht University, Utrecht, the Netherlands, 1988.

[2] Hans L. Bodlaender and Fedor V. Fomin. Approximation of pathwidth of outerplanar graphs. *J. Algorithms*, 43(2):190–200, 2002.

[3] Hans L. Bodlaender and Ton Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21:358–402, 1996.

[4] Nathaniel Dean. Open problems. In *Graph Structure Theory*, pages 677–688. AMS, Providence, Rhode Island, 1993.

[5] J. A. Ellis, I. H. Sudborough, and J. S. Turner. The vertex separation and search number of a graph. *Inform. and Comput.*, 113(1):50–79, 1994.

[6] R. Halin. Über simpliziale Zerfällungen beliebiger (endlicher oder unendlicher) Graphen. *Math. Ann.*, 156:216–225, 1964.

[7] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. Assoc. Comput. Mach.*, 35(1):18–44, 1988.

[8] Neil Robertson and Paul D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Theory Series B*, 35:39–61, 1983.

[9] Konstantin Skodinis. Construction of linear tree-layouts which are optimal with respect to vertex separation in linear time. *J. Algorithms*, 47(1):40–59, 2003.

[10] Dimitrios M. Thilikos. Algorithms and obstructions for linear-width and related search parameters. *Discrete Appl. Math.*, 105(1-3):239–271, 2000.

[11] Robin Thomas. Tree-decompositions of graphs. Technical report, Lecture notes, School of Mathematics, Georgia Institute of Technology, 1996.

On $(P_5, \overline{P_5})$-sparse graphs and other families

Jean-Luc Fouquet [1] , Jean-Marie Vanherpe [2]

L.I.F.O, Université d'Orléans, 45067, Cédex 2, France

Abstract

We extend the notion of P_4-sparse graphs previously introduced by HOÀNG in [12] by considering $\mathcal{F}$-sparse graphs were $\mathcal{F}$ denotes a finite set of graphs on p vertices. Thus we obtain some results on $(P_5, \overline{P_5})$-sparse graphs already known on $(P_5, \overline{P_5})$-free graphs. Finally we completely describe the structure of $(P_5, \overline{P_5}, bull)$-sparse graphs, it follows that those graphs have bounded clique-width.

1 Introduction

P_4-free graphs, also called *Cographs*, were designed to be completely decomposable by complementation and motivated researchers for studying graph classes characterized with forbidden configurations. In addition, a number of optimization problems on a graph can be reduced to their weighted version on the set of subgraphs which are indecomposable with respect to modular decomposition (see [14]). Thus sub-classes of P_5- free graphs were intensively studied (see e.g. [3, 4, 5]), in particular FOUQUET in [8] consider $(P_5, \overline{P_5})$-free graphs and the subclass of $(P_5, \overline{P_5}, Bull)$-free graphs (see Figure 1). Later GIAKOUMAKIS and RUSU [11] provide efficient solutions for some optimization problems on $(P_5, \overline{P_5})$-free graphs.

By the way, Hoàng introduced in [12] the P_4-sparse graphs. In a P_4-sparse graph, every induced subgraph on 5 vertices contains at most one P_4. Several extensions of P_4-sparse graphs have arisen, let's cite P_4-tidy [10],(q, t) graphs [2, 1] (in a (q, t)-graph every induced subgraph on q vertices contains at most t induced P_4), or $PL-graphs$ [15] where the number of partners of a given P_4 is at most 2 (a vertex x is said to be a partner of a P_4 $abcd$ when x together with 3 vertices of $\{a, b, c, d\}$ induces a P_4). Moreover, subclasses of bipartite P_7-sparse graphs were defined and studied in [9].

In this paper, we extend the notion of P_4-sparse in the following way : A graph G is said to be $\mathcal{F}$-sparse, where $\mathcal{F}$ denotes a set of graphs of order p, whenever any induced subgraph of G on $p+1$ vertices contains at most one graph of $\mathcal{F}$ as induced subgraph. We first propose a theorem which recursively reduces the recognition problem of $\mathcal{F}$-sparse graphs (when $\mathcal{F}$ is a set of prime graphs) to those of the set of representative graphs (recall that the representative graph of graph G is obtained from G by contracting every maximal proper module of G into

[1] E-mail: `fouquet@lifo.univ-orleans.fr`.
[2] E-mail: `vanherpe@lifo.univ-orleans.fr`.

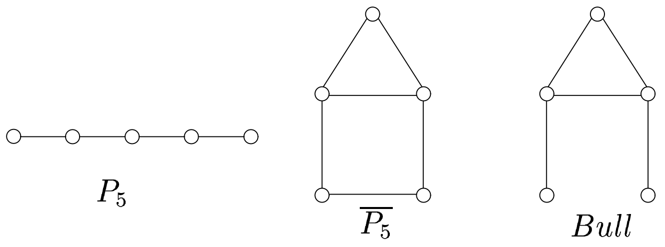


Figure 1. The forbidden configurations in a $(P_5, \overline{P_5}, Bull)$-free graph

a single vertex). Then we study $(P_5, \overline{P_5})$-sparse and $(P_5, \overline{P_5}, Bull)$-sparse, graphs classes defined with configurations which are prime with respect to modular decomposition (see Figure 1) and which properly intersect graphs classes such that PL-graphs or some (q,t)-graphs classes.

A recognition theorem

Definition 1 *A vertex x of a graph G is $\mathcal{F}$-special whenever x belongs to an induced subgraph of G which is isomorphic to a graph of $\mathcal{F}$.*

When $\mathcal{F}$ is a set of prime graphs we have the following recognition theorem.

Theorem 1 *Let $\mathcal{F}$ be a set of prime graphs.*
A graph G is $\mathcal{F}$-sparse if and only if the following holds :

(1) The representative graph of G is $\mathcal{F}$-sparse.
(2) For every $\mathcal{F}$-special vertex x, the module represented by x is a singleton.
(3) For every vertex x which is not $\mathcal{F}$-special, the module represented by x induces a $\mathcal{F}$-sparse graph.

2 On $(P_5, \overline{P_5})$-sparse graphs.

In this section we consider $\mathcal{F}$-sparse graphs when $\mathcal{F} = \{P_5, \overline{P_5}\}$ and we call those graphs $(P_5, \overline{P_5})$-sparse. Recall that in a such graph every induced subgraph on 6 vertices contains at most one P_5 or $\overline{P_5}$.

Theorem 2 *A prime $(P_5, \overline{P_5})$-sparse graph is either C_5-free or isomorphic to a C_5.*

Welsh-Powell perfect graphs are perfectly orderable and are characterized with 17 forbidden configurations (see [6]). It is a straightforward exercise to see that $(P_5, \overline{P_5})$-sparse graphs which are also C_5-free are Welsh-Powell perfect. In [13], Hoàng gives algorithms to solve the *Maximum Weighted Clique* problem as well as the *Minimum Weighted Coloring* problem on perfectly orderable graphs within $O(nm)$ time complexity.

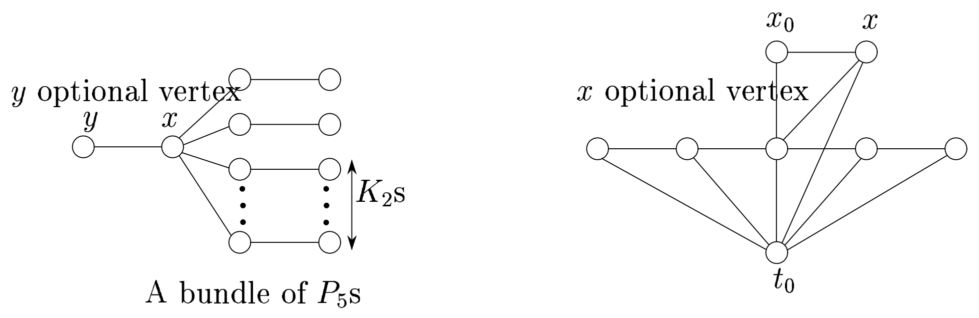


Figure 2. The 2 types of prime $(P_5, \overline{P_5}, Bull)$-sparse graphs which are C_5-free and contain a P_5.

Thus, as well as for $(P_5, \overline{P_5})$-free graphs (see [11]), there exists algorithms running in $O(nm)$ time, for computing a *Maximum Weigted Clique* and a *Minimum Weighted Coloring* in a weighted $(P_5, \overline{P_5})$-sparse graph. Since the class of $(P_5, \overline{P_5})$-sparse graphs is auto-complementary the parameters *Maximum Weighted Stable Set* and *Minimum Weighted Clique Cover* can be computed within the same time complexity.

3 $(P_5, \overline{P_5}, Bull)$-sparse graphs.

In this section we will study $\mathcal{F}$-sparse graphs where $\mathcal{F} = \{P_5, \overline{P_5}, Bull\}$, namely the $(P_5, \overline{P_5}, Bull)$-sparse graphs. We will characterize the prime graphs of this family and give some consequences.

Let's first recall a main result on $(P_5, \overline{P_5}, Bull)$-free graphs.

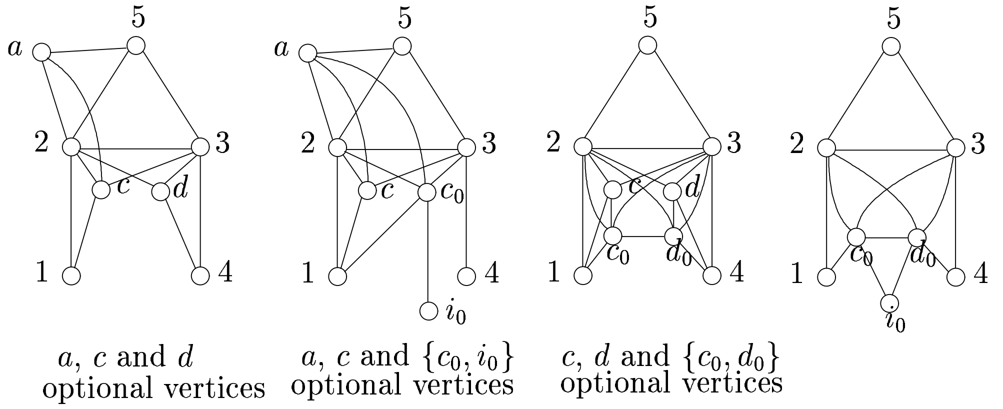
Theorem 3 *([8]) A prime graph G is $(P_5, \overline{P_5}, bull)$-free if and only if one of the following holds :*

(1) G is isomorphic to a C_5
(2) G or its complement is a bipartite P_5-free graph.

Since Theorem 2 also holds for $(P_5, \overline{P_5}, Bull)$-sparse graphs we consider henceforth only C_5-free graphs.

Theorem 4 *Let G be a prime C_5-free which contains an induced P_5 (resp. $\overline{P_5}$).*
G is $(P_5, \overline{P_5}, Bull)$-sparse if and only if G (resp. $\overline{G}$) is isomorphic to one of the graphs depicted in Figure 2.

Observe on one hand that the number of partner for a P_4 in a bundle of P_5 is not limited (see Figure 2) and on the other hand that a P_6 is not a $(P_5, \overline{P_5}, Bull)$-sparse graph. Thus the class of $(P_5, \overline{P_5}, Bull)$-sparse graphs properly intersects the class of PL-graphs. Moreover, since a bundle of P_5 on $2k+1$ vertices contains at least $k(k-1)$ distinct P_4's, there exist $(P_5, \overline{P_5}, Bull)$-sparse graphs which do not belong to the (q,t)-graphs family studied in [1].



5
a
2
3
c
d
1
4
a, c and d optional vertices
5
a
2
3
c
c0
1
4
i0
a, c and {c0, i0} optional vertices
5
2
3
c
d
c0
d0
1
4
c, d and {c0, d0} optional vertices
5
2
3
c0
d0
1
4
i0

Figure 3. The 4 types of prime $(P_5, \overline{P_5}, Bull)$-sparse graphs which are $(P_5, \overline{P_5}, C_5)$-free and contain a *Bull*.

Theorem 5 *Let G be a prime $(P_5, \overline{P_5}, C_5)$-free graph which contains an induced bull. G is $(P_5, \overline{P_5}, Bull)$-sparse if and only if G is isomorphic to one of the graphs depicted in Figure 3.*

It follows from Theorem 3, 2, 3 and 5 that a prime $(P_5, \overline{P_5}, Bull)$-sparse graph or its complement is either a C_5 or a P_5-free bipartite graph or a bundle of P_5s (see Figure 2) or is a graph on less than 10 vertices. Consequently Theorem 1 leads to a linear time recognition algorithm for $(P_5, \overline{P_5}, Bull)$-sparse graphs, moreover those graphs have bounded clique-width (see [7]).

References

[1] L. Babel and S. Olariu. On the isomorphism of graphs with few P_4s. *Lecture notes in Computer Science*, 944, WG 95:24–36, 1995.

[2] L. Babel and S. Olariu. A new characterisation of P_4-connected graphs. *Lecture notes in Computer Science*, 1197,WG 96:17–30, 1996.

[3] G. Bacsó and ZS. Tuza. Dominating cliques in P_5-free graphs. *Periodica Mathematica Hungarica*, 21:303–308, 1990.

[4] A. Brandstädt and D. Kratsch. On the structure of (P_5, gem)-free graphs. 2002. To appear in *Discrete Applied Mathematics*.

[5] A. Brandstädt and R. Mosca. On the structure and stability number of P_5 and co-chair-free graphs. *Discrete Applied Mathemaitics*, 2003.

[6] V. Chvátal, C.T. Hoàng, N.V.R. Mahadev, and D. de Werra. Four classes of perfectly orderable graphs. *Journal of Graph Theory*, 11:481–495, 1987.

[7] B. Courcelle, J.A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique width. *Lecture Notes in Computer Science*, 1517:1–16, 1998.

[8] J.L. Fouquet. A decomposition for a class of $(P_5, \overline{P_5})$-free graphs. *Discrete Mathematics*, 121:75–83, 1993.

[9] J.L. Fouquet and J.M. Vanherpe. On bipartite graphs with weak density of some subgraphs. 2003. To appear in the special issue of *Discrete Mathematics* devoted to the Fourth Cracow Conference on Graph Theory "Czorsztyn '02".

[10] V. Giakoumakis, F. Roussel, and H. Thuillier. On P_4-tidy graphs. *Discrete Mathematics and Theoretical Computer Science*, 1:17–41, 1997.

[11] V. Giakoumakis and I. Rusu. Weighted parameters in $(P_5, \overline{P_5})$-free graphs. *Discrete Applied Mathematics*, 80:255–261, 1997.

[12] C.T. Hoàng. *Perfect Graphs.* PhD thesis, School of Computer Science, McGill University, Montreal, 1985.

[13] C.T. Hoàng. Efficient algorithms for minimum weighted colouring of some classes of perfect graphs. *Discrete Applied Mathematics*, 55:133–143, 1994.

[14] R.H. Möhring and F.J. Radermacher. Substitution decomposition for discrete structures and connections with combinatorial optimization. *Annals of Discrete Mathematics*, 19:257–356, 1984.

[15] F. Roussel, I. Rusu, and H. Thuillier. On graphs with limited number of P_4-partners. *International Journal of Fundations of Computer Science*, 10:103–121, 1999.

Lexico-smallest representations, duality and matching polyhedra

Komei Fukuda [1]

Institute For Operations Research, ETH, Switzerland

Stefano Picozzi [2]

Institute of Mathematics, EPFL, Switzerland

Abstract

Every convex polyhedron in the Euclidean space $\mathbb{R}^d$ admits both H- and V-representations. In both types, a representation is *canonical* if it is minimal and unique up to some elementary operations. In this paper, we show the duality of canonical representations and that the canonical V-representations coincide with certain canonical H-representations. Also, we show how the lexico-smallest representation, a computationally convenient alternative to the usual orthogonal representation, can be computed efficiently. Finally, we illustrate our results by considering H-representations of the perfect matching polytope. In particular, we show that using the properties of the underlying graph results in sensible improvements in the running time of the computation of canonical representation.

1 Introduction

A *(convex) polyhedron* in $\mathbb{R}^d$ is the solution set to a finite system of inequalities with real coefficients in d real variables. For a matrix $A \in \mathbb{R}^{m\times d}$, a vector $b \in \mathbb{R}^m$ and a partition (I, L) of $[m] := \{1, 2, \ldots, m\}$, a quadruple (b, A, I, L) is said to be an *H-representation* of a convex polyhedron P if $P = \{x \in \mathbb{R}^d \mid b_I + A_I x \geq 0,\ b_L + A_L = \mathbf{0}\}$. For matrices $V \in \mathbb{R}^{p\times d}$, $R \in \mathbb{R}^{q\times d}$ and $M \in \mathbb{R}^{r\times d}$, a triple (V, R, M) is said to be a *V-representation* of a polyhedron P if $P = conv(V) + cone(R) + lin(M)$, where $conv(A)$, $cone(A)$ and $lin(A)$, denote respectively the convex hull, the nonnegative hull and the linear hull of the row vectors of the matrix A. Motzkin's decomposition theorem (see, e.g. [2, 3]) states that every polyhedron has both H- and V-representations.

Clearly, neither H- nor V-representation is unique. In [1], we described a family of polynomialy computable H- and V-representations that were unique up to some elementary operations. In particular we defined the *lexico-smallest representation* which guarantees certain sparsity

[1] E-mail: `fukuda@ifor.math.ethz.ch`.
[2] E-mail: `stefano.picozzi@epfl.ch`.

properties. In the present paper, we propose a slightly different definition which makes the computations easier. Also, we show the duality of canonical representations and that the canonical V-representations coincide with certain canonical H-representations. Finally, we illustrate our results by considering the case of the perfect matching polytope. In particular, we determine its lexico-smallest H-representation in terms of the underlying graph, which results in sensible improvements in the running times of the computation. Also, we show that in the case of complete bipartite graphs, the lexico-smallest representation is simpler than the *orthogonal representation*, its usual alternative described in [2, 3]. Finally, it is worth mentionning that a C-package computing the canonical representations of a polyhedron P from any other representation of P will be released in the future.

2 Representations of convex polyhedra

We define a quadruple (b, A, I, L) to be an H-representation of the polyhedron $P_H = \{x \in \mathbb{R}^d \mid b_I + A_I x \geq \mathbf{0},\, b_L + A_L x = \mathbf{0}\}$ and a V-representation of the polyhedron $P_V = \{x \in \mathbb{R}^d \mid x = A^T y,\, y_I \geq \mathbf{0},\, b^T y = 1\}$. We will also refer to H- and V-representations as $*$-*representation*, intended that $*$ refers to one of H or V. A V-representation is called *standard* if $b_i \in \{0, 1\}$ and $b_L = \mathbf{0}$. Note that (b, A, I, L) is a standard V-representation of $P_V = conv(A_{I^+}) + cone(A_{I^0}) + lin(A_L)$, where $I^0 := \{i \in I \mid b_i = 0\}$ and $I^+ := \{i \in I \mid b_i = 1\}$. As any V-representation can be transformed to a standard V-representation of the same polyhedron in quadratic time, we assume for the sequel of the paper that every V-representation is standard.

2.1 Canonical representations

We say two representations (b, A, I, L) and (b', A', I', L') of the same type *equivalent* if the represented polyhedra are equal. They are said to be *equal* if $b_L + A_L x = 0 \Leftrightarrow b_{L'} + A_{L'} x = 0$ and if there is a permutation π of I such that $\pi(I) = I'$ and each (b'_i, A'_i) is a positive multiple of $(b_{\pi(i)}, A_{\pi(i)})$ for any $i \in I$. Note that for V-representations, $b_L = \mathbf{0}$ and then the first equivalence coincides with the statement $\{x \in \mathbb{R}^d \mid x = (A_L)^T y\} = \{x \in \mathbb{R}^d \mid x = (A_{L'})^T y\}$.

For an index set J we let $J + i := J \cup \{i\}$ and (provided that $i \in J$) $J - i := J \setminus \{i\}$. A row index $i \in [m]$ is called *redundant in a representation* (b, A, I, L) *of* P if $(b, A, I - i, L)$ is a representation of P. We say that $i \in I$ is in the *implicit linearity* of (b, A, I, L) if $(b, A, I - i, L + i)$ is a representation of P. It is *minimal* if it has no redundant row index and has empty implicit linearity. For every polyhedron P, we let $L_V(P) := lin.space(P)$ and $L_H(P) := aff(P)^\perp$, where $lin.space(P) = \{z \in \mathbb{R}^d \mid x + \lambda z \in P,\, \forall x \in P,\, \lambda \in \mathbb{R}\}$ and $aff(P)^\perp$ is the orthogonal complement of the affine hull $aff(P)$ of P. Finally, we say two linear subspaces S_1 and S_2 of $\mathbb{R}^d$ are *complementary* if every basis of S_1 and every basis of S_2 form, together, a basis of $\mathbb{R}^d$.

Theorem 1 *A minimal $*$-representation (b, A, I, L) of a nonempty polyhedron P such that all row vectors A_i, $i \in I$, belong to a fixed linear subspace S complementary with $L_*(P)$ exists and is unique.*

Selecting S as the orthogonal complement $L_*(P)^\perp$ of $L_*(P)$ results in the *orthogonal representation*. Another choice is to let S be a *coordinate subspace*, which is any vector subspace of $\mathbb{R}^d$ generated by some unit vectors e^j, $j = 1, 2, \ldots, d$. It is easy to show that $S := lin(\{e^j\})_{j \notin J}$ and $L_*(P)$ are complementary if and only if the columns of A_{LJ} form a basis of the space spanned by the columns of A_L. Requiring that J is lexicographically largest results in the *lexico-smallest representation*. Clearly, the matrix A_I of this representation has at least $|L| = rank(A_L)$ zero columns. This definition of the lexico-smallest representation, which differs from the one proposed in [1], has the advantage that the computation of S amounts to apply a single gaussian elimination on the matrix A_L, instead of the $O(d)$ eliminations required with the previous definition.

2.2 Canonical V-representations via H-representations

Here, we show the duality of canonical representations and that the canonical V-representations coincide with certain canonical H-representations. Firstly,

Theorem 2 *A representation $(\mathbf{0}, A, I, L)$ is a canonical H-representation of a cone C if and only if it is a canonical V-representation of the polar C^* of C.*

Now, for each V-representation (H-representation, respectively) (b, A, I, L) of a nonempty polyhedron P, we define $[b^0\ A^0] := [b\ A]$ ($[b^0\ A^0] := [b\ A]$ if P is bounded and $[b^0\ A^0] := \begin{bmatrix} 1 & \mathbf{0}^T \\ b & A \end{bmatrix}$ otherwise.)

Theorem 3 *Let (b, A, I, L) be a $*$-representation of a nonempty polyhedron P. Then, (b, A, I, L) is a canonical $*$-representation $\Leftrightarrow$ $[b^0 A^0]$ is a canonical $*$-representation.*

Corollary 3.1 *Let (b, A, I, L) be a V-representation of $P_V \ni \mathbf{0}$. Then, (b, A, I, L) is a canonical V-representation if and only if it is a canonical H-representation.*

2.3 From minimal to canonical representations

Let (b, A, I, L) be any minimal $*$-representation of a nonempty polyhedron P.

Lemma 1 *Let $S := lin(\{e^j\}_{j \in J})$ be complementary $L_*(P)$, and let $\bar{A}_L$ the matrix arising from A_L by setting to zero its columns $j \notin J$. Then,*

(1) the rows A'_i of the matrix A'_I of the orthogonal $*$-representation are the rows $A'_i := A_i - \lambda A_L$, $i \in I$, where $\lambda(A_L A_L^T) = A_i A_L^T$;
(2) the rows of the matrix $A'_{I'}$ of the lexico-smallest $*$-representation are the rows $A'_i := A_i - \lambda A'_{L'}$, $i \in I$, where $\lambda(\bar{A}_L \bar{A}_L^T) = A_i \bar{A}_L^T$.

3 Perfect matching polyhedra

We apply the results of the last section to the perfect matching polytope. Given a bipartite graph $G = (V, E)$ the perfect matching polytope $P_{MA}(G)$ is

$$P_{MA}(G) = \left\{ x \in \mathbb{R}^{|E|} \;\middle|\; \begin{array}{ll} \sum\limits_{e \in \delta(v)} x_e = 1 & \forall v \in V \\ x_e \geq 0 & \forall e \in E, \end{array} \right\} \tag{1}$$

where $\delta(V)$ denotes the set of edges which have exactly one endnode in V. In the following, G denotes a bipartite graph which contains a perfect matching.

Lemma 2 *A subspace $S; = lin(\{e^j\}_{j \in F})$ is complementary with aff$(P_{MA}(G))^\perp$ if and only if (V, F) is a spanning forest in $G = (V, E)$. In turn, if G has c connected components, the affine hull of $P_{MA}(G)$ has dimension $|E| - |V| + c$.*

From now on, we assume that G is connected. We let $T := (V, E_T)$ be any fixed spanning tree in G, and for every $W \subset V$ we let $T[W]$ be the *subtree of T induced by W*. For each $\bar{e} \in E_T$, we denote by $FC_G(T, \bar{e})$ the *fundamental cycle of T with $\bar{e}$*, by $FC^*_G(T, \bar{e})$ the *fundamental cut of T with $\bar{e}$*, and by V_e any of the nodesets such that $FC^*_G(T, \bar{e}) = \delta(V_{\bar{e}})$. We define $\delta^+(V_{\bar{e}})$ ($\delta^-(V_{\bar{e}})$, repsectively) as the set of edges $e \neq \bar{e} \in \delta(V_{\bar{e}}))$ such that $FC^*_G(T, e) \cap T[V_{\bar{e}}]$ has an odd (even) number of vertices. Finally, $E^+(V_{\bar{e}})$ ($E^+(V_{\bar{e}})$, respectively) denotes the set of edges $e \neq \bar{e}$ in $T[V_{\bar{e}}]$ such that the smallest path in T containing both e and $\bar{e}$ has an odd (even) number of edges. If T is the lexicographically largest spanning tree in G and $\Delta_{E(V_{\bar{e}})} := |E^+(V_{\bar{e}})| - |E^-(V_{\bar{e}})|$ we have,

Theorem 4 *The affine hull of $P_{MA}(G)$ is the solution set to the following minimal system of equations: For all $\bar{e} \in T$,*

$$x_{\bar{e}} + \sum_{e \in \delta^+(V_{\bar{e}})} x_e - \sum_{e \in \delta^-(V_{\bar{e}})} x_e = 1 + \Delta_{E(V_{\bar{e}})}.$$

Theorem 5 *Let $x_{\bar{e}} \geq 0$ be any nonredundant inequality in (1), and assume that it is not an implicit equation. Then, the corresponding inequality in the lexico-smallest representation is $x_{\bar{e}} \geq 0$ if $\bar{e} \in E \setminus E_T$ and*
$\sum\limits_{e \in \delta^+(V_{\bar{e}})} x_e \leq 1 + \Delta_{E(V_{\bar{e}})}$ otherwise.

As a consequence, A_I is a $(\{-1, 0, 1\})$-matrix, and its $2n - 1$ columns corresponding to edge in $T = (V, E_T)$, $|V| = 2n$, are completely zero. Furthermore $|b_i| \leq n$ for all $i \in [m]$. In the case when $G = K_{n,n}$, we have

Property 1 *Let $x_{\bar{e}} \geq 0$, $\bar{e} = (u, v)$, be any inequality in (1). Then, the corresponding inequality in the orthogonal representation is*

$$(n-1)^2 x_{\bar{e}} - (n-1) \sum_{e \in \delta(\{u,v\})} x_e + \sum_{e \notin \delta(\{u,v\})} x_e \geq -n,$$

Note that A_I is completely dense. Also, the numbers in the lexico-smallest representation are smaller than the one in the orthogonal representation.

To conclude, note that using properties of the graph results in sensible improvements of the running time of the computation: removing linearly independent rows of A_L amounts to computing a spanning tree, while transforming the inequalities amounts to computing the length of certain paths in a tree. In the full paper, we will discuss how similar improvements can be obtained for the computation of minimal representations.

References

[1] D. Avis, K. Fukuda, S. Picozzi. On Convex Representation of Convex Polyhedra. *Proc. of the First International Congress of Mathematical Software*, 350–360, 2002.

[2] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, New York, 1986.

[3] G.M. Ziegler. *Lectures on polytopes.* Graduate Texts in Mathematics 152. Springer-Verlag, 1994.

The set of prime extensions of a graph: the finite and the infinite case

Vassilis Giakoumakis [a 1] Stephan Olariu [b 2]

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

[b] *Dept. of Computer Science, Old Dominion University, USA*

Abstract

Let H be a graph then a graph H' is a *prime extension* of H if H' is prime (in the sense of modular decomposition), it contains an induced subgraph isomorphic to H and is minimal with respect to set inclusion and primality. An open problem concerning the set of prime extensions $Ext(H)$ of H is the following: find the necessary and sufficient conditions establishing the finiteness of $Ext(H)$. We solve the above problem by characterizing all classes of graphs whose set of prime exensions is finite. We give also a simple way for generating an infinite number of extensions for each graph belonging to any other class of graphs.

Key words: module, modular decomposition, prime extension

1 Notations and previous results

All considered graphs are finite, without loops nor multiple edges. For termes not defined here the reader is refered to [1]. Let $G = (V, E)$ be a graph with vertex set V (or $V(G)$) and edge set E. The subgraph induced by $X \subseteq V$ will be denoted $[X]$ and the set of vertices outside X that are adjacent to at least one vertex of X will be denoted $N(X)$. A chordless chain of k vertices will be denoted P_k. A set $M \subseteq V$ is a module of G if every vertex outside M is adjacent to all vertices of M or to none of them. The empty set, V and the singletons are trivial modules and whenever G has only trivial modules is called *prime*. M will be a *strong module* if for any other module M' of G we have either $M \cap M' = \oslash$ or one module is included to the other. A graph whose every module is a subset of a P_4 will be called P_4*-homogeneous* graph. The modular decomposition of G associates to G a unique decomposition tree $T(G)$ whose set of leafs is V and each set of leafs of the subtree rooted on an internal node of $T(G)$, forms a strong module of G. An internal node is labeled P, S or N if its corresponding set of leafs induces respectivily an edgeless, complete or a prime graph. The substitution graph G of two disjoint graphs G_1 and G_2 is obtained by first removing a vertex x from G_2 and then making every vertex in G_1 adjacent to all neighbours of x in G_2. Let F a family of graphs defined by a set Z of forbidden configurations. We

[1] E-mail: `giakouma@laria.u-picardie.fr`.
[2] E-mail: `olariu@cs.odu.edu`.

know ([3]) that the closure F^* under substitution of F can be characterized by a set Z^* of forbidden configurations, these configurations being all the *prime extensions* of Z. We know also ([3]) that Z^* is not necessarily a finite set. Since substitution preserves many of the properties of the composed graphs (as for exemple perfection), an important open problem is the following: find the necessary and sufficient conditions establishing the finiteness of Z^*. Different researchers investigated the solution of this problem and many sufficient conditions were presented for it (see for example [2], [3] and [6]). In this paper we solve this problem by characterizing all classes of graphs having a finite number of prime extensions. We give also a simple way for generating an infinite number of prime extensions for each graph belonging to any other class of graphs. A complete version of this extended abstract can be found in [4].

Theorem 1.1 ([6]) *Let G be a P_4-homogeneous graph, then $Ext(G)$ is a finite set.*

Definition 1.2 ([5]) Let G be an induced subgraph of a graph H, and let W be a homogeneous set of G. We define a reducing *W-pseudopath* in H as a sequence $R = (u_1, u_2, ..., u_t)$, $t \geq 1$, of pairwise distinct vertices of $V(H) \setminus V(G)$ satisfying the following conditions:

(1) u_1 is partial for W.
(2) $\forall\ i = 2, ..., t$ either u_i est adjacent to u_{i-1} and indifferent to $W \cup \{u_1, ..., u_{i-2}\}$ or u_i est adjacent to $W \cup \{u_1, ..., u_{i-2}\}$ and not adjacent to u_{i-1} (when $i = 2$, $\{u_1, u_2, ..., u_t\} = \oslash$).
(3) $\forall\ i = 2, ..., t-1$, u_i is total to $N(W)$ and indifferent to $V(G) - N(W)$ and either u_t is not adjacent to a vertex of $N(W)$ or u_t is adjacent to a vertex of $V(G) - N(W)$.

Theorem 1.3 ([5]) *Let H be an extension of its induced subgraph G and let W be a homogeneous set of G. Then there exists a reducing W-pseudopath with respect to any induced copy of G in H.*

In the two following sections we shall give two constructions for obtaining a prime extension of a decomposable graph which will be the framework for our main result given in the the last section.

2 The basic extension of a decomposable graph

Let $G = (V, E)$ be a connected graph and let $T(G)$ be the corresponding modular decomposition tree. Let $\pi(G) = \{H_1, ..., H_l\}$ be a partition of V obtained by the following equivalence relation R in V: for two vertices $x, y \in V$ we have that xRy if and only if x and y have the same father in $T(G)$. Assume in the following of this section that G is not prime. Consequently there exists $H_i \in \pi(G)$ which is a non trivial module of G.Let $\rho(G) = \{M_1, ..., M_k\}$ be the subset of $\pi(G)$ such that every $M_i, i = 1, ..., k$, is a non trivial module in G . Let us associate with every module M_i of $\rho(G)$ a set V_i' of new vertices (i.e. $V_i' \cap V = \emptyset, V_i' \cap V_j = \emptyset$, $i, j = 1, ..., k$, $i \neq j$) and a set E_i' of edges relying the vertices of M_i with the vertices of V_i' in the following manner:

(1) if M_i is a stable set or a complete set $\{x_1, ..., x_r\}$ then $V_i' = \{y_1, ..., y_{r-1}\}$ and E_i' is the set of edges $x_j y_j$, $j = 1, ..., r-1$.
(2) if M_i induces in G a prime graph then V_i' is a singleton $\{y\}$ and E_i' is the edge yx where x is a vertex of M_i.

Let $basic(G)$ be the graph whose vertex set is $V \cup V'$, where $V' = V'_1 \cup ... \cup V'_k$ and whose edge set is $E \cup E'$, where $E' = E'_1 \cup ... \cup E'_k$. Clearly V' is a stable set in $basic(G)$ and each vertex of this set has exactly one neighbour in $basic(G)$, this neighbour being its own *"private"* neighbour.

Theorem 2.1 *Basic(G) is an extension of G.*

Outline of the proof. The primality of $basic(G)$ follows from the fact that in every non trivial module M of G there exists a vertex of M having a private neighbour in V'. $Basic(G)$ is a prime extension of G since every proper subgraph of $basic(G)$ containing a copy of G is not a prime graph. □

3 The path extension of a decomposable graph

The proof of the following theorem suggests a way of constructing an infinite number of extensions of a graph G.

Theorem 3.1 *Let G be a connected graph containing a maximal non trivial module M inducing a connected graph not isomorphic to a P_k, then $Ext(G)$ is an infinite set.*

Outline of the proof. We may assume w.l.o.g. that M is maximal with respect to set inclusion, connectivity and the fact that $[M]$ is not isomorphic to a chordless chain. Let A be the neighbourhood of M in G and B its non neighbourhood. Since G is supposed to be connected we have that A $\neq \oslash$. Consider the graphe $basic(G)$ and denote by Q the set of vertices of $V(basic(G)) - V(G)$ such that $N(Q) \subset M$ and by D the vertices of $V(G') - V(G)$ such that $N(D) \subset \{A \cup B\}$. In other words Q is the set of new vertices that *"break"* the module M in G and any non trivial module of $[M]$ and D is the set of new vertices that *"break"* any non trivial module of $[A \cup B]$ in the graph G. Clearly, $Q \cup D$ is a stable set and every vertex $x \in Q \cup D$, has exactly one neighbour in G and this neighbour is "private". Finaly let $D_A = N(D) \cap A$ and $D_B = N(D) \cap B$. Let G^+ be the graph obtained from $basic(G)$ in the following way: $V(G^+) = V(basic(G))$ and $E(G^+) = E(G') \cup \{\{x, y\} \mid x \in Q, y \in A\}$. In other words, every vertex of Q is adjacent in G^+ to every vertex of A which implies that $M \cup Q$ is a non trivial module of G^+. Let $G \otimes P_k$ be the graph obtained from G^+ in the following way: $V(G \otimes P_k) - V((G)^+)$ induces a chordless chain $P_k = x_1, ..., x_k$ such that x_1 is adjacent to exactly one vertex of Q , every vertex of $\{x_1, ..., x_{k-1}\}$ is total for A and not adjacent to any vertex of $M \cup B \cup D$, every vertex of $\{x_2, ..., x_{k-1}\}$ is not adjacent to any vertex of Q and finaly x_k is not adjacent to any vertex of G^+. The structure of $G \otimes P_k$ is illustrated in Figure 1 below:

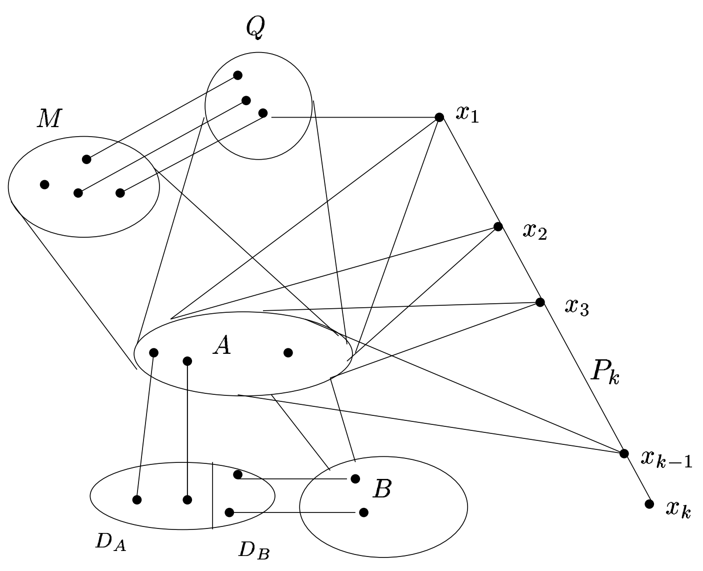


Figure 1: the path extension of a graph G

We first prove that $G \otimes P_k$ is a prime graph. Then we show that any prime subgraph of $G \otimes P_k$ containing a copy of G must contain the whole chain P_k which is of arbirtrary length. $\square$

4 The main result

We know that any P_4-homogeneous graph has a finite number of extensions ([6]). Let us define a second class of graphs, the class of $2P_4$-homogeneous graphs having a finite number of extensions too.

Definition 4.1 Let G be a connected which is not P_4-homogeneous such that $\overline{G}$ contains exactly two connected components C_1 and C_2. Then G and $\overline{G}$ are called $2P_4$ - *homogeneous* if $[C_1]$ is a P_4-homogeneous graph and $[C_2]$ is a subgraph of a P_4.

Theorem 4.2 *Let G be a graph, then $Ext(G)$ is a finite set if and only if G is a P_4-homogeneous graph or a $2P_4$-homogeneous graph.*

Outline of the proof. The structure of this proof is depicted in Figure 2 below. The reader can be find the whole proof in [4]. We only point out here that whenever G is a $2P_4$-homogeneous graph the finiteness of $Ext(G)$ is obtained using Theorem 1.1 and Theorem 1.3. Indeed, in any extension G^* of G, any W-pseudopath whose length is greater than the cardinality of the module W creates a copy of G in G^* which implies that $\mid V(G^*) \mid = \mid V(G) \mid +c$, where c is a constant. $\square$

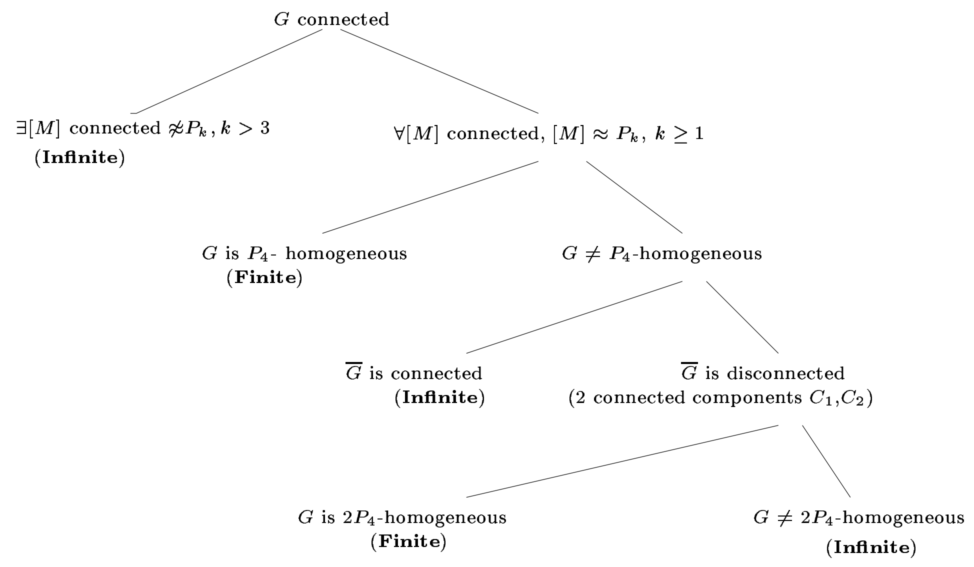

G connected
$\exists[M]$ connected $\not\approx P_k$, $k > 3$
(**Infinite**)
$\forall[M]$ connected, $[M] \approx P_k$, $k \geq 1$
G is P_4- homogeneous
(**Finite**)
$G \neq P_4$-homogeneous
$\overline{G}$ is connected
(**Infinite**)
$\overline{G}$ is disconnected
(2 connected components C_1,C_2)
G is $2P_4$-homogeneous
(**Finite**)
$G \neq 2P_4$-homogeneous
(**Infinite**)

Figure 2: the different cases that may occur for $Ext(G)$

References

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

[2] A. Brandstädt, C. Hoàng and J.M. Vanherpe, On minimal prime extensions of a four-vertex graph in a prime graph, submitted, *http://www.informatik.uni-rostock.de/(en)/~ab*

[3] V. Giakoumakis, On the closure of graphs under substitution,*Disc. Math.*, **177**, (1-3), (1997), 83-97.

[4] V. Giakoumakis, S. Olariu, All prime extensions of hereditary classes of graphs, *Rutcors research report RRR* **40**-2003, http:*//rutcor.rutgers.edu/~rrr/2003.html*

[5] I. Zverovich, Extension of hereditary classes with substitutions, *Disc. Appl. Math.* **128**, (2-3), (2003) 487-509

[6] I. Zverovich, A generalization of Giakoumakis's theorem, *Rutcor research report* , RRR **20**-2003, *http://rutcor.rutgers.edu/~rrr*

On a packet scheduling problem for smart antennas and polyhedra defined by circular-ones matrices

Dion Gijswijt [1]

Department of Mathematics, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands

Abstract

In [1, 2] E. Amaldi et al. posed a combinatorial optimization problem that arises when scheduling packets in a smart antenna. The objective is to partition the set of users so as to minimize the number of time slots needed to transmit all the given packets. Here we will present a polynomial time algorithm for solving this packet scheduling problem. More generally, the algorithm solves an integer decomposition problem for polyhedra determined by a circular-ones constraint matrix, which might make it interesting also for other cyclic scheduling problems.

Key words: packet scheduling, cyclic scheduling, polytime algorithm, integer programming

1 Introduction

In recent years, there has been a growing interest in adaptive antenna arrays known as "smart antennas". The combination of an antenna array and digital signal processing capabilities, enables a smart antenna to transmit and receive signals in a spatially sensitive manner. The spatial radiation pattern can be adjusted in real time in response to the signal environment. Exploiting this, signals to different users can be transmitted simultaneously over the same radio channel.

This allows us to view a smart antenna as a collection of co-located directive antennas that each transmit to (or receive from) a narrow beam (approximately 12 degrees). Each of these directive antennas can be independently oriented and can serve one user at a time. However, in order to avoid unwanted interference, there is a restriction on the sets of users that can be served simultaneously: a user that is being served, cannot be in the beam corresponding to a directive antenna that serves another user. This restriction limits the number of users that can be served during the same time slot.

As an example, suppose that the angle of the beams from the directive antennas is 12 degrees and that three users are in a common sector of 12 degrees. If the middle of the three users

[1] E-mail: `gijswijt@science.uva.nl`.

is served, then the beam corresponding to the antenna that serves it must either contain the clockwise or the anticlockwise neighbour which therefore cannot be served at the same time. This implies that for a set of users that are served simultaneously, the angle between any of these users and it's second clockwise neighbour is more than 12 degrees. Hence the number of users that can be served in a single time slot is less than 60. In fact we will assume that the number of available directive antennas is unlimited and the sets of users that can be served simultaneously are determined exactly by this interference constraint.

2 Modelling the packet scheduling problem

In [2], Amaldi et al. considered the following scheduling problem: given a set of users, we want to serve all of them, minimizing the total number of time slots needed. That is, we want to partition the users into a minimal number of classes, where the members of each class can be served simultaneously by the smart antenna.

Following Amaldi et al., we model the problem in the following manner. Since the exact position of the users is not needed, only their direction as seen from the smart antenna, we model the users by points on the unit circle and let the beams from the directive antennas correspond to arcs of a fixed length α of the unit circle. We will always assume that $0 < \alpha < 2\pi$.

For two points a, b on the unit circle the closed segment running clockwise from a to b is called an *arc* and is denoted by $[a, b]$. Let $\alpha > 0$ be fixed. A finite set S of points on the unit circle will be called *independent*[2] if there exist $|S|$ arcs on the unit circle, each of length α, such that each point in S is in exactly one of these arcs and each of these arcs contains exactly one element of S. Note that any two of the $|S|$ arcs may intersect as long as the intersection does not contain a point in S. The independent sets correspond to the sets of users that can be served simultaneously. The scheduling problem can now be stated as follows.

Problem 1 *Given a finite subset V of the unit circle, find a partition of V into a minimal number of independent sets.*

3 Results

A basic observation, which led us to a polynomial time algorithm for the scheduling problem is the following

Observation 1 *A finite set S of points on the unit circle is independent if and only if $|S \cap [s, s']| \leq 2$ for each arc $[s, s']$ of length α starting at a point $s \in S$.*

[2] In [2] arcs are half-open segments, but for the definition, this is equivalent to using closed segments of the same length.

To see necessity, suppose that some arc of length α contains $u, v, w \in S$ in this order, then any arc of length α containing v also contains u or w and hence S is not independent. For sufficiency, suppose that $|S \cap [s, s']| \leq 2$ for each arc $[s, s']$ of length α with $s \in S$. Let $v \in S$ and let u and w be the anticlockwise and clockwise neighbour in S of v respectively. The length of $[u, w]$ must be larger than α since $|[u, w] \cap S| > 2$, and hence there exists an arc of length α intersecting S only in v. Note that the last argument also shows that given an independent set S, $|S|$ arcs of length α as in the definition of independent set, are easily constructed from S.

This observation allows us to identify the independent sets with the zero-one solutions to a system of linear inequalities $Ax \leq \mathbf{2}$. Here the zero-one constraint matrix A is a *circular-ones matrix*: in each row of A, the ones or the zeros form a contiguous block. The scheduling problem is then to partition the all-one vector into a minimal number of zero-one solutions to this system. By extending the algorithm in [3] for colouring proper circular arc graphs, we obtain an algorithm that given an $m \times n$ circular-ones matrix A, a vector $b \in \mathbb{Z}_+^m$, and a vector $x \in \mathbb{Z}_+^n$, decomposes x into a minimal number of integral solutions to $Ax \leq b$. The running time of the algorithm is $O(nm \log(x^T \mathbf{1}))$. The packet scheduling problem is a special case where b is the all-two vector and x is the all-one vector. Applied to the packet scheduling problem, the algorithm finds in time $O(n^2 \log n)$ an optimal schedule, where n is the number of users.

References

[1] E. Amaldi, A. Capone, F. Malucelli, Circular Arc Models and Algorithms for Packet Scheduling in Smart Antennas, *IV ALIO/EURO Workshop on Applied Combinatorial Optimization*, see: http://www-di.inf.puc-rio.br/~celso/artigos/pucon.ps.

[2] E. Amaldi, A. Capone, F. Malucelli, Discrete models and algorithms for packet scheduling in smart antennas, *2nd Cologne Twente Workshop on Graphs and Combinatorial Optimization.*

[3] J.B. Orlin, M.A. Bonuccelli, D.P. Bovet, An $O(n^2)$ algorithm for colouring proper circular arc graphs, *SIAM J. Algebraic Discrete Methods* 2 (1981), no. 2, 88–93.

[4] A. Perez-Neira, X. Mestre, J.R. Fonollosa, Smart antennas in software radio base stations, *IEEE Communications Magazine*, Vol. 39 (2), Feb. 2001, 166–173.

[5] K. Sheikh, D. Gesbert, D. Gore, A. Paulraj, Smart antennas for broadband wireless access networks, *IEEE Communications Magazine*, Vol. 37 (11), Nov. 1999, 100–105.

More on orbital matrices

Harald Gropp [1]

Mühlingstr.19, D-69121 Heidelberg, Germany

Abstract

In this paper a nonexistence theorem of Schützenberger of 1949 for symmetric designs is discussed.This result together with the theorems of Bruck-Ryser and Chowla-Ryser yield many nonexistence results for orbital matrices as well. In the second part, relations of orbital matrices to other combinatorial structures, e.g. weighing matrices, are discussed.

Key words: Symmetric designs, Orbital matrices, Weighing designs, Configurations.

1 Introduction

In one of his early papers Schützenberger [7] proved that the order of a symmetric 2-design with an even number of points must be a square. This result is contained in the socalled theorem of Bruck-Ryser-Chowla of 1950, and hence the earlier contribution of Schützenberger is often forgotten.

The results of Schützenberger, Bruck, Ryser, and Chowla do not only apply to symmetric 2-designs but also to the bigger class of orbital matrices defined as follows.

Definition 1 *An orbital matrix $OM(v,k,x;\lambda)$ is a square matrix A of size v with non-negative integer entries such that*

(1) the sum of the entries in each row and in each column is equal to k,

(2) the sum of the square of the entries in each row and in each column is equal to $k+x$,

(3) the inner product of two different rows or two different columns is equal to λ where $\lambda = (k(k-1)-x)/(v-1)$.

Orbital matrices were introduced in [6]. The reader is referred to this paper for all details and results which are not explained here. Orbital matrices with $x = 0$ are exactly the incidence matrices of symmetric 2-designs.

[1] E-mail: `d12@ix.urz.uni-heidelberg.de`.

While the earlier paper [6] mainly discusses orbital matrices with small values of λ (i.e. $\lambda \leq 3$) and exhibits some nonexistence proofs for certain orbital matrices which pass the Bruck-Ryser- Chowla test the current paper will discuss some relations of orbital matrices to other combinatorial structures, mainly to weighing matrices.

2 The theorem of Schützenberger-Bruck-Ryser-Chowla

The following theorem is usually called the theorem of Bruck, Ryser, and Chowla or the BRC Theorem. In fact, it consists of three different theorems published in the papers of Schützenberger [7], Bruck and Ryser [1], and Chowla and Ryser [2].

Theorem 1 *The existence of a symmetric 2-design with parameters (v, k, λ) implies the following:*

If v is even, $k - \lambda$ is a square.

If v is odd, the diophantine equation

$$w^2 = (k - \lambda)y^2 + (-1)^{(v-1)/2}\lambda z^2$$

has a non-trivial solution (i.e. $(w, y, z) \neq (0, 0, 0)$) in integers.

As already mentioned the incidence matrix of a symmetric 2-design is an orbital matrix with $x = 0$. Anyhow, the definition of a symmetric 2-design can be given as follows.

Definition 2 *A symmetric 2-(v, k, λ)-design consists of v points and v blocks (subsets of points) such that each block contains k points, each point occurs k times on a block, and each pair of different points occurs together in a block exactly λ times.*

In [7] the case of even values of v is proved. In [1] the theorem is proved for projective planes, i.e. symmetric designs with $\lambda = 1$. Here only odd values of λ can occur since $v = 1+k(k-1)$. Finally, by using the result of [1] the general theorem is proved for all values of λ in [2].

This theorem was the first general nonexistence theorem in the field at all, and the whole theorem of Schützenberger-Bruck-Ryser-Chowla still is the only one. For even v no nonexistence case is known which is not a consequence of this theorem of Schützenberger. For odd v the only singular result is the nonexistence of the projective plane of order 10, i.e. a (111,11,1)-design.

A general nonexistence theorem for orbital matrices holds which exactly corresponds to the theorem of Schützenberger-Bruck-Ryser-Chowla in the case of symmetric designs. The reason is just that the proofs of the papers [7], [1], and [2] are valid for integer matrices and not only for 0-1-matrices.

Theorem 2 *(see [6])*

The existence of an orbital matrix $OM(v,k,x;\lambda)$ implies the following:

If v is even, $k+x-\lambda$ is a square.

If v is odd, the diophantine equation

$$w^2 = (k+x-\lambda)y^2 + (-1)^{(v-1)/2}\lambda z^2$$

has a non-trivial solution (i.e. $(w,y,z) \neq (0,0,0)$) in integers.

It is quite interesting that there are already many nonexistence results for orbital matrices which cannot be deduced from this general theorem. For details see [6].

3 Weighing matrices and orbital matrices

Orbital matrices with $x = 0$ or with only 2 different entries correspond to incidence matrices of symmetric 2-designs. The easiest and maybe most natural case of three different entries is the one of entries 0,1, and 2. These matrices are related to weighing matrices defined as follows.

Definition 3 *A weighing matrix $W(n,w)$ of weight $w \neq 0$ and order n is a square matrix of size n with entries from $\{-1,0,+1\}$ satisfying $WW^t = wI$.*

A $W(n,n)$ is called an Hadamard matrix of order n, a $W(n,n-1)$ is called a conference matrix of order n.

For further details and references concerning weighing matrices, Hadamard matrices, and conference matrices see [3] and [4].

The characteristic property of a weighing matrix is that it is orthogonal, i.e. the inner product of two different rows or columns is 0. In that sense the property of orbital matrices to have a constant inner product is just a generalization of being orthogonal. If the 3 entries 0, 1, and 2 of an orbital matrix are replaced by -1, 0, and +1 resp., the value of the inner product remains constant but is shifted. Sometimes it occurs that it is 0 which means that it is a weighing matrix.

However, not all weighing matrices are orbital matrices since the number of -1, 0, and +1 in a row or column need not be constant.

3.1 A series of examples

As an example the series of orbital matrices $OM(a,a-3,6;a-6)$ will be discussed. It follows that there are two different row-column types:

The 3-type $(d_0, d_1, d_2, d_3) = (5, a-6, 0, 1), a \geq 6$ and

the 2-type $(d_0, d_1, d_2) = (6, a-9, 3), a \geq 9$.

For details concerning type equations and row-column types see [6]. The value d_i denotes the number of entries i in a row or column. For example, in a 2-type there are 6 entries 0, $a-9$ entries 1, and 3 entries 2.

Weighing designs are used in order to obtain the following results on orbital matrices. A more systematical study is the task for the future.

Lemma 1 *(i) There exists an $OM(10,7,6;4)$ of 2-type.*

(ii) There is no $OM(11,8,6;5)$ of 2-type.

(iii) There exists an $OM(13,10,6;7)$ of 2-type.

Remark 1 *(i) There is no $OM(10,7,6;4)$ of 3-type. This was proved in [6].*

(ii)There are exactly 2 $OM(12,9,6;6)$ of 3-type and exactly 2 $OM(12,9,6;6)$ of 2-type. There is at least one mixed-type $OM(12,9,6;6)$ (see [6]).

(iii) There is the following hierarchy: Each circulant matrix is regular; each regular matrix $W(n,9)$ is an $OM(n,n-3,6;n-6)$ of 2-type; each $OM(n,n-3,6;n-6)$ of 2-type is a $W(n,9)$.

3.2 $OM(12,9,6;6)$ —some construction methods

$OM(12,9,6;6)$ as an example of the series of the previous section will be discussed in detail to show some typical construction methods of orbital matrices.

At first the matrices of 3-type (i.e. the type(5,6,0,1)) are constructed. It turns out that there are exactly 2 OM(12,9,6;6) of 3-type.

In the following the matrices of 2-type (i.e. the type(6,3,3)) are constructed. A detailed analysis shows that there are exactly 2 OM(12,9,6;6) of 2-type.

4 Configurations and semibiplanes

Symmetric configurations are defined as follows (For further definitions and references see [5] and other papers of the author.)

Definition 4 *A configuration v_k consists of v points and v lines such that (1) each line contains k points, (2) each point lies on k lines, and*

(3) two different points are connected by at most one line.

Like orbital matrices, configurations are generalizations of symmetric 2-designs. Moreover, they can occur as substructures of orbital matrices by identifying certain entries in the orbital matrix (compare [6], page 121).

References

[1] R.H. Bruck, H.J. Ryser, The nonexistence of certain finite projective planes, Canadian Journal of Math. 1 (1949), 88-93.

[2] S. Chowla, H.J. Ryser, Combinatorial problems, Canadian Journal of Math. 2 (1950), 93-99.

[3] R. Craigen, Hadamard matrices and designs, in The CRC Handbook of Combinatorial Designs, (eds.) C.J. Colbourn, J.H. Dinitz, Boca Raton (1996), 370-377.

[4] R. Craigen, Weighing matrices and conference matrices, in The CRC Handbook of Combinatorial Designs, (eds.) C.J. Colbourn, J.H. Dinitz, Boca Raton (1996), 496-504.

[5] H. Gropp, Configurations, in The CRC Handbook of Combinatorial Designs, (eds.) C.J. Colbourn, J.H. Dinitz, Boca Raton (1996), 253-255.

[6] H. Gropp, On orbital matrices, in: K. Denecke, O. Lüders (eds.), General Algebra and Applications in Discrete Math., Aachen (1997), 111-124.

[7] M.P. Schützenberger, A non-existence theorem for an infinite family of symmetrical block designs, Annals of Eugenics 14 (1949), 286-287.

Discrepancy of Sums of Arithmetic Progressions

Nils Hebbinghaus [1]

Mathematisches Seminar, Bereich II,
Christian-Albrechts-Universität zu Kiel,
Christian-Albrechts-Platz 4, 24118 Kiel, Germany

1 Introduction

Let V be a finite set and $\mathcal{E}$ a subset of 2^V. Then $\mathcal{H} := (V, \mathcal{E})$ is called a *hypergraph*. The elements of V are called *vertices* and those of $\mathcal{E}$ *hyperedges*. The discrepancy problem is to color the vertices with two colors, such that all hyperedges are colored as balanced as possible. The *discrepancy* of $\mathcal{H}$ is a measure for the deviation from an optimal distribution and is defined by

$$\operatorname{disc}(\mathcal{H}) = \min_{\chi : V \to \{1,-1\}} \max_{E \in \mathcal{E}} \left| \sum_{x \in E} \chi(x) \right|,$$

For further information on discrepancies, we refer to Beck and Sós [BS95] and Matoušek [Mat99].
Let N be a positive integer, $[N] := \{0, 1, 2, \ldots, N-1\}$ and $\mathcal{H}_N = ([N], \mathcal{E}_N)$ the hypergraph of all arithmetic progressions in $[N]$. That means $\mathcal{E}_N$ contains all sets of the form $\{\{a + k\delta \mid k \in [L]\} \mid a, \delta, L \in \mathbb{N}\}$ in $[N]$. Determing the discrepancy of the hypergraph of arithmetic progressions was a long standing open problem. Although he did not use the discrepancy notion explicitly, Roth proved $\operatorname{disc}(\mathcal{H}_N) = \Omega(N^{1/4})$ [R64]. Roth himself thought that this lower bound should be improvable to $\Omega(N^{1/2-\epsilon})$. But in 1974, Sárközy disproved this hypothesis by giving an upper bound of $O(N^{1/3} \log^{1/3} N)$. A sketch of his beautiful proof can be found in [ES74]. It was Beck, who showed in 1981 that Roth's lower bound is nearly optimal. His upper bound of $O(N^{1/4} \log^{5/2} N)$ was improved in 1996 by Matousek and Spencer [MS96] removing the logarithmic factor. Thus $\operatorname{disc}(\mathcal{H}_N)$ is of order $\Theta(N^{1/4})$. Therefore this classical discrepancy problem was solved. But it is an interesting question whether the methods used for the hypergraph of arithmetic progressions can be used for the discrepancy problem of related hypergraphs, or new methods are needed. For example Doerr, Srivastav and Wehr [DSW] investigated the hypergraph of the d-dimensional arithmetic progressions $\mathcal{H}_{N,d} = ([N]^d, \mathcal{E}_{N,d})$ $(d \geq 1)$, where the set of hyperedges is $\mathcal{E}_{N,d} := \{A_1 \times \ldots \times A_d \mid A_i$ arithmetic progression in $[N]\}$. They proved $\operatorname{disc}(\mathcal{H}_{N,d}) = \Theta(N^{d/4})$. Valkó [V2002] determined the discrepancy of the hypergraph $\mathcal{H}'_{N,d}$ of one-dimensional arithmetic progressions in $[N]^d$. He showed $\operatorname{disc}(\mathcal{H}'_{N,d}) = \Omega(N^{d/(2d+2)})$ and

[1] E-mail: `nhe@numerik.uni-kiel.de`. The author thanks the DFG-Graduiertenkolleg 357 "Effiziente Algorithmen und Mehrskalenmethoden" for supporting this research.

$\operatorname{disc}(\mathcal{H}'_{N,d}) = O(N^{d/(2d+2)} \log^{5/2} N)$.
In this paper we are interested in the discrepancy of the hypergraph $\mathcal{H}_{N,k} = ([N], \mathcal{E}_{N,k})$, where $\mathcal{E}_{N,k}$ consists of all sums of k arithmetic progressions for a positive integer k. Let $\mathcal{A}$ be the set of all arithmetic progressions in $\mathbb{Z}$. Then the set of hyperedges is

$$\mathcal{E}_{N,k} := \{(A_1 + A_2 + \dots A_k) \cap [N] \mid A_i \in \mathcal{A}\}.$$

Our main result is the following

Theorem 1 *For all positive integers k we have* $\operatorname{disc}(\mathcal{H}_{N,k}) = \Omega(N^{\frac{k}{2k+2}})$.

$\mathcal{H}_{N,1}$ is the hypergraph of arithmetic progressions and we get Roth's lower bound. Note that the probabilistic method [AS00] gives an upper bound of $O(N^{1/2} \log^{1/2} N)$ for $\operatorname{disc}(\mathcal{H}_{N,k})$.

2 Discrepancy of $\mathcal{H}_{N,k}$

The structure of the hyperedges of $\mathcal{H}_{N,k}$ is not that regular as that of arithmetic progressions. For instance in the sum of two or more arithmetic progressions some elements can have several ways to be expressed as sum of elements of the arithmetic progressions. Thus we cannot apply Fourier analysis in a direct way, but have to look for hyperedges of $\mathcal{H}_{N,k}$, which do not have those ambiguities. We are calculating a lower bound for the discrepancy of the subhypergraph containing only this special hyperedges, which is of course also a lower bound for $\operatorname{disc}(\mathcal{H}_{N,k})$.
For convenience we assume that $2^{k-1} | N^{1/(k+1)}$. The Bertrand postulate (or Chebyshev's theorem) states the existence of prime numbers p_i for all $i \in \{1, 2, \dots k-1\}$ with $2^{i-k+1} N^{1/(k+1)} < p_i < 2^{i-k+2} N^{1/(k+1)}$. Every sum of k arithmetic progressions is characterised by a starting point, a k-tuple $\delta = (\delta_1, \delta_2, \dots, \delta_k)$ of differences and a k-tuple $L = (L_1, L_2, \dots, L_k)$ which fixes the length of the k arithmetic progressions. At first we define the length of the i-th arithmetic progression ($i \in \{1, 2, \dots k\}$) by

$$L_i := 2^{i-k-1} N^{\frac{1}{k+1}}.$$

With $\widetilde{\Delta} := \prod_{i=1}^{k} \{1, 2, \dots, 2^{i-k} N^{1/(k+1)}\}$ we define a set Δ of k-tuples of differences by

$$\Delta := \{(\delta_1, \delta_2, \dots, \delta_k) \mid \delta_i = \prod_{j=1}^{i} \widetilde{\delta_j} \prod_{j=i}^{k-1} p_j, 1 \le i \le k, (\widetilde{\delta_1}, \widetilde{\delta_2}, \dots, \widetilde{\delta_k}) \in \widetilde{\Delta}\}.$$

For all $j \in \mathbb{Z}$ and all $\delta = (\delta_1, \delta_2, \dots, \delta_k) \in \Delta$ we set

$$A_{j,\delta} := \{j + \sum_{i=1}^{k} a_i \delta_i \mid a_i \in [L_i], 1 \le i \le k\}.$$

The hyperedges $A_{j,\delta}$ with $j \in \{-N, -N+1, \ldots, N-1\}$ and $\delta \in \Delta$ are bilding the subhypergraph mentioned above.

2.1 Fourier analysis

For the proof of Theorem 1 we need some well known facts about Fourier analysis. Here we restrict ourselves to the Fourier analysis on $\mathbb{Z}$. Let $f : \mathbb{Z} \to \mathbb{C}$ be a function in $L^1(\mathbb{Z})$. The Fourier transform $\hat{f}$ of f is

$$\hat{f} : [0,1] \to \mathbb{C}, \quad \alpha \mapsto \sum_{z \in \mathbb{Z}} f(z) e^{-2\pi i z \alpha}.$$

For two functions $f, g \in L^1(\mathbb{Z})$ the convolution of f and g is defined by $(f * g)(y) = \sum_{x \in \mathbb{Z}} f(x) g(y-x)$ for all $y \in \mathbb{Z}$. For the Fourier transform of $f * g$ it holds

$$\widehat{f * g} = \hat{f}\hat{g}.$$

In this special situation the Plancherel Theorem gives for all $f \in (L^1 \cap L^2)(\mathbb{Z})$

$$\|\hat{f}\|_2^2 = \|f\|_2^2.$$

2.2 The proof

Let us fix a 2-coloring $\chi : V \to \{-1, 1\}$ and define the natural extension χ_e of χ to $\mathbb{Z}$

$$\chi_e(j) = \begin{cases} \chi(j)\,, & \text{if } j \in V, \\ 0\,, & \text{otherwise.} \end{cases}$$

For all $i \in \{1, 2, \ldots k\}$ let $\eta_{\delta_i} : \mathbb{Z} \to \{0, 1\}$ be defined by

$$\eta_{\delta_i}(j) := \begin{cases} 1\,, & \text{if } -j \in \delta_i[L_i], \\ 0\,, & \text{otherwise.} \end{cases}$$

The function $\eta_\delta := \eta_{\delta_1} * \eta_{\delta_2} * \ldots * \eta_{\delta_k}$ is an indicator function for $-A_{0,\delta}$ as the following lemma states.

Lemma 1 *Let $\delta \in \Delta$ and $x \in \mathbb{Z}$. Then*

$$\eta_\delta(x) = \begin{cases} 1\,, & \textit{if } -x \in A_{0,\delta}, \\ 0\,, & \textit{else.} \end{cases}$$

The proof of Lemma 1 can be done by induction over k using the structur of the set Δ of differences. With η_δ and the extended coloring χ_e of the hypergraph $\mathcal{H}_{N,k}$ we are able to express the coloring of a given hyperedge $\chi(A_{j,\delta}) := \sum_{x \in A_{j,\delta}} \chi_e(x)$ in the following way.

Lemma 2 *For all $j \in \mathbb{Z}$ and all $\delta \in \Delta$ it holds $\chi(A_{j,\delta}) = (\chi_e * \eta_\delta)(j)$.*

This equality, which is easy to calculate, is the connection between the Fourier analysis and our discrepancy problem. The next lemma gives an estimation for exponential sums that we will use later on for the proof of Theorem 1.

Lemma 3 *Let $\alpha \in \mathbb{R} \setminus \mathbb{Q}$, $L \in \mathbb{N}$. There exists a $\delta \in \{1, 2, \ldots, 2L\}$ with*

$$\left| \sum_{j=0}^{L-1} e^{2\pi i \delta j \alpha} \right|^2 \geq \left(\frac{2}{\pi} L \right)^2 .$$

The next step in the proof of Theorem 1 is to show that for a constant $c_1 > 0$ and all $\alpha \in (0,1) \setminus \mathbb{Q}$ it holds

$$\sum_{\delta \in \Delta} |\hat{\eta}_\delta|^2 \geq c_1 N^{2k/(k+1)} .$$

Using this estimation, Lemma 2 and the Plancherel Theorem one can calculate

$$\sum_{\delta \in \Delta} \sum_{j=-N}^{N-1} \left(\chi\left(A_{j,\delta} \right) \right)^2 \geq c_1 N^{\frac{3k+1}{k+1}} .$$

Hence there exist a $\delta \in \Delta$ and a $j \in \{-N, \ldots, N-1\}$ and a constant $c_2 > 0$ with

$$|\chi(A_{j,\delta})| \geq \left(\frac{c_1}{2N|\Delta|} N^{\frac{3k+1}{k+1}} \right)^{\frac{1}{2}} \geq c_2 N^{\frac{k}{2k+2}},$$

and our assertion follows.

3 Conclusion

We have studied the discrepancy of the hypergraph of sums of k arithmetic progressions. Our main result is the lower bound of $\Omega(N^{k/(2k+2)})$ for this hypergraph. Here $k = 1$ gives back Roth's lower bound for the hypergraph of arithmetic progressions. For the proof of this lower bound we used Fourier analysis on $\mathbb{Z}$ and a special set of hyperedges. There is a large gap between this lower bound and the upper bound of $O(N^{1/2} \log^{1/2}(N))$, which is easily derived by the probabilistic method [AS00]. Our believe is that - as in the situation of the arithmetic progressions - the probabalistic method tells not the whole truth and a better upper bound can be found.

References

[AS00] N. Alon, J. Spencer, and P. Erdős. *The Probabilistic Method* (Second Edition). John Wiley & Sons, Inc., Ney York, 2000.

[B81] J. Beck. Roth's Estimate of the Discrepancy of Integer Sequences is Nearly Sharp. *Combinatorica* **1**(4) (1981), 319-325.

[BS95] J. Beck and V. T. Sós. *Discrepancy theory.* In R. Graham, M. Gröschel and L. Lovász, editors, Handbook of Combinatorics, 1405-1446. 1995.

[DSW] B. Doerr, A. Srivastav, P. Wehr. Discrepancies of Cartesian Products of Arithmetic Progressions. *Electronic Journal of Combinatorics*, to appear.

[ES74] P. Erdős, J. Spencer. *Probibalistic Methods in Combinatorics.* Akademiai Kiado, Budapest, 1974.

[Mat99] J. Matoušek. *Geometric Discrepancy.* Springer, Berlin, 1999.

[MS96] J. Matoušek, J. Spencer. Discrepancy in Arithmetic Progressions. *Journal of the American Mathematical Society* **9**(1) (1996), 195-204.

[R64] K. F. Roth. Remark Concerning Integer Sequences. *Acta Arithmetica* **9** (1964), 257-260.

[V2002] B. Valkó. Discrepancy of Arithmetic Progressions in Higher Dimensions. *Journal of Number Theory* **92** (2002), 117-130.

Computing Optimal Discrete Morse Functions

Michael Joswig [1]

Institut für Mathematik, MA 6–2, TU Berlin, Germany

Marc E. Pfetsch [2]

Zuse Institute Berlin, Germany

Abstract

The essential structural information of discrete Morse functions is captured by so-called Morse matchings. We show that computing optimal Morse matchings is $\mathcal{NP}$-hard and give an integer programming formulation for the problem. Then we present first polyhedral results for the corresponding polytope and report on some preliminary computational results.

1 Introduction

Discrete Morse functions where introduced by Forman [3] as a combinatorial analogy of classical smooth Morse theory and have many applications in combinatorial topology, e.g., they can be used to compute a compact representation of a simplicial complex as an CW-complex; for details and other applications see [3], Chari [1], and Joswig [6]. It turns out that the essential information of discrete Morse functions can be stored in a *Morse matching.* To be concise, we will therefore not give the definition of discrete Morse functions but state everything in terms of Morse matchings. In the applications one is interested in optimal Morse matchings, a problem which leads to a combinatorial optimization problem that we will describe in the following.

We first need some notation. Let Δ be a *(finite abstract) simplicial complex*, i.e., a set of subsets of a finite set V with the following property: if $F \in \Delta$ and $G \subseteq F$, then $G \in \Delta$; hence Δ is an independence system with ground set V. In the following we will ignore $\varnothing$ as a member of Δ. The elements in V are called *vertices* and the elements of Δ are called *faces.* The *dimension* of a face F is $\dim F := |F| - 1$. In the following let $d = \max\{\dim F \,:\, F \in \mathcal{F}\}$ be the dimension of Δ. Let $\mathcal{F}$ be the set of faces of Δ and let $f_i = f_i(\Delta)$ be the number

[1] E-mail: `joswig@math.tu-berlin.de`.
[2] E-mail: `pfetsch@zib.de`.

of faces of dimension $i \geq 0$. The maximal faces with respect to inclusion are called *facets* and Δ is *pure*, if all facets have the same dimension.

Consider the *Hasse diagram* $H = (\mathcal{F}, A)$ of Δ, that is, a directed graph on the faces of Δ with an arc $(F, G) \in A$ if $G \subset F$ and $\dim G = \dim F - 1$. It will be convenient not to distinguish between H and its underlying undirected graph, i.e., when we speak of matchings and (undirected) cycles we mean the corresponding structures in the underlying undirected graph.

Let $M \subset A$ be a matching in H and let $H(M)$ be the directed graph obtained from H by reversing the direction of the arcs in M. Then M is a *Morse matching* of Δ if $H(M)$ does not contain directed cycles, i.e., is acyclic (in the directed sense). Chari [1] showed that the essential structure of discrete Morse functions are contained in Morse matchings. As stated above, one is interested in maximum Morse matchings, i.e., the size of M is maximized. The complementary measure to $|M|$ is the number of *critical faces* of M, i.e., faces not matched by M. Hence, by maximizing $|M|$, we minimize the number of critical faces.

It seems helpful to briefly describe the case of Morse matchings for a one-dimensional simplicial complex Δ. Then Δ represents the incidences of a graph G. A Morse matching M of Δ matches edges with nodes of G. Let $\tilde{G}$ be the following oriented subgraph of G: take all edges which are matched in M and orient them towards its matched node. Since M is a matching this construction is well defined and the in-degree of each node is one. The acyclicity property shows that $\tilde{G}$ contains no directed cycles and hence is a branching. Therefore, the Morse matchings on a graph G are in one-to-one correspondence with orientations of subgraphs of G which are branchings. Generalizing this idea, Lewiner, Lopes, and Tavares [7] developed a heuristic for computing optimal Morse matchings, which works well for the data set which we also use in Section 4.

In the following we will show that computing optimal Morse matchings is $\mathcal{NP}$-hard. Then we will give an integer programming formulation for the problem and sketch polyhedral results for the corresponding polytope. We end with some preliminary computational results.

2 Hardness of Computing an Optimal Morse Matching

Eğecioğlu and Gonzalez [2] proved a hardness result which in terms of Morse matchings reads as follows: Given a pure 2-dimensional simplicial complex Δ and an integer K, it is $\mathcal{NP}$-complete to decide whether there exist a Morse matching with at most K critical 2-faces, i.e., faces of dimension 2. In fact we can remove the "restriction" to 2-faces and prove:

Theorem 1 *Given a simplicial complex Δ and an integer K, it is $\mathcal{NP}$-complete to decide whether there exists a Morse matching with at most K critical simplices.*

This result holds even when Δ is connected (i.e., H is connected), pure, 2-dimensional, and it can be embedded in $\mathbb{R}^3$. A crucial part in the proof of this theorem is the following lemma:

Lemma 1 *Given any Morse matching M on Δ, we can compute a Morse matching M' which has exactly one critical vertex and at most as many critical 2-faces as M.*

This lemma and the Euler equation make it possible to reduce the general case to the problem discussed by Eğecioğlu and Gonzalez. In fact, they proved strong inapproximability results for their problem. Lewiner, Lopes, and Tavares [7] claimed the same inapproximability results for computing Morse matchings with a minimum number of critical faces, but did not supply a reasoning as in the lemma. The reduction in our proof is not approximation preserving. Therefore, the approximability status seems to be open; the same holds for computing maximum Morse matchings.

3 An IP-formulation

In this section we will discuss an integer programming formulation of the problem to compute a maximum Morse matching. We introduce a binary variable x_a for every arc in H, where $x_a = 1$ if and only if a should be reversed in a Morse matching. The matching conditions are modeled by:

$$\boldsymbol{x}(\delta(F)) := \sum_{a \in \delta(F)} x_a \leq 1 \qquad \forall\, F \in \mathcal{F}. \tag{1}$$

To handle the acyclicity requirement, let M be a Morse matching and assume C to be a directed cycle in $H(M)$. Because of the matching property, the nodes in C can only belong to two levels in the Hasse diagram, i.e., $\{\dim F \,:\, F \in C\} = \{i, i{+}1\}$ for some $i \in \{0, \ldots, d{-}1\}$. Therefore define H_i to be the subgraph of H induced by the faces of dimension i and $i+1$, for $i \in \{0, \ldots, d-1\}$. Again by the matching property, the values x_a for the arcs in C alternate. A little thought reveals that the following constraints suffice to eliminate directed cycles:

$$\boldsymbol{x}(C) := \sum_{a \in C} x_a \leq \frac{|C|}{2} - 1 \qquad \forall\, C \text{ cycle in } H_i,\ i = 0, \ldots, d-1. \tag{2}$$

Hence, the convex hull of all incidence vectors of Morse matchings is the following polytope:

$$P_M := \Big\{\boldsymbol{x} \in \{0,1\}^A \,:\, \boldsymbol{x} \text{ satisfies (1) and (2)}\Big\}.$$

A Morse matching with incidence vector $\boldsymbol{x} \in P_M$ has $|\mathcal{F}| - 2\,\boldsymbol{x}(A)$ critical faces. The problem to compute an optimal Morse matching is then to solve $\max\{\boldsymbol{x}(A) \,:\, \boldsymbol{x} \in P_A\}$.

It is easy to see that P_M is a monotone, full dimensional polytope and that $x_a \geq 0$ defines a facet for every $a \in A$. Let us remark that the incidence vectors of Morse matchings do not have to be monotone if H is an arbitrary acyclic digraph. We can prove the following results:

Proposition 1 *The matching constraints $\boldsymbol{x}(\delta(F)) \leq 1$ define facets of P_M for $F \in \mathcal{F}$, except if $|\delta(F)| \leq 1$.*

It follows that the inequalities $x_a \leq 1$, $a \in A$, never define facets.

Proposition 2 *The cycle constraints* (2) *define facets of P_M and can be separated in polynomial time.*

Some of the features of our problem resemble the acyclic subgraph problem (ASP), studied by Grötschel, Jünger, and Reinelt [4]. The separation algorithm referred to in Proposition 2, however, is more complicated than the one for ASP, since the usual affine transformation trick ($\boldsymbol{x}' = \mathbb{1} - \boldsymbol{x}$) to turn the separation problem into a shortest cycle problem does not work in our case.

One can strengthen the LP relaxation considerably by adding so called *Morse inequalities*, which say that the number of critical faces of dimension i is at least the Betti number β_i, see Forman [3]. This translates to the inequality $\sum_{F \in \mathcal{F}_i} \boldsymbol{x}(\delta(F)) \leq f_i - \beta_i$.

4 Computational Results

We performed preliminary computational experiments with a branch-and-cut code for the above integer programming formulation. The algorithm was implemented using the branch-and-cut-and-price framework SCIP, developed by Tobias Achterberg at the Zuse Institute Berlin. We computed Morse matchings for the smaller problems in a collection of simplicial complexes maintained by Hachimori [5]. As a primal heuristic we used a simple greedy algorithm. Whenever possible we branched as follows: for a face $F \in \mathcal{F}$, we branch on the following three constraints: $\boldsymbol{x}(\delta^-(F)) = 1$, $\boldsymbol{x}(\delta^+(F)) = 1$, $\boldsymbol{x}(\delta(F)) = 0$; this seems to work very well. Additionally, we added Gomory cuts.

Computing optimal Morse matchings in practice appears to be hard for relatively small problems. The reason seems to be the high symmetry of the problems and the weakness of the LP relaxation. One has a good chance, however, if the absolute difference between the optimal value and the bounds implied by the Morse inequalities is small. In fact, for many of the problems in Hachimori's collection this difference is 0 and the algorithm "only" has to find the optimal primal solution, which it usually finds fast. Summarizing, our code can solve all 10 problems in the collection with up to 160 arcs in the Hasse diagram (and two larger ones) in about an hour; about 50% of these problems are solved in a few seconds.

It is clear, that there are still many things to investigate. Our plan for the future is to find other (facet defining) inequalities for P_A that can help to improve the dual bound. Furthermore, it seems interesting to check whether local search methods can help to improve primal solutions.

References

[1] M. K. Chari, *On discrete Morse functions and combinatorial decompositions*, Discrete Math., 217 (2000), pp. 101–113.

[2] Ö. Eğecioğlu and T. F. Gonzalez, *A computationally intractable problem on simplicial complexes*, Comut. Geom., 6 (1996), pp. 85–98.

[3] R. Forman, *Morse theory for cell-complexes*, Advances in Math., 134 (1998), pp. 90–145.

[4] M. Grötschel, M. Jünger, and G. Reinelt, *On the acyclic subgraph polytope*, Math. Program., 33 (1985), pp. 28–42.

[5] M. Hachimori, *Simplicial complex library.* `http://infoshako.sk.tsukuba.ac.jp/~hachi/math/library/index_eng.html`, 2001.

[6] M. Joswig, *Computing invariants of simplicial manifolds.* Preprint, arXiv: math.AT/0401176, 2003.

[7] T. Lewiner, H. Lopes, and G. Tavares, *Towards optimality in discrete Morse theory*, Exp. Math., to appear (2003).

Minimum-Cost Single-Source 2-Splittable Flow

Stavros G. Kolliopoulos[1]

Abstract

In the *single-source unsplittable flow* problem, commodities must be routed simultaneously from a common source vertex to certain sinks in a given graph with edge capacities and costs. The demand of each commodity must be routed along a single path so that the total flow through any edge is at most its capacity. Moreover the cost of the solution should not exceed a given budget. An important open question is whether a simultaneous $(2, 1)$-approximation can be achieved for minimizing congestion and cost, i.e., the budget constraint should not be violated. In this note we show that this is possible for the case of 2-splittable flows, i.e., flows where the demand of each commodity is routed along at most two paths.

Key words: Approximation Algorithms; Unsplittable Flow; Maximum Flow.

1 Introduction

In the *single-source b-splittable flow* problem we are given a positive integer b, a directed graph $G = (V, E)$ with edge capacities $u : E \to \mathbb{R}^+$ and edge costs $c : E \to \mathbb{R}^+$, a budget $B > 0$, a designated source vertex $s \in V$, and k commodities each with a sink vertex $t_i \in V$ and associated demand $d_i > 0$, $i = 1, \ldots, k$. A feasible b-splittable flow routes for each i, d_i units of commodity i along at most b paths from s to t_i so that the total flow through an edge e is at most its capacity u_e. As is standard in the relevant literature we assume that no edge can be a bottleneck, i.e., the minimum edge capacity is assumed to have value at least $\max_i d_i$. The cost $c(f)$ of flow f is given by $c(f) = \sum_{e \in E} c_e f_e$. The cost $c(P_i)$ of a path P_i is defined as $c(P_i) = \sum_{e \in P_i} c_e$. We seek a feasible b-splittable flow whose total cost does not exceed the budget. When $b = 1$ we obtain the well-studied *single-source unsplittable flow problem* (UFP). The cost of an unsplittable flow f given by paths $P_1, \cdots, P_k$ can also be written as $c(f) = \sum_{i=1}^{k} d_i \cdot c(P_i)$. The feasibility question for UFP is strongly NP-complete [4] even without a budget constraint. An optimization version which has attracted considerable attention is to *minimize congestion:* Find the smallest $\alpha \geq 1$ such that there exists a feasible unsplittable flow if all capacities are multiplied by α.

[1] Department of Computing and Software, McMaster University, Hamilton, Ontario, L8S 4K1, Canada (`stavros@mcmaster.ca`). Research partially supported by NSERC Grant 227809-00.

A relaxation of b-splittable flow, $b \geq 1$, is obtained by allowing the demands of commodities to be split along an arbitrary number of paths; this yields a standard maximum flow problem. We will call a solution to this relaxation, a *fractional flow.* In this note we study the simultaneous approximation of congestion and cost for the single-source 2-splittable flow problem. This corresponds to the strictest possible relaxation of UFP as far as the usage of paths is concerned. We slightly abuse terminology and view such flows as 2-splittable solutions to UFP .

UFP was introduced by Kleinberg [4] and contains several well-known NP-complete problems as special cases: Partition, Bin Packing, scheduling on parallel machines to minimize makespan [4]. In addition UFP generalizes single-source edge-disjoint paths and models aspects of virtual circuit routing. The first constant-factor approximations were given in [5]. Kolliopoulos and Stein [6] gave a 3-approximation algorithm for congestion with a simultaneous performance guarantee of 2 for cost, which we denote as a $(3, 2)$-approximation. Dinitz, Garg, and Goemans [2] improved the congestion bound to 2 although their algorithm cannot handle the budget constraint. To be more precise, their basic result is that any splittable flow satisfying all demands can be turned into an unsplittable flow while increasing the total flow through any edge by less than the maximum demand. This result is tight if the congestion achieved by the fractional flow is used as a lower bound [2]. Skutella [8] obtained a $(3, 1)$-approximation algorithm and this is currently the best bicriteria bound. Various results on b-splittable flows were obtained by Baier, Köhler and Skutella in [1]. The main focus of their paper is that of finding a maximum value b-splittable s-t flow. Finding a feasible solution to UFP reduces to solving optimally the latter problem but there does not seem to be a further connection with the formulation we examine. As Baier et al. observe [1] the analysis of any algorithm that uses as lower bounds the optima of a fractional flow applies for the single-source b-splittable flow problem. Therefore the result of [8] extends to a $(3, 1)$-approximation for the 2-splittable case.

In terms of negative results, Erlebach and Hall [3] prove that for UFP and arbitrary $\varepsilon > 0$ there is no $(2 - \varepsilon, 1)$-approximation algorithm for congestion and cost unless $P = NP$. Matching this bicriteria lower bound is an important open question that has attracted a lot of attention. Such a $(2, 1)$-approximation is known only for the scheduling problem $R|p_{ij} = p_j$ or $\infty|C_{\max}$ with assignment costs [7]. This scheduling problem reduces to a UFP instance on a 2-level graph where minimizing congestion is equivalent to minimizing the makespan.

In this note we show that the simultaneous $(2, 1)$-approximation for congestion and cost can be obtained for the single-source 2-splittable flow problem. The precise bound we achieve is given in Theorem 2. Better bounds can be achieved for b-splittable solutions with $b > 2$; we omit the details.

2 The algorithm for 2-splittable flow

We set $d_{\max} = \max_{1 \le i \le k} d_i$, $d_{\min} = \min_{1 \le i \le k} d_i$ for the instance of interest. We assume without loss of generality that there is a feasible fractional solution f_0 for the given UFP instance I and that $d_{\max} \le 1$. The cost of the final splittable solution will not exceed the cost of the initial fractional solution f_0. If the latter solution is a minimum-cost flow we obtain a best possible budget. The following result of Kolliopoulos and Stein [6] will be of use.

Theorem 1 [6] *Given a* UFP *instance where all demands are powers of* $1/2$ *and an initial fractional flow solution, there is an algorithm, called* POWER-ALG *which finds an unsplittable flow* f *that violates the capacity of any edge by at most* $d_{max} - d_{min}$ *and whose cost is bounded by the cost of the initial fractional flow.*

To solve the 2-splittable flow problem one can naively break a commodity into two equal demands and run a UFP algorithm. The Skutella algorithm would thus provide a $(2.5, 1)$-approximation. We will break every demand d_i $i = 1, \dots, k$ into two commodities with demands a_i and b_i s.t. $a_i + b_i \le d_i$, and both a_i and b_i are powers of $1/2$. In a repairing stage at the end we will ensure that all of d_i is routed.

Let $floor_2(x)$ denote the largest number which is a power of $1/2$ and does not exceed x. We define the following operator $\lfloor \cdot \rfloor_2$

$$\lfloor x \rfloor_2 = \begin{cases} \text{floor}_2(x) & \text{if } x < 1 \\ 1/2 & \text{if } x = 1 \end{cases}$$

Set $a_i \doteq \lfloor d_i \rfloor_2$ and $b_i \doteq \lfloor d_i - a_i \rfloor_2$. Create a new UFP instance I^2 with $2k$ commodities where commodity i of I is mapped to two commodities with demands a_i, b_i. Run the POWER-ALG of Theorem 1 on I^2 to obtain a flow f. Observe that $x \le y \Rightarrow \lfloor x \rfloor_2 \le \lfloor y \rfloor_2$.

Lemma 1 *Given the* UFP *instance* I^2 *with initial fractional solution* f_0^2 *one can find an unsplittable flow* f *(i) violates the capacity of any edge by at most* $\lfloor d_{max} \rfloor_2 - \lfloor d_{min} \rfloor_2$ *and (ii) whose cost is bounded by the cost of the initial fractional flow* f_0^2. *Flow* f *corresponds to a* 2*-splittable flow for instance* I *which routes* $a_i + b_i$ *units of flow for commodity* i, $i = 1, \dots, k$.

The task that remains is to transform the flow f of Lemma 1 so that d_i units of flow are routed for commodity i. By the definition of a_i and b_i we have that $a_i \ge \frac{d_i}{2}$ and if $b_i > 0 \Rightarrow b_i \ge \frac{d_i}{4}$. Moreover $b_i = 0$ only when $a_i = d_i$. Therefore we always have that

$$a_i + b_i \ge \frac{3d_i}{4}, \quad i = 1, \dots, k.$$

We obtain a flow f' from f by scaling the flow on each of the at most two s-t_i paths used in f *by the same amount* $\lambda_i \in (1, 4/3)$ so that

$$\lambda_i(a_i + b_i) = d_i.$$

This transformation yields a 2-splittable flow f' which (i) satisfies all demands d_i and (ii) satisfies $f'_e \leq (4/3)u_e + (4/3)\left(\lfloor d_{\max} \rfloor_2 - \lfloor d_{\min} \rfloor_2\right)$ for all edges $e \in E$. We distinguish two cases.

Case 1. $d_{\max} \geq 1/2$. Then $\lfloor d_{\max} \rfloor_2 = 1/2$. Therefore $f'_e \leq (4/3)u_e + (2/3) - \lfloor d_{\min} \rfloor_2$.

Case 2. $d_{\max} < 1/2$. Then $\lfloor d_{\max} \rfloor_2 \leq 1/4$. Therefore $f'_e \leq (4/3)u_e + (1/3) - \lfloor d_{\min} \rfloor_2$.

We have shown the following lemma.

Lemma 2 *Given a* UFP *instance with initial fractional solution f_0 one can find a 2-splittable flow f' such that (i) f' satisfies all demands d_i, $i = 1, \ldots, k$ (ii) $f'_e \leq (4/3)u_e + (2/3) - \lfloor d_{min} \rfloor_2$ for all $e \in E$ and (iii) the cost of f' is bounded by at most 4/3 times the cost of the initial fractional flow f_0.*

To obtain the bound on the cost we define carefully the fractional solution for the instance I^2 using a method proposed by Skutella [8]. The fractional solution f_0^2 is obtained from f_0 as follows: for the commodities for which $d_i = a_i + b_i$ f_0^2 sends flow as f_0. For the remaining commodities we decrease the flow by $\bar{d}_i = d_i - (a_i + b_i)$ along the most expensive, i.e., higher cost, $s - t_i$ paths [8]. To be precise, let P^i be the set of paths from s to t_i that carry nonzero flow in f_0. Order them in order of nonincreasing cost $P_1^i, P_2^i, \ldots$ and let $f_1^i, f_2^i, \ldots$ be the corresponding flow amounts. Remove $\gamma_i = \min\{f_1^i, \bar{d}_i\}$ units of flow from P_1^i and decrease $\bar{d}_i$ by γ_i. Repeat on the next path in P^i with nonzero flow until $\bar{d}_i = 0$. The ordering of the paths based on nonincreasing cost can be implemented in polynomial time because we can assume without loss of generality that f_0 does not send flow along cycles [8]. After this preprocessing step $c(f_0^2) \leq c(f_0)$. By Lemma 1, $c(f) \leq c(f_0^2)$. Inspection of the POWER-ALG yields that the paths used in the unsplittable solution must be paths with nonzero flow in the initial fractional solution. Therefore all the paths from s to t_i that carry nonzero flow in f have cost less than or equal to the cost of the paths on which flow was decreased while obtaining f_0^2 from f_0. Routing $d_i - (a_i + b_i)$ additional units from s to t_i along the at most two paths used in f results in a solution f' for which $c(f') \leq c(f_0)$. The main result has been proved.

Theorem 2 *Given a* UFP *instance I with initial fractional solution f_0 one can find in polynomial time a 2-splittable flow f' such that (i) f' satisfies all demands d_i, $i = 1, \ldots, k$ (ii) $f'_e \leq (4/3)u_e + (2/3) - \lfloor d_{min} \rfloor_2$ for all $e \in E$ and (iii) the cost of f' is bounded by the cost of the initial fractional flow f_0.*

Corollary 2.1 *Given a* UFP *instance I one can find in polynomial time a 2-splittable flow solution that achieves a simultaneous $(2, 1)$-approximation for congestion and cost.*

References

[1] G. Baier, E. Köhler, and M. Skutella. On the k-splittable flow problem. To appear in Algorithmica, special issue on ESA 2002.

[2] Y. Dinitz, N. Garg, and M. X. Goemans. On the single-source unsplittable flow problem. *Combinatorica*, 19:1–25, 1999.

[3] T. Erlebach and A. Hall. NP-hardness of broadcast sheduling and inapproximability of single-source unsplittable min-cost flow. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[4] J. M. Kleinberg. *Approximation algorithms for disjoint paths problems.* PhD thesis, MIT, Cambridge, MA, May 1996.

[5] J. M. Kleinberg. Single-source unsplittable flow. In *Proc. 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 68–77, 1996.

[6] S. G. Kolliopoulos and C. Stein. Approximation algorithms for single-source unsplittable flow. *SIAM Journal on Computing*, 31:919–946, 2002.

[7] D. B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming A*, 62:461–474, 1993.

[8] M. Skutella. Approximating the single-source unsplittable min-cost flow problem. *Mathematical Programming B*, 91:493–514, 2002.

The Kissing Number Problem: A new result from global optimization

Leo Liberti[1]

DEI, Politecnico di Milano, P.zza L. da Vinci 32, 20133 Milano, Italy

Nelson Maculan[2]

COPPE, Universidade Federal do Rio de Janeiro, P.O. Box 68511, 21941-972 Rio de Janeiro, Brazil

Sergei Kucherenko[3]

CPSE, Imperial College, SW7 2AZ, London, UK

Abstract

Determining the maximum number of D-dimensional spheres of radius r that can be adjacent to a central sphere of radius r is known as the Kissing Number Problem (KNP). The problem has been solved for 2 and 3 dimensions. The smallest open case is 4 dimensions: a solution with 24 spheres is known, and an upper bound of 25 has been found. We present a new nonlinear mathematical programming model for the solution of the KNP. This problem is solved using a quasi Monte Carlo variant of a multi level single linkage algorithm for global optimization. The numerical results indicate that the solution of the KNP is 24 spheres, and not 25.

1 Introduction

When rigid balls touch each other, in technical terms, they "kiss". This is the etimology of the term "kissing number". In mathematical terms, the *kissing number* in D dimensions is the number of D-spheres of radius R that can be arranged around a central D-sphere of radius R so that each of the surrounding spheres touches the central one without overlapping. Determining the maximum kissing number in various dimensions has become a well-known problem in Combinatorial Geometry.

[1] E-mail: `liberti@elet.polimi.it`.
[2] E-mail: `maculan@cos.ufrj.br`.
[3] E-mail: `s.kucherenko@imperial.ac.uk`.

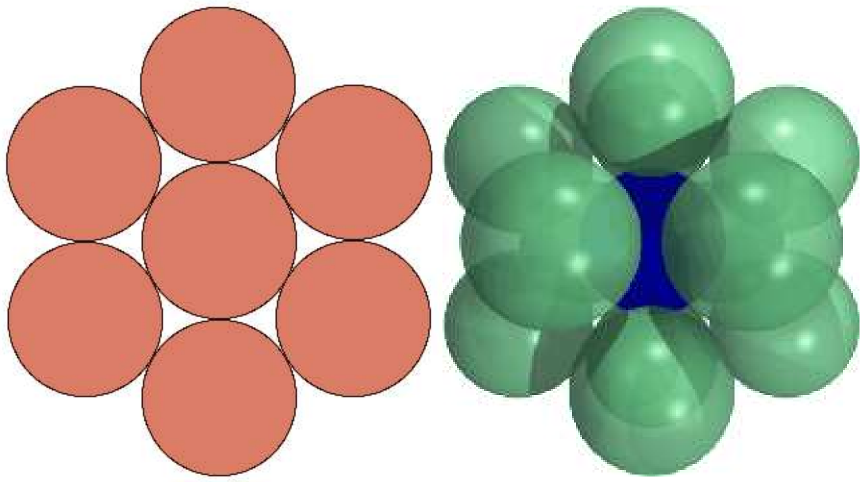


Figure 1.1. The problem in $\mathbb{R}^2$ (a) and $\mathbb{R}^3$ (b)

In $\mathbb{R}^2$ the result is trivial: the maximum kissing number is 6 (Fig. 1, a). The situation is far from trivial in $\mathbb{R}^3$. The problem earned its fame because, according to Newton, the maximum kissing number in 3D is 12, whereas according to his contemporary fellow mathematician David Gregory, the maximum kissing number in 3D is 13 (this conjecture was stated without proof). This question was settled, at long last, more than 250 years after having been stated, when J. Leech finally proved that the solution in 3D is 12 [1]. The question for the 4-dimensional case is still open. A best known solution is 24, and the tightest upper bound is 25 [2]. In this paper, we propose a mathematical programming approach to settle the question. The continuous nonconvex optimization models we shall propose are not dissimilar, in nature, from those found in [3]. However, whereas that paper carried the infamous sentence "the solution of the above programs remains an open question" in its conclusion, in this paper we report the numerical solution of this problem. Our results seem to indicate the solution should be 24. We derived this result by using a general-purpose global optimization software [4] that includes both a stochastic method [5] and a deterministic one [6]. Methodologically speaking, neither method produces an output that is equivalent to a mathematical proof. We are offering strong numerical evidence that seems to point out that the solution of the maximum kissing problem in 4 dimensions is 24.

2 The model

The formulation we propose is a special case of a more general formulation found in [3]. As has been said above, we know a feasible solution with 24 spheres around a central one, and we know that 25 spheres is a tight upper bound. We maximize a decision variable α, bounded by the interval $[0,1]$, which represents the degree of separation of the 25-sphere configuration being tested, subject to the necessary geometric constraints. Since the constraints are nonconvex, there may be multiple local minima. If the global optimization of the model determines that the global maximum is at $\alpha = 1$, then there is enough space for 25 balls, otherwise the kissing number is 24. The geometric variables $x^i = (x^i_1, x^i_2, x^i_3, x^i_4)$,

$1 \leq i \leq 25$, determine the position of the center of the i-th sphere around the central one. The following formulation was used:

$$\begin{aligned} \max \quad & \alpha & & (1) \\ \text{s.t.} \quad & \|x^i\|^2 = 4 & \forall i \leq 25 \quad & (2) \\ & \|x^i - x^j\|^2 \geq 4\alpha & \forall i < j \leq 25 \quad & (3) \end{aligned}$$

3 A quasi Monte Carlo variant of a multi level single linkage algorithm based on Sobol' sequences

A stochastic approach for global optimization, in its simplest form, consists only of random search and it is called Pure Random Search (PRS). In PRS an objective function $f(x)$ is evaluated at N randomly chosen points and the smallest value of $f(x)$ is taken as the global minimum. Advanced stochastic techniques use stochastic methods to search for local minima and then utilize deterministic methods to solve a local minimisation problem. Two phases are considered: global and local. In the global phase, the function is evaluated in a number of randomly sampled points from a uniform distribution over a unit hypercube H_n. In the local phase the sample points are used as starting points for a local minimization search. The efficiency of the multistage methods depends both on the performance of the global stochastic and the local minimization phases.

In the most basic form of the multistage approach a local search is applied to every sample point. Inevitably, some local minima would be found many times. Since the local search is the most CPU-time consuming stage, ideally it should start just once in every region of attraction. This is the idea behind various versions of the so-called clustering methods. Extensive reviews on this subject can be found in [7, 8, 9] and [10]. One of the most efficient clustering methods is a multi level single linkage (MLSL) algorithm developed by Rinnooy-Kan and Timmer in [8, 9].

The efficiency of stochastic methods depends on the quality of sampled points. It has been recognized through theory and practice that uniformly distributed deterministic sequences provide more accurate results than purely random sequences. Low-discrepancy sequences (LDS) are designed specifically to place sample points as uniformly as possible. Unlike random numbers, successive low discrepancy points "know" about the position of their predecessors and fill the gaps left previously. Methods based on LDS are known as quasi Monte Carlo (QMC) methods. In the majority of applications, QMC methods have superior performance compared to that of MC methods. Improvement in time-to-accuracy using QMC methods can be as large as several orders of magnitude. It was shown in [5] that application of LDS can significantly increase the efficiency of MLSL methods.

Central to the QMC approach is the choice of LDS. Different principles were used for constructing LDS by Holton, Faure, Sobol', Niederreiter and others. Many practical studies have proven that Sobol' LDS in many aspects are superior to other LDS [11], [12]. For this reason they were used in the present study. A C++ program called SobolOpt which employs a QMC variant of MLSL based on Sobol' sequences was developed and used in the present study.

4 Computational results

We solved the KNP by using the stochastic solver SobolOpt within the framework of a general-purpose global optimization software *oo*OPS [4]. The solution yielded a result of $\alpha = 0.924126$ in 7m36s of CPU time on a 2.66 GHz Pentium IV CPU with 1GB RAM. This indicates that the solution of the KNP is 24 spheres, and not 25 (we would need $\alpha = 1$ for 25 spheres).

References

[1] J. Leech, The problem of the thirteen spheres. Math. Gazette 40 (1956), 22-23.

[2] Conway, J.H. and Sloane, N.J.A., Sphere Packings, Lattices and Groups, Springer-Verlag, 1993, Berlin

[3] Maculan, N. and Michelon, P. and MacGregor Smith, J., Bounds on the Kissing Numbers in $\mathbb{R}^n$: Mathematical Programming Formulations, Technical Report, University of Massachussets, Amherst, USA, 1996

[4] Liberti, L. and Tsiakis, P. and Keeping, B. and Pantelides, C.C., *oo*OPS, 1.24, Centre for Process Systems Engineering, Chemical Engineering Department, Imperial College London, UK,2001

[5] Kucherenko, S. and Sytsko, Yu., Application of Deterministic Low-discrepancy Sequences to Nonlinear Global Optimization Problems, Computational Optimization and Applications (accepted for publication), 2003

[6] Liberti, L., Reformulation and Convex Relaxation Techniques for Global Optimization, Ph.D. Thesis, Imperial College London, March, 2004

[7] Törn, A. and Žilinskas, A., Global Optimization, Springer-Verlag, 1989, Berlin

[8] Rinnooy-Kan, A.H.G. and Timmer, G.T., Stochastic Global Optimization Methods; Part I: Clustering Methods, Mathematical Programming, 1987, 39, 27-56

[9] Rinnooy-Kan, A.H.G. and Timmer, G.T., Stochastic Global Optimization Methods; Part II: Multilevel Methods, Mathematical Programming, 1987, 39, 57-78

[10] Schoen, F., Stochastic Global Optimization: Two Phase Methods, Encyclopedia of Optimization., Encyclopedia of Optimization., Chris Floudas and Panos Pardalos (eds), Kluwer Academic Publishers, 1999

[11] S. Paskov and J.F. Traub, "Faster evaluation of financial derivatives," The Journal of Portfolio Management, vol. 22, no. 1, pp. 113-120, 1995.

[12] I.M. Sobol', "On quasi-Monte Carlo integrations," Mathematics and Computers in Simulation, vol. 47, pp. 103-112, 1998.

A Special Dynamic Programming Technique for Multiobjective Discrete Control and for Dynamic Games on Graph-Based Networks

D. Lozovanu [1]

Institute of Mathematics and Computer Science, Academy of Sciences, Academy str. 5, Kishinev, MD-2028, Moldova

S.W. Pickl [2]

Department of Computer Science, University of New Mexico, Albuquerque, NM 87131-1386, U.S.A.

Abstract

In this paper we extend the dynamic programming technique for the multiobjective version of a special class of problems representing a graph-theoretic structure on a certain network. We assume that the dynamics of the system is controlled by p actors (players) and each of them intend to minimize his own integral-time cost which is described by a certain trajectory. Applying Nash and Pareto optimality principles we study the multiobjective control problems on dynamic networks where the dynamics is described by a directed graph. Polynomial-time algorithms for determining the optimal strategies of the players in the considered multiobjective control problems are proposed exploiting the special structure of the underlying graph.

Key words: Polynomial Time-Algorithm, Network, c-games

1 Introduction

We study the multiobjective control of time-discrete systems with a finite set of states [1,2]. The main results are based on a game theoretical approach to the following control problem [1–3]: Let L be a time-discrete system with a finite set of states X. At every time step $t = 0, 1, 2, \dots$ the state of L is $x(t) \in X$. Two states x_0 and x_f are given in X where $x_0 = x(0)$ represents the starting point of L and x_f is the state in which the system must be brought, i.e. x_f is the final state of L. We assume that the system L should reach the final

[1] E-mail: `lozovanu@math.md`.
[2] E-mail: `pickl@cs.unm.edu`.

state x_f at the time moment $T(x_f)$ such that

$$T_1 \leq T(x_f) \leq T_2,$$

where T_1 and T_2 are given.

2 Problem Formulation on Dynamic Networks

The dynamics of system L is described by a directed graph $G = (X, E)$, where the vertices $x \in X$ correspond to the states of L and an arbitrary edge $e = (x, y) \in E$ identify the possibility of the system's passage from the state $x = x(t)$ to the state $y = x(t+1)$ at every moment of time $t = 0, 1, 2, \ldots$. So, the set of edges $E(x) = \{e = (x, y) \mid (x, y) \in E\}$ originated in x corresponds to an admissible set of control parameters which determine the next possible states $y = x(t+1)$ of L, if the state $x = x(t)$ at the moment of time t is given. Therefore we consider $E(x) \neq 0$, $\forall\, x \in X \setminus \{x_f\}$ and $E(x_f) = \varnothing$. In addition we assume that to each edge $e = (x, y) \in E$ a cost function $c_e(t)$ is associated which depends on time and which expresses the cost of system L to pass from the state $x = x(t)$ to the state $y = x(t+1)$ at the stage $[t, t+1]$ (like a *transition*). The graph of these state transitions contains edges which represents the time depending cost functions. Furthermore, two vertices correspond to the starting and the final states of the system. We call such a special graph a dynamic network [2,4]. For a given dynamic network we regard the problem of finding a sequence of system's transitions $(x(0), x(1))$, $(x(1), x(2))$, $\ldots$, $(x(T(x_f)-1), x(T(x_f))$ which transfer the system from the starting state $x_0 = x(0)$ to the final state $x_f = x(t(x_f))$, such that $T(x_f)$ satisfy the condition

$$T_1 \leq T(x_f) \leq T_2$$

and the integral time cost

$$F_{x_0 x_f} = \sum_{t=0}^{T(x_f)-1} c_{(x(t), x(t+1))}(t)$$

of system's transitions by a trajectory

$$x_0 = x(0), x(1), x(2), \ldots, x(T(x_f)) = x_f$$

is minimal. This problem generalize the well-known *shortest path problem* in a weighted directed graph and arose as an auxiliary one solving the minimum-cost flow problem on dynamic networks. Algorithms based on dynamic programming methods for finding the optimal trajectory in dynamic networks have been elaborated in [4]. In this paper we extend the dynamic programming technique for the multiobjective version of the mentioned above problem. We assume that the dynamics of the system is controlled by p actors (players) and each of them intend to minimize his own integral-time cost which is described by a certain trajectory. Applying Nash and Pareto optimality principles we study the multiobjective control problems on dynamic networks where the dynamics is described by a directed graph. Polynomial-time algorithms for determining the optimal strategies of the players in considered multiobjective control problems are proposed exploiting the special graph-theoretic structure.

3 Multiobjective control and noncooperative games on dynamic networks

Our main results are related to the following two multiobjective control models concerning stationary and nonstationary strategies.

3.1 The problem of determining the optimal stationary strategies in a dynamic c-game

Let $G = (X, E)$ be the graph introduced in the last section with given starting and final states $x_0, x_f \in X$. Assume that the vertex set X is divided into p disjoint subsets $X_1, X_2, \dots, X_p$ $(X = \bigcup_{i=1}^{p} X_i,\ X_i \cap X_j = \varnothing,\ i \neq j)$ and regard the vertices $x \in X_i$ as states of player i, $i = \overline{1,p}$. Moreover we assume that to each edge $e = (x, y)$ of the graph p functions $c_e^1(t), c_e^2(t), \dots, c_e^p(t)$ are assigned, where $c_e^i(t)$ expresses the cost of system's passage from the state $x = x(t)$ to the state $y = x(t+1)$ at the stage $[t, t+1]$ for player i. We define the stationary strategies of players $1, 2, \dots, p$ as maps:

$$s_i\colon\ x \to y \in X(x), \quad \text{for}\ \ x \in X_i \setminus \{x_f\},\ i = \overline{1,p},$$

where $X(x)$ is a set of edges $e = (x, y)$ starting in x, i.e.

$$X(x) = \{y \in X \mid e = (x, y) \in E\}.$$

Let $s_1, s_2, \dots, s_p$ be an arbitrary set of strategies of the players. We denote by $G_s = (X, E_s)$ the subgraph generated by the edges $e = (x, s_i(x))$ for $x \in X_i \setminus \{x_f\}$ and $i = \overline{1,p}$. Obviously, for fixed $s_1, s_2, \dots, s_p$ either a unique directed path $P_s(x_0, x_f)$ from x_0 to x_f exists in G_s or such a path does not exist in G_s. The set of edges of path $P_s(x_0, x_f)$ is denoted by $E(P_s(x_0, x_f))$. For fixed strategies $s_1, s_2, \dots, s_p$ and fixed states x_0 and x_f we define the quantities

$$H^1_{x_0 x_f}(s_1, s_2, \dots, s_p), H^2_{x_0 x_f}(s_1, s_2, \dots, s_p), \dots, H^p_{x_0 x_f}(s_1, s_2, \dots, s_p)$$

in the following way:
Let us assume that the path $P_s(x_0, x_f)$ exists in G_s. Then it is unique and we can assign to its edges numbers $0, 1, 2, 3, \dots, k_s$, starting with the edge that begins in x_0. These numbers characterize the time steps $t_e(s_1, s_2, \dots, s_p)$ when the system passes from one state to another, if the strategies $s_1, s_2, \dots, s_p$ are applied. We put

$$H^i_{x_0 x_f}(s_1, s_2, \dots, s_p) = \sum_{e \in E(P_s(x_0, x_f))} c_e^i(t_e(s_1, s_2, \dots, s_p)),$$

$$\text{if}\ \ T_1 \leq |E(P_s(x_0, x_f))| \leq T_2;$$

otherwise we put $H^i_{x_0x_f}(s_1, s_2, \dots, s_p) = \infty$. We regard the problem of finding maps $s_1^*, s_2^*, \dots, s_p^*$ for which the following conditions are satisfied

$$H^i_{x_0x_f}(s_1^*, s_2^*, \dots, s_{i-1}^*, s_i^*, s_{i+1}^*, \dots, s_p^*) \le$$

$$\le H^*_{x_0x_f}(s_1^*, s_2^*, \dots, s_{i-1}^*, s_i, s_{i+1}^*, \dots, s_p^*),\ \forall\, s_i,\ i = \overline{1,p}.$$

So we consider the problem of finding the optimal solutions in the sense of Nash. Nash equilibria conditions for this problem have been found and algorithms for determining the optimal stationary strategies of the players have been elaborated [6] and can be derived.

3.2 The problem of determining the optimal nonstationary strategies in a dynamic c-game

We define the nonstationary strategies of the players as maps:

$$u_1\colon\ (x,t) \to (y, t+1) \in X(x) \times \{t+1\} \text{ for } X_1 \setminus \{x_f\},\ t = 0, 1, 2, \dots;$$
$$u_2\colon\ (x,t) \to (y, t+1) \in X(x) \times \{t+1\} \text{ for } X_2 \setminus \{x_f\},\ t = 0, 1, 2, \dots;$$
$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$
$$u_p\colon\ (x,t) \to (y, t+1) \in X(x) \times \{t+1\} \text{ for } X_p \setminus \{x_f\},\ t = 0, 1, 2, \dots.$$

Here (x,t) has the same meaning as the notation $x(t)$, i.e. $(x,t) = x(t)$. For any set of nonstationary strategies $u_1, u_2, \dots, u_p$ we define the quantities

$$F^1_{x_0x_f}(u_1, u_2, \dots, u_p),\ F^2_{x_0x_f}(u_1, u_2, \dots, u_p),\ \dots,\ F^p_{x_0x_f}(u_1, u_2, \dots, u_p)$$

in the following way:

Let $u_1, u_2, \dots, u_p$ be an arbitrary set of strategies. Then either $u_1, u_2, \dots, u_p$ generate in G a finite trajectory

$$x_0 = x(0), x(1), x(2), \dots, x(T(x_f)) = x_f$$

from x_0 to x_f and $T(x_f)$ represents the time moment when x_f is reached, or $u_1, u_2, \dots, u_p$ generate in G an infinite trajectory

$$x_0 = x(0), x(1), x(2), \dots, x(t), x(t+1), \dots$$

which does not pass through x_f, i.e. $T(x_f) = \infty$. In such trajectories the next states $x(t+1)$ is determined uniquely by $x(t)$ and a map u_k, $k \in \{1, 2, \dots, p\}$ as follows

$$x(t+1) = u_k(x(t), t),\ x(t) \in X_k.$$

If the state x_f is reached at a finite moment of time $T(x_f)$ and

$$T_1 \le T(x_f) \le T_2$$

then we set

$$F^i_{x_0 x_f}(u_1, u_2, \ldots, u_p) = \sum_{t=0}^{T(x_f)-1} c_{(x(t),x(t+1))}(t), \ i = \overline{1,p};$$

otherwise we put

$$F^i_{x_0 x_f}(u_1, u_2, \ldots, u_p) = \infty, \ i = \overline{1,p}.$$

Thus we regard the problem of finding the nonstationary strategies $u_1^*, u_2^*, \ldots, u_p^*$ for which the following condition is satisfied

$$F^i_{x_0 x_f}(u_1^*, u_2^*, \ldots, u_{i-1}^*, u_i^*, u_{i+1}^*, \ldots, u_p^*) \leq$$

$$\leq F^*_{x_0 x_f}(u_1^*, u_2^*, \ldots, u_{i-1}^*, u_i, u_{i+1}^*, \ldots, u_p^*), \forall\, u_i, \ i = \overline{1,p}.$$

So, we consider again the problem of finding the optimal solution in the sense of Nash [5].

Conclusion

Regarding the special class of c-games we proved that there exist Nash equilibria if there exists in G a directed path $P_G(x_0, x_f)$ from x_0 to x_f, such that $T_1 \leq |E(P_G(x_0, x_f))| \leq T_2$. Algorithms for determining the optimal strategies for the players are elaborated and will be presented.

References

[1] R. Bellman, R. Kalaba, Dynamic Programming and Modern Control Theory. Academic Press, New York and London, 1965.

[2] D. Lozovanu, S. Pickl, Polynomial time algorithm for determining optimal strategies, Electronic Notes in Discrete Mathematics, 13 (2003), 154–158.

[3] W. Krabs, S. Pickl, Analysis, controllability and optimization of time-discrete systems and dynamical games, Lecture Notes in EConomics and Mathematical Systems, Springer Verlag, Heidelberg, 2003.

[4] J.F. Nash, Non cooperative games, Annals of Math. 2, 1 (1951), 286–295.

[5] D. Lozovanu, Network models of discrete optimal control and dynamic games with p players, Discrete Mathematics and Applications 13 (2001) 126–143.

[6] S. Pickl, Combinatorial Structures and Discrete Optimization within Time-Discrete Systems and Dynamical Games, Habilitation Theses, in preparation, 2004.

On Deviation Measures in Stochastic Integer Programming

Andreas Märkert[1], Rüdiger Schultz[2]

Institute of Mathematics
University of Duisburg-Essen, Campus Duisburg
Lotharstr. 65, D-47048 Duisburg, Germany

This paper deals with mean-risk extensions of the following two-stage mixed-integer linear stochastic program:

$$\min\{Q_{I\!E}(x,\mu)\ :\ x \in X\} \tag{1}$$

where

$$Q_{I\!E}(x,\mu) := \int_{I\!R^s} (c^\top x + \Phi(z - Tx))\, \mu(dz) \tag{2}$$

and

$$\Phi(t) := \min\{q^\top y + q'^\top y'\ :\ Wy + W'y' = t,\ y \in Z\!\!\!Z_+^{\bar{m}},\ y' \in I\!R_+^{m'}\}. \tag{3}$$

All ingredients above are assumed to have conformable dimensions. Moreover, W, W' are rational matrices, and $X \subseteq I\!R^m$ is a nonempty polyhedron, possibly involving integer requirements to components of x. The probability measure μ belongs to $\mathcal{P}(I\!R^s)$, the set of all Borel probability measures on $I\!R^s$. Dependence of $Q_{I\!E}$ on both x and μ is marked explicitly since, later on, $Q_{I\!E}$ and related objects will be studied both as functions in x and jointly in (x,μ). For the moment, let $\mu \in \mathcal{P}(I\!R^s)$ be fixed.
The following assumptions ensure that the above model is well-defined in the sense that $Q_{I\!E}(x,\mu) \in I\!R$ for all $x \in I\!R^m$. For details see Schultz [11].

- (complete recourse) $W(Z\!\!\!Z_+^{\bar{m}}) + W'(I\!R_+^{m'}) = I\!R^s$,
- (sufficiently expensive recourse) $\{u \in I\!R^s\ :\ W^T u \le q,\ W'^T u \le q'\} \neq \emptyset$,
- (finite first moment) $\int_{I\!R^s} \|z\| \mu(dz) < +\infty$.

The model (1)-(3) arises from two-stage mixed-integer linear programs under uncertainty. Decision variables subdivide into two categories. The first-stage decision x has to be taken in a here-and-now manner, before knowing the outcome of the random data $z = z(\omega)$. The second-stage decision (y, y') is taken after x has been fixed and $z(\omega)$ has been observed. Assuming that (y, y') is selected best possible, this two-stage decision process leads to a random cost value that can be expressed as $f(x, z(\omega)) := c^\top x + \Phi(z(\omega) - Tx)$. For further basic details of two-stage stochastic programming we refer to the books Birge and Louveaux

[1] E-mail: `maerkert@math.uni-duisburg.de`.
[2] E-mail: `schultz@math.uni-duisburg.de`.

[1], Kall and Wallace [3], Prékopa [9], Ruszczyński and Shapiro [10].
The problem of finding a best here-and-now decision x in the above setting can be seen as finding a best random variable in the family $\{f(x,z(\omega)) : x \in X\}$. In (1) the traditional approach in two-stage stochastic programming is reflected, namely, the random variables $f(x,z)$ are ranked by their expectations with respect to μ, and $x \in X$ is declared optimal if the expectation of $f(x,z)$ is the least possible. The present paper goes beyond this setting by ranking the above random variables according to weighted sums of their means and suitable terms expressing risk. This leads to mean-risk extensions

$$\min\{Q_{I\!E}(x,\mu) + \alpha \cdot Q_{\mathcal{R}}(x,\mu) \; : \; x \in X\} \tag{4}$$

of the model (1) with fixed weight factor $\alpha \geq 0$ and risk term $Q_{\mathcal{R}}$.
We will study three versions of (4) which are given by the following specifications:

the central deviation, where

$$Q_{\mathcal{R}}(x,\mu) := Q_{\mathcal{D}}(x,\mu) := \int_{I\!R^s} \Big| f(x,z) - Q_{I\!E}(x,\mu) \Big| \, \mu(dz) \tag{5}$$

and $\quad 0 \leq \alpha \leq \frac{1}{2}$,

the semideviation, where

$$Q_{\mathcal{R}}(x,\mu) := Q_{\mathcal{D}^+}(x,\mu) := \int_{I\!R^s} \max\Big\{ f(x,z) - Q_{I\!E}(x,\mu), 0 \Big\} \, \mu(dz) \tag{6}$$

and $\quad 0 \leq \alpha \leq 1$,

and the expected excess of a given target $\eta \in I\!R$, where

$$Q_{\mathcal{R}}(x,\mu) := Q_{\mathcal{D}^\eta}(x,\mu) := \int_{I\!R^s} \max\Big\{ f(x,z) - \eta, 0 \Big\} \, \mu(dz) \tag{7}$$

and $\quad \alpha \geq 0$.

We call the above quantities deviation measures since they are based on expected deviations of the random variable from its mean or from some preselected target. Our emphasis on deviation measures determined by piecewise linear operations (here, taking the absolute value or the maximum) is mainly motivated algorithmically. We will see that, thanks to the piecewise linearity, problem (4) can be tackled successfully by extensions of mixed-integer linear programming techniques provided the underlying measure μ is discrete.
Another motivation for considering the above deviation measures rests in the consistency with stochastic dominance they induce. Stochastic dominance is an established notion of partial order for random variables, see Fishburn [2], Levy [4], Müller and Stoyan [6], Ogryczak and Ruszczyński [7, 8] where different aspects of stochastic dominance as such or in relation with stochastic programming are covered. As an example let us consider second degree stochastic dominance. When preferring small outcomes to big ones, as we do in view of our minimization setting, a (real-valued) random variable X is said to dominate a random variable Y to second degree ($\mathsf{X} \succeq_2 \mathsf{Y}$) if $I\!Eh(\mathsf{X}) \leq I\!Eh(\mathsf{Y})$ for all nondecreasing

convex functions h for which both expectations exist. The mean-risk model (4) is said to be consistent with (second degree) stochastic dominance if the ranking its objective incurs on the random variables $f(x,z)$ inherits a ranking possibly already existing wit h respect to (second degree) stochastic dominance, more precisely, if $f(x_1,z) \succeq_2 f(x_2,z)$ implies $Q_{\mathbb{E}}(x_1,\mu) + \alpha Q_{\mathcal{R}}(x_1,\mu) \leq Q_{\mathbb{E}}(x_2,\mu) + \alpha Q_{\mathcal{R}}(x_2,\mu)$. The catch is that the specifications (5)-(7), with α restricted to the given intervals, all make (4) consistent with second degree stochastic dominance. For details see Fishburn [2], Märkert [5], Ogryczak and Ruszczyński [7].
The purpose of this paper is to study mathematical structures of (4) with the specifications (5)-(7), and to discuss algorithmic approaches to the stochastic integer programs arising. In particular we will study analytical properties of the functionals in (5)-(7) with respect to the decision variable x and the underlying probability measure μ. For problems involving discrete μ we will propose solution procedures relying on decomposition.

References

[1] Birge, J.R.; Louveaux, F.: Introduction to Stochastic Programming, Springer, New York, 1997.

[2] Fishburn, P.C.: Mean-risk analysis with risk associated with below-target returns, American Economic Review 67 (1977), 116-126.

[3] Kall, P.; Wallace, S.W.: Stochastic Programming, Wiley, Chichester, 1994.

[4] Levy, H.: Stochastic dominance and expected utility: survey and analysis, Management Science 38 (1992), 555-593.

[5] Märkert, A.: Deviation measures in stochastic programming with mixed-integer recourse, PhD Thesis, Institute of Mathematics, University of Duisburg-Essen, 2004.

[6] Müller, A.; Stoyan, D.: Comparison Methods for Stochastic Models and Risk, Wiley, Chichester, 2002.

[7] Ogryczak, W.; Ruszczyński, A.: From stochastic dominance to mean-risk models: Semideviations as risk measures, European Journal of Operational Research 116 (1999), 33-50.

[8] Ogryczak, W.; Ruszczyński, A.: Dual stochastic dominance and related mean-risk models, SIAM Journal on Optimization 13 (2002), 60-78.

[9] Prékopa, A.: Stochastic Programming, Kluwer, Dordrecht, 1995.

[10] Ruszczyński, A.; Shapiro, A. (Eds.): Handbooks in Operations Research and Management Science, 10: Stochastic Programming, Elsevier, Amsterdam, 2003.

[11] Schultz, R.: On structure and stability in stochastic programs with random technology matrix and complete integer recourse, Mathematical Programming 70 (1995), 73-89.

A particular class of graphic matroids

Francesco Maffioli[1]

Dip. di Elettronica e Informazione Politecnico di Milano, P.zza L. da Vinci 32, 20133 Milano, Italy

Norma Zagaglia Salvi[2]

Dip. di Matematica, Politecnico di Milano, P.zza L. da Vinci 32, 20133 Milano, Italy

Abstract. *Let $M = (E, \mathcal{F})$ be a matroid on a set E and B one of its bases. A closed set $\theta \subseteq E$ is saturated with respect to B when $\mid \theta \cap B \mid = r(\theta)$, where $r(\theta)$ is the rank of θ. The collection of subsets I of E such that $\mid I \cap \theta \mid \leq r(\theta)$ for every closed saturated set θ turns out to be the family of independent sets of a new matroid on E, called base-matroid and denoted by M_B. In this paper we determine a characterization of a matroid M isomorphic to M_B, for a given base B of M. We also characterize graphic matroids M which are never isomorphic to M_B, for every base B of M.*

AMS Classification: 05B35, 90C27

Keywords: matroid, graphic matroid, uniform matroid, spanning tree, chord.

Let $M = (E, \mathcal{F})$ be a matroid on a set E, having $\mathcal{F}$ as its family of independent sets. Given a set $S \subseteq E$ let us denote by r(S) the rank of S, i.e. the cardinality of the largest independent set contained in S; moreover the closure of S, denoted by $cl(C)$, is the set obtained by adding to S all elements $e \in E$ such that

$$r(S \cup e) = r(S).$$

A set $\theta \subseteq E$ is closed if $\theta = cl(\theta)$, i.e.

$$r(\theta \cup e) = r(\theta) + 1$$

for all $e \in E \setminus \theta$. In the following we denote by Ξ the set of all closed sets of M.

Recall also that

$$\mathcal{F} = \{S \subseteq E : \mid S \cap \theta \mid \leq r(\theta), \forall \theta \in \Xi\}$$

[1] E-mail: `maffioli@elet.polimi.it`.

[2] E-mail: `norzag@mate.polimi.it`.

Definition 1 *A set $\theta \subseteq E$ is said saturated with respect to a base B of M if*

$$\mid \theta \cap B \mid = r(\theta).$$

If θ also belongs to Ξ, we have a saturated closed set (see [3]). The set of all the saturated closed sets of M, with respect to a base B, is denoted by Ξ_B. Note that given any saturated closed set θ, we have

$$cl(\theta \cap B) = \theta;$$

in other words θ coincides with the closure of its intersection with B.

A circuit of M is a minimal dependent set, i.e. a set $S \notin \mathcal{F}$ such that for each $i \in S$, $S \setminus \{i\} \in \mathcal{F}$. Given a base B and an element $i \in E \setminus B$, the fundamental circuit of i is the minimal subset of $B \cup i$ which is not in $\mathcal{F}$. A circuit is *fundamental* with respect to B when it is the fundamental circuit of an element $i \in E \setminus B$. We use the notation

$$\mathcal{F}_B = \{S \subseteq E : \mid S \cap \theta \mid \leq r(\theta), \forall \theta \in \Xi_B\}$$

and

$$M_B = (E, \mathcal{F}_B).$$

M. Dell'Amico, F. Maffioli, F. Malucelli [3] showed that M_B is a matroid, in particular a transversal matroid.

An application of these matroids, named base-matroids, is in the field of inverse combinatorial optimization problems; indeed many different inverse problems have been addressed in the recent literature [1],[3].

In this paper we determine a characterization of a matroid M isomorphic to M_B, for a given base B of M. We also characterize graphic matroids $M = M(G)$ which are never isomorphic to M_B, for every base B, i.e. every spanning tree of G.

First we recall [2] the following definition.

Definition 2 - *Let C be a circuit of a matroid M. An element $e \in E \setminus C$ is a chord of C if $C \cup \{e\}$ is the union of two distinct circuits containing e.*

Lemma 1 - *Let B be a base of M. A fundamental circuit of M with respect to B does not contain chords which belong to B.*

Theorem 1 *Let M be a matroid on E and B a base of M. Then $M \cong M_B$ if and only if every circuit not fundamental with respect to B contains at least one chord which belongs to B.*

Now let us consider the following notion concerning a simple graph G and one of its cycles.

Definition 3 *Let B be a spanning tree and C a cycle of G; C is said independent with respect to B or B-independent if*

$$\mid cl(C) \cap B \mid < \mid C \mid -1.$$

Thus C is dependent with respect to B or B-dependent if

$$\mid cl(C) \cap B \mid = \mid C \mid -1.$$

If C is B - dependent, then $cl(C)$ is saturated with respect to B; in other words a circuit C is B - dependent if it is a circuit also in M_B.

Definition 4 *Two cycles C and H of G are said to be path-intersecting if their intersection is reduced to a path P of length ≥ 2, while $C \setminus P$ and $H \setminus P$ are vertex- disjoint.*

Proposition 1 *If G contains two path-intersecting cycles, then, for every spanning tree B, G contains at least one cycle independent with respect to B.*

Definition 5 *A cycle C of G is said covered when every edge of C belongs to another cycle, distinct from C. For every edge e of C, we denote by $C(e)$ a cycle, distinct from C, which contains e.*

Proposition 2 *If G contains a covered cycle it contains also a cycle independent with respect to B, for every spanning tree B of G.*

Proof : Let B be a fixed spanning tree of G and C a covered cycle. We may assume that C is without chords. Indeed if vw is a chord of C then we may replace a path of C which connects v to w by the edge vw. By a similar motivation we may assume that $C(e)$, for every edge e of C is without chords.

Because C is a cycle, B can not contain all its edges; let g be an edge of C which does not belong to B. If all the edges of $C(g)$, but g, belong to B, then it follows that another edge of C, say f, does not belong to B. If also the edges of $C(f)$ belong to B, we may continue until we found an edge h of C which does not belong to B and such that $C(h)$ contains an edge $\acute{h}$, distinct from h, which does not belong to B. Because $C(h)$ is without chords, it implies

$$\mid cl(C(h)) \cap B \mid < \mid C(h) \mid -1.$$

In other words $C(h)$ is independent with respect to B.

Theorem 2 *Let G be a graph which contains either two path-intersecting cycles or a covered cycle. Then, for every spanning tree B, the graphic matroid $M = M(G)$ is not isomorphic to M_B.*

Proof : It follows from Propositions 1, 2 and [6].

Theorem 3 *Let $M = M(G)$ be a graphic matroid not isomorphic to M_B, for every spanning tree B of G. Then G contains either a pair of path-intersecting cycles or a covered cycle.*

From the above theorems we obtain

Theorem 4 *A graphic matroid $M = M(G)$ is never isomorphic to M_B for every base B of M if and only if G contains either a pair of path-intersecting cycles or a covered cycle.*

References

[1] M. Cai, Inverse problems of matroid intersection, J. Comb. Optim. 3(1999), n. 4, pp 465-474.

[2] W.H. Cunningham, Chords and disjoint paths in matroids, Discrete Mathematics, 19 (1977), 7 - 15.

[3] M. Dell'Amico, F. Maffioli, F. Malucelli, The base-matroid and inverse combinatorial optimization problems, Discrete Applied Mathematics, 128 (2003), 337 - 353.

[4] J. Edmonds and D.R. Fulkerson, Transversals and matroid partition, J. Res. Nat. Bur. Standards Sect. B 69B, 1965, pp. 147-153.

[5] J.G. Oxley, *Matroid Theory*, Oxford University Press, New-York, 1992.

[6] F. Maffioli, N. Zagaglia Salvi, A characterization of the base-matroids of a graphic matroid, submitted.

[7] F. Maffioli, N. Zagaglia Salvi, On some properties of base-matroids, submitted.

Exact algorithms for a discrete metric labeling problem

Gaia Nicosia [1]

Dipartimento di Informatica ed Automazione, Università di Roma Tre, Via della Vasca Navale 79, I-00146 Roma, Italy

Andrea Pacifici [2]

Dipartimento di Ingegneria dell'Impresa, Università di Roma "Tor Vergata", via del Politecnico 1, I-00133 Roma, Italy

Abstract

In this work we deal with a vertex coloring problem where we are given an undirected graph $G = (V, E)$ and a set of colors $C = \{1, 2, \dots k\}$. With each edge $e \in E$ is associated a weight and with each vertex $v \in V$ a subset $\emptyset \neq C_v \subseteq C$. A coloring of the vertices is feasible if each vertex v is colored with a color of C_v. A coloring uniquely defines a subset $E' \subseteq E$ of edges having different colored endpoints. The problem of finding a feasible coloring which defines a minimum weight E' is, in general, NP-complete. In this work an implicit enumeration scheme for finding such an optimal coloring is presented. Upper and lower bounds evaluations are based on a $O(|V|k)$ combinatorial algorithm for the special case of trees and cacti.

Key words: Branch and Bound, Dynamic Programming, Multiway Cut.

1 Introduction

The problem we address in this work was inspired by the following application in Flexible Manufacturing Systems. A set of assembly operations, with precedence constraints among them, must be processed by a set of multi-purpose machines with different capabilities, i.e., every operation may be processed only on a subset of the machines. A *part transfer* occurs every time a part (subassembly) completes its processing on one machine and must be transferred to another machine for the next processing operation [6]. Obviously, it is desirable to assign each operation to a feasible machine minimizing the number of part transfers, in order to reduce possible machines setup and transportation costs.

[1] E-mail: `nicosia@dia.uniroma3.it`.
[2] E-mail: `pacifici@disp.uniroma2.it`.

The above problem can be modeled as follows: a graph $G = (V, E)$ (representing n operations and their pairwise-relationships: e.g., precedences, connections, etc.) and a set of colors $C = \{1, 2, \dots k\}$ (representing the flexible machines) are given together with nonnegative weights $w : E \to \mathbb{Z}_+$ on the edges. Each vertex (operation) $v \in V$ is associated to a set $C_v \subseteq C$ of *feasible colors* (machines) for that vertex. A coloring $\phi : V \to C$ of the vertices is feasible if $\phi(v) \in C_v$. This problem is a special case of the *Metric Labeling Problem* first introduced in [5], where the metric d is a *discrete metric* (i.e. $d(i, j) = 1$ if $i \neq j$ and equal to 0, otherwise), thus we will refer to our problem as *Discrete Metric Labeling Problem* or DMLP .

Any coloring uniquely defines a subset $E' \subseteq E$ of edges having different colored endpoints. E' is a multiway cut, in the sense that its removal disconnects vertices with different colors. Our problem consists of finding a coloring which defines a minimum cost multiway cut. When a color $i \in \bigcap_{v \in V} C_v$ exists, DMLP is trivial: simply color all the vertices with i thus obtaining an empty set of edges with different colored endpoints. When $k = 2$ ($C = \{1, 2\}$), DMLP reduces to a standard maximum $\{s, t\}$-flow/minimum $\{s, t\}$-cut computation on the graph obtained by merging all the vertices having $C_u = \{1\}$ (resp. $C_u = \{2\}$), in one vertex s (resp. one vertex t), and setting the edge capacities for the new graph. Unfortunately, the problem is in general NP-hard since it is a generalization of Multiway Cut, whose complexity was investigated by Dahlhaus *et al.* in [1]. Regarding the approximation complexity of DMLP , it is possible to show that it belongs to APX, since it admits a 2-approximation algorithm [5].

Other problems related to DMLP are the *Colored* Multiway Cut problem studied in [2, 3] and the *Query Tree Coloring* problem, arising in parallel query optimization [4].

2 DMLP on trees

We now describe a dynamic programming algorithm for finding an optimal solution of DMLP , when G is a tree, in time $O(|V|k)$. Let the root of the tree $T = (V, E)$ be any fixed vertex $r \in V$. We denote by $S_u(r)$, or simply by S_u,the sub-tree rooted in u with respect to the root r, i.e, the connected component that contains u, in the graph obtained by removing the edges of the path between u and the root. A vertex v adjacent to u on this path is called the *parent* of u and u is a *child* of v. A *leaf*, with respect to the root r, is any vertex $u \neq r$ whose degree is 1.

The validity of a dynamic programming algorithm is guaranteed by the following lemma.

Lemma 1 *Let $f : V \to C$ be an optimal coloring of the tree $T = (V, E)$ such that $f(v) = i$, then any optimal coloring of S_v obtained by setting $C_v = \{i\}$, is also optimal in the the whole tree.*

Proof. Trivial by contradiction. In fact, if a better solution $\bar{f}$ exists for the sub-tree S_v, a better solution for the whole tree can be obtained by simply exchanging the coloring f relative to this sub-tree, with $\bar{f}$. □ □ Lemma 1, suggest the following procedure: for each vertex $v \in V$ and $i \in C$ we compute the quantity $F_v(i)$ as the *minimum weight of the coloring relative to the sub-tree*

S_v, when vertex v is colored i. Labels are initialized as follows. For each leaf v of T, let

$$F_v(i) = \begin{cases} 0 & \text{if } i \in C_v; \\ +\infty & \text{otherwise.} \end{cases} \tag{1}$$

The following recursion relation holds:

$$F_v(i) := \sum_{u \text{ child of } v} \left(\min_{h \in C_u} \{F_u(h) + w(uv)\delta_{h,i}\} \right) \tag{2}$$

The minimum cost for DMLP is therefore: $z^* = \min_{h \in C_r} \{F_r(h)\}$. Starting from the optimal coloring of the root $f(r) := \arg\min_{h \in C_r} \{F_r(h)\}$ an optimal coloring of all the vertices may be obtained by simple backtracking: $f(u) = \arg\min_{h \in C_u} \{F_u(h) + w(uv)\delta_{h,f(v)}\}$. Based on the Recursion 2 we may derive an optimal solution in $O(|V|k)$. In fact, it is easy to see that in $O(|V|k^2)$ iterations we are able to find $F_v(i)$ for all $v \in V$, $i \in C$. Computation of the minima in the above equations may be devised in time $O(1)$ by keeping track of two additional data for each vertex v of T, during the computation of $F_v(i)$.

The procedure for trees can be easily extended to cycles. On the ground of the last considerations it is not a hard task to devise a procedure for a more general class of graphs, namely *cacti*, which are simple connected graphs with the property that every edge belongs to at most one cycle ([7]).

3 An enumeration scheme for DMLP

In this section we present a branch-and-bound scheme for solving DMLP on arbitrary graphs that uses the above algorithm for trees for determining lower and upper bounds. In the branch-and-bound enumeration tree, which illustrates successive decompositions of the original problem, every node of the enumeration tree represents a particular instance of DMLP in which the set C_u of feasible colors for each vertex u has been changed with respect to the original instance. In particular, any node i of the enumeration tree represents a feasible partial coloring of the vertices of $S_i \subseteq V$. The root node is associated to the original problem where the coloring of no vertex has been decided yet (i.e., $S_{\text{root}} = \emptyset$.) The children of any node i of the enumeration tree, that represent successive decomposition of the subproblem associated to i, are obtained as follows. Pick a vertex $u \in V \setminus S_i$ and generate $|C_u|$ children of i, one for every feasible color for u. Then, any child of i represents a feasible partial coloring of $S_i \cup \{u\}$.

We next show how to obtain a lower bound for DMLP so that it can be used for each node in the enumeration scheme.

3.1 A lower bound

Given two vectors $w, w' \in \mathbb{R}^n$, we write $w' \leq w$ if w' is component-wise not greater than w. Moreover, denote an instance of DMLP with the triple $(G, w, \mathcal{C})$, where G is an undirected graph with its edge-weights $w \in \mathbb{R}^n$, and $\mathcal{C} = \{C_u | \; C_u \subseteq \{1, \ldots, k\}, \; u \in V(G)\}$ is a family of feasible sets of colors for the vertices. It is trivial to show that the following holds.

Lemma 2 *Let z^* and z' be the optimal solution values for two instances of DMLP where $(G, w, \mathcal{C})$ and $(G, w', \mathcal{C})$ of DMLP with $w' \leq w$, then $z' \leq z^*$.*

Based on the preceding Lemma 2, we may easily derive a lower bound for the optimal solution value of DMLP as follows. Consider any spanning tree T of G and let $w'_e = w_e$ for all $e \in T$, else let $w'_e = 0$. Then, any optimal solution for DMLP on T with weights w and feasible labels $\mathcal{C}$ is feasible for $(G, w, \mathcal{C})$ and its value z_T is equal to the optimal solution value of $(G, w', \mathcal{C})$. Then by Lemma 2 z_T is not greater than the optimum of $(G, w, \mathcal{C})$.

Clearly, it would be desirable to know the spanning tree T^* producing the largest lower bound, i.e. $z^* = \max_{T \text{ sp. tree of } G} z_T$. Observe that T^* is not, in general, a maximum weight spanning tree.

3.2 Implementation strategies

Several local improvements have been designed to improve the performance of our branch-and-bound. Here we only cite a *Pre-processing* operation aiming at reducing the size of the instances, and enhancing the quality of the bounds. Furthermore, different branching strategies are tested and simple dominance rules have been devised to reduce the number of children generated at each step.

Finally, since any feasible coloring for a spanning tree is feasible for the original graph, the same approach used for computing a lower bound may be exploited for obtaining an upper bound. This value is computed at each node of the enumeration tree for possible updating of the incumbent solution. Encouraging preliminary computational results are available.

4 Results and conclusions

In this work we proposed exact polynomial time algorithms for DMLP when the given graph is a tree or a cactus. Future research will deal with the problem of extending the class of graphs admitting a polynomial time solution algorithm . More importantly, enhancing the quality of lower bounds is one of the main task for future research. We used the dynamic programming algorithm for trees to obtain such a lower bound. It is open the problem of determining the spanning tree that returns the best (largest) bound. In addition, comparisons between "combinatorial" bounds and bounds obtained via integer programming relaxations will be the object of further investigations.

References

[1] Dahlaus E., Johnson D. S., Papadimitriou C. H., Seymour P. D., Yannakakis M., (1994), *The Complexity of Multiterminal Cuts*, SIAM J. of Comp., 23, 864–894.

[2] Erdös, P. L. and Székeli (1992), *Evolutionary trees: An integer multicommodity max-flow min-cut*, Adv. in Appl. Math., 13, 375–389.

[3] Erdös, P. L. and Székeli (1994), *On weighted multiway cuts in trees*, Mathematical Programming: Series A, v.65 n.1, 93–105.

[4] Hasan, W., and Motwani R., (1995), *Coloring Away Communication in Parallel Query Optimization* Proc. of 21st VLDB Conference.

[5] Kleinberg J., É. Tardos, (1999) Approximation Algorithms for Classification Problems with Pairwise Relationships: Metric Labeling and Markov Random Fields, *Proc. FOCS '99*, 14–15.

[6] Lucertini M., Pacciarelli D., Pacifici A., (1996), *Optimal flow Management in Assembly Processes: The Minimal Part Transfer Problem*, Syst. Sc., 22, 69-80.

[7] Nicosia G., Pacifici A. (2004), *Exact algorithms for a discrete metric labeling problem*, Tech. rep. RT-DIA-90-2004, Dip. Inf. e Autom., Univ. "Roma Tre".

On the Strongly Connected and Biconnected Components of the Complement of Graphs

Stavros D. Nikolopoulos[1], Leonidas Palios[2]

Department of Computer Science, University of Ioannina, P.O.Box 1186, GR-45110 Ioannina, Greece

Abstract

In this paper, we consider the problems of computing the strongly connected components and the biconnected components of the complement of a given graph. In particular, for a directed graph G on n vertices and m edges, we present a simple algorithm for computing the strongly connected components of $\overline{G}$ which runs in optimal $O(n+m)$ time. The algorithm can be parallelized to yield an $O(\log^2 n)$-time and $O(m^{1.188}/\log n)$-processor solution. As a byproduct, we obtain a very simple optimal parallel co-connectivity algorithm.

Additionally, we establish properties which, for an undirected graph on n vertices and m edges, enable us to describe an $O(n+m)$-time algorithm for computing the biconnected components of $\overline{G}$, which can be parallelized resulting in an algorithm that runs in $O(\log n)$ time using $O((n+m)/\log n)$ processors.

1 Theoretical Framework

We consider finite (directed) undirected graphs with no (directed) loops or (directed) multiple edges. Let G be an undirected graph; then, $V(G)$ and $E(G)$ denote the set of vertices and of edges of G respectively.

Lemma 1.1.

(i) Let G be an undirected graph on n vertices and m edges. If v is a vertex of G of minimum degree, then the degree of v does not exceed $\sqrt{2m}$.

(ii) Let G be a directed graph on n vertices and m edges. If v is a vertex of G of minimum sum of indegree and outdegree, then the sum of indegree and outdegree of v does not exceed $2\sqrt{m}$.

[1] E-mail: stavros@cs.uoi.gr.
[2] E-mail: palios@cs.uoi.gr.

Let G be a graph. We say that a set $E \subseteq E(G)$ of cardinality ≥ 2 has the *biconnectivity property in G* if, for every pair of edges $e, e' \in E$, the subgraph of G spanned by the edges in E contains a simple cycle that passes through both e and e' [7].

Lemma 1.2. *Let G be an undirected graph, let set $E \subseteq E(G)$ having the biconnectivity property in G and let $V(E)$ be the set of vertices incident to at least one edge in E. Then,*

(i) the edge set of the subgraph of G induced by $V(E)$ also has the biconnectivity property;

(ii) for every edge $e \in E$ and any two vertices $x, y \in V(E)$, the subgraph of G spanned by the edges in E contains a simple path from x to y that passes along e.

Due to the transitivity of the relation "to have the biconnectivity property" [7], it follows that if two edge sets E_1 and E_2 have the biconnectivity property and are not disjoint then the set $E_1 \cup E_2$ also has the biconnectivity property.

Lemma 1.3. *Let G be an undirected graph, let $E_1, E_2 \subseteq E(G)$ be disjoint sets of edges having the biconnectivity property in G, and let $V(E_1), V(E_2)$ be the sets of vertices incident to at least one edge in E_1 and E_2 respectively.*

(i) If $V(E_1) \cap V(E_2) = \emptyset$ and there exist distinct vertices $u, v \in V(E_1)$ and $x, y \in V(E_2)$ such that $ux \in E(G)$ and $vy \in E(G)$, then the edge set of the subgraph of G induced by $V(E_1) \cup V(E_2)$ has the biconnectivity property.

(ii) Suppose that $V(E_1) \cap V(E_2) = \{v\}$.

a) If there exist vertices $x \in V(E_1) - \{v\}$ and $y \in V(E_2) - \{v\}$ such that $xy \in E(G)$, then the edge set of the subgraph of G induced by $V(E_1) \cup V(E_2)$ has the biconnectivity property;

b) If there exist vertices $x \in V(E_1) - \{v\}$, $y \in V(E_2) - \{v\}$, and vertex $z \in V(G) - (V(E_1) \cup V(E_2))$ such that $xz, yz \in E(G)$, then the edge set of the subgraph of G induced by $V(E_1) \cup V(E_2) \cup \{z\}$ has the biconnectivity property;

c) If there exist vertices $x \in V(E_1) - \{v\}$ and $y \in V(E_2) - \{v\}$, and edge set $E_3 \subseteq E(G)$ for which E_3 has the biconnectivity property and $V(E_3) \cap (V(E_1) \cap V(E_2)) = \emptyset$ such that $xa, yb \in E(G)$ for two distinct vertices $a, b \in V(E_3)$, then the edge set of the subgraph of G induced by $V(E_1) \cup V(E_2) \cup V(E_3)$ has the biconnectivity property.

(iii) If $|V(E_1) \cap V(E_2)| \geq 2$, then the edge set of the subgraph of G induced by $V(E_1) \cup V(E_2)$ has the biconnectivity property.

2 Strongly Connected Components of the Complement of a Graph

Next, we present a simple optimal algorithm for computing the strongly connected components (s.c.c, for short) of the complement $\overline{G}$ of a directed graph G.

Lemma 2.1. *Let G be a directed graph, and let v be a vertex of G.*

(i) Vertex v and the vertices x such that neither vx nor xv belongs to $E(G)$ belong to the same s.c.c of $\overline{G}$.

(ii) Let G'_v be the directed graph where
$V(G'_v) = \{v\} \cup \{\, x \mid vx \in E(G) \text{ or } xv \in E(G)\,\}$;
$E(G'_v) = \{\, xy \mid x, y \in V(G'_v) - \{v\} \text{ and } xy \in E(G)\,\}$
$\quad \cup \{\, vx \mid x \in V(G'_v) - \{v\} \text{ and } \forall z \in V(G) - (V(G'_v) - \{v\}),\ zx \in E(G)\,\}$
$\quad \cup \{\, yv \mid y \in V(G'_v) - \{v\} \text{ and } \forall z \in V(G) - (V(G'_v) - \{v\}),\ yz \in E(G)\,\}$.
Then, two vertices $x, y \in V(G'_v)$ belong to the same s.c.c of $\overline{G}$ iff they belong to the same s.c.c of $\overline{G'_v}$.

The algorithm takes advantage of Lemma 1.1(ii) and Lemma 2.1. It uses an array `sccc[]` of size equal to the number of vertices of the input graph G in which it records the s.c.c of $\overline{G}$; in particular, `sccc[`a`]` = `sccc[`b`]` iff a, b belong to the same s.c.c of $\overline{G}$. In more detail, the algorithm works as follows:

Algorithm Strong_ Co-components

```
1. $v \leftarrow$ a vertex of $G$ of minimum degree (sum of indegree and outdegree);
2. if the indegree and outdegree of $v$ are both equal to 0
   then    {$G$ is trivial or a disconnected graph; $\overline{G}$ is strongly connected}
        for each vertex $w$ of $G$ do
              sccc[$w$] $\leftarrow v$;    {$v$: representative of the s.c.c of $\overline{G}$}; stop;
3. construct the auxiliary graph $G'_v$ defined in Lemma 2.1 and, from that, its complement;
4. compute the strongly connected components of the graph $\overline{G'_v}$ and store them in the standard
   representative-based representation in an array c[];
5. for each vertex $w$ in $V(G'_v)$ do  sccc[$w$] $\leftarrow$ c[$w$];
   for each vertex $w$ in $V(G) - V(G'_v)$ do  sccc[$w$] $\leftarrow$ c[$v$];
```

The above algorithm gives us a very simple s.c.co-components algorithm, which is also optimal. Indeed, because of Lemma 1.1(ii) (which implies that $\overline{G'_v}$ has $O(\sqrt{m})$ vertices, where m is the number of edges of G) and the fact that the strongly connected components of a graph can be computed in time linear in the size of the graph, it is not difficult to see that:

Theorem 2.1. *Let G be a directed graph on n vertices and m edges. Then, the algorithm Strong_ Co-components computes the strongly connected components of $\overline{G}$ in $O(n+m)$ time.*

Using standard parallel algorithmic techniques and the CREW algorithm for computing the strongly connected components of a graph on N vertices in $O(\log^2 N)$ time using $O(N^{2.376}/\log N)$ processors [1, 13, 15], we have:

Theorem 2.2. *Let G be a directed graph on n vertices and m edges. Then, the strongly connected components of $\overline{G}$ can be computed in $O(\log^2 n)$ time using $O(m^{1.188}/\log n)$ processors on the CREW PRAM.*

Moreover, in light of the fact that the connected components of a graph G are identical to the strongly connected components of the directed graph that results by replacing each undirected edge by two oppositely directed edges, a result similar to Lemma 2.1(ii) holds for an appropriate auxiliary graph on $O(\sqrt{m})$ vertices. Then, an algorithm similar to Strong_Co-components, along with the algorithm of Chong *et al.* [4] for computing the connected components of a graph on N vertices in $O(\log N)$ time using $O(N^2/\log N)$ processors on the EREW PRAM, yield an optimal parallel co-connectivity algorithm simpler than the one in [6].

Corollary 2.1. *Let G be an undirected graph on n vertices and m edges. Then, the connected components of $\overline{G}$ can be computed in $O(\log n)$ time using $O((n+m)/\log n)$ processors on the EREW PRAM.*

3 Biconnected Components of the Complement of a Graph

We next present an $O(n+m)$-time algorithm for computing the biconnected components of $\overline{G}$, which can be parallelized resulting in an algorithm that runs in $O(\log n)$ time using $O((n+m)/\log n)$ processors.

Lemma 3.1. *Let G be an undirected graph on m edges and x be any of its vertices. If $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ are the connected components of the subgraph $\overline{G}[M(x)]$ induced by the set $M(x)$ of non-neighbors of x in G, then*

(i) the vertex sets $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ are disjoint;

(ii) their number k does not exceed $2\sqrt{m}$;

(iii) for each $\mathcal{C}_i$, the edge set of the subgraph $\overline{G}[\mathcal{C}_i \cup \{x\}]$ has the biconnectivity property in $\overline{G}$.

Lemma 3.2. *Let G be an undirected graph, v a vertex of G, $E_1, E_2, \ldots, E_\ell$ the biconnected components of $\overline{G}[N(v)]$ with vertex sets $V(E_1), \ldots, V(E_\ell)$ respectively, and $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ the connected components of $\overline{G}[M(v)]$.*

(i) If $|E(G) \cap \{\, xy \mid x \in V(E_i),\, y \in M(v) \,\}| = |V(E_i)| \cdot |M(v)| - 1$, then the two vertices $u \in V(E_i)$ and $w \in M(v)$ which are not adjacent in G define a potential bridge in $\overline{G}$.

(ii) If there exists a vertex $w \in M(v)$ such that $\{\, xy \mid x \in V(E_i),\, y \in M(v) - \{w\} \,\} \subseteq E(G)$ and $|\{\, xw \mid x \in V(E_i)$ and $xw \notin E(G) \,\}| \geq 2$, then the edge set $E_i \cup \{\, xw \mid x \in V(E_i)$ and $xw \notin E(G) \,\}$ has the biconnectivity property in $\overline{G}$ and vertex w is a potential articulation point in $\overline{G}$.

(iii) If there exists a vertex $u \in V(E_i)$ such that $\{\, xy \mid x \in V(E_i) - \{u\},\, y \in M(v) \,\} \subseteq E(G)$ and $|\{\, uy \mid y \in M(v)$ and $uy \notin E(G) \,\}| \geq 2$, then the edge set of the subgraph of $\overline{G}$ induced by $\{v, u\} \cup \{\, \mathcal{C}_j \mid \exists y \in \mathcal{C}_j : uy \notin E(G) \,\}$ has the biconnectivity property in $\overline{G}$ and vertex u is a potential articulation point in $\overline{G}$.

(iv) If there exist vertices $u, u' \in V(E_i)$ and $w, w' \in M(v)$ such that $uw,\, u'w' \notin E(G)$, then the edge set of the subgraph of $\overline{G}$ induced by $\{v\} \cup V(E_i) \cup \{\, \mathcal{C}_j \mid \exists x \in V(E_i)$ and $y \in \mathcal{C}_j : xy \notin E(G) \,\}$ has the biconnectivity property in $\overline{G}$.

In general terms, the algorithm works as follows: It finds a minimum-index vertex of G; let it be v. Next, it computes the biconnected components of $\overline{G}[N(v)]$ and the connected components of $\overline{G}[M(v)]$; recall that the edge set of the subgraph of $\overline{G}$ induced by each of the latter components and v has the biconnectivity property in $\overline{G}$ (Lemma 3.1). Next, the algorithm takes advantage of Lemma 3.2 in order to do a first round of merging of the collected edge sets; to do that, it constructs a graph $\widetilde{G}$ in which the connected components indicate the sets to be merged. Additionally, it has collected potential articulation points and bridge endpoints of $\overline{G}$, from which it constructs another auxiliary graph $\widehat{G}$; the biconnected components of $\widehat{G}$ determine which edge sets will be merged in the second and final round of merging, which yields the biconnected components of $\overline{G}$.

The above algorithm gives us an optimal biconnected co-components algorithm, in light of Lemmas 1.1(i), 3.1, and 3.2 (which imply that the graphs $\overline{G}[N(v)]$, $\widetilde{G}$, and $\widehat{G}$ have $O(\sqrt{m})$ vertices) and the fact that the connected and the biconnected components of a graph can be computed in time linear in the size of the graph. Thus, we have:

Theorem 3.1. *Let G be an undirected graph on n vertices and m edges. Then, the algorithm Biconnected_ Co-components computes the biconnected components of $\overline{G}$ in $O(n+m)$ time.*

Using standard parallel algorithmic techniques, the CREW algorithm for computing the biconnected components of a graph on N vertices in $O(\log N)$ time using $O(N^2/\log N)$ processors [1, 13, 15], and the optimal co-connectivity algorithm of [6] (see also Corollary 2.1), we have the following theorem.

Theorem 3.2. *Let G be an undirected graph on n vertices and m edges. Then, the biconnected components of $\overline{G}$ can be computed in $O(\log n)$ time using $O((n+m)/\log n)$ processors on the CREW PRAM.*

References

[1] S.G. Akl, *Parallel Computation: Models and Methods*, Prentice Hall, 1997.

[2] B. Awerbuch and Y. Shiloach, New connectivity and MSF algorithms for ultra-computer and PRAM, *IEEE Trans. Computers* 36 (1987) 1258–1263.

[3] F.Y. Chin, J. Lam, and I. Chen, Efficient parallel algorithms for some graph problems, *Communications of the ACM* 25 (1982) 659–665.

[4] K.W. Chong, Y. Han, Y. Igarashi, and T.W. Lam, Improving the efficiency of parallel minimum spanning tree algorithms, *Discrete Applied Math.* 126 (2003) 33–54.

[5] K.W. Chong, Y. Han, and T.W. Lam, Concurrent threads and optimal parallel minimum spanning trees algorithm, *J. ACM* 48 (2001) 297–323.

[6] K.W. Chong, S.D. Nikolopoulos, and L. Palios, An optimal parallel co-connectivity algorithm, to appear in *Theory of Computing Systems*, 2004.

[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms* (2nd edition), MIT Press, Inc., 2001.

[8] E. Dahlhaus, J. Gustedt, and R.M. McConnell, Partially complemented representation of digraphs, *Descrete Math. and Theoret. Comput. Science* 5 (2002) 147–168.

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

[10] D.S. Hirschberg, Parallel algorithms for the transitive closure and the connected components problems, *Proc. 8th ACM Symp. on Theory of Computing (STOC'76)*, 55–57, 1976.

[11] D.S. Hirschberg, A.K. Chandra and D.V. Sarwate, Computing connected components on parallel computers, *Communications of the ACM* 22 (1979) 461–464.

[12] H. Ito and M. Yokoyama, Linear time algorithms for graph search and connectivity determination on complement graphs, *Inform. Process. Letters* 66 (1998) 209–213.

[13] J. JáJá, *An Introduction to Parallel Algorithms*, Addison-Wesley, 1992.

[14] D. Nath and S.N. Maheshwari, Parallel algorithms for the connected components and minimal spanning trees, *Inform. Process. Letters* 14 (1982) 7–11.

[15] J. Reif (ed.), *Synthesis of Parallel Algorithms*, Morgan Kaufmann Publishers, San Mateo, California, 1993.

[16] Y. Shiloach and U. Vishkin, An $O(\log n)$ parallel connectivity algorithm, *J. Algorithms* 3 (1982) 57–67.

k-Pseudosnakes in n-dimensional Hypercubes

Erich Prisner [1]

Franklin College, Sorengo/Lugano, Switzerland

Abstract

A k-pseudosnake in a graph is an induced subgraph of maximum degree at most k. In this paper we show that k-pseudosnakes with more than 2^{n-1} vertices exist in the hypercubes Q_n, provided $n \leq 2k$. We also give upper bounds, and show that the generated k-pseudosnakes are maximum provided k is even and $n = 3k/2$. The results also yield better constructions of k-pseudosnakes in large n-dimensional grids in certain cases.

Key words: hypercubes, independent sets, pseudosnakes
PACS:

Let $G = (V, E)$ be any graph. A *k-pseudosnake* is any subset S of V such that the induced subgraph $G[S]$ has maximum degree at most k [1] (also called *k-independent sets* in [2] and [5]). $\alpha_k(G)$ denotes the maximum cardinality of a k-pseudosnake in G. Obviously 0-pseudosnakes are just independent sets, and $\alpha_0(G) = \alpha(G)$ is the usual independence number. We define the densities of the maximum k-pseudosnakes, $\lambda_k(G) := \alpha_k(G)/|V|$.

In this paper we consider hypercubes. The leading question is: For which combinations of k and n is there a k-pseudosnake in Q_n of density larger than 1/2, i.e. for which combinations is $\lambda_k(Q_n) > 0.5$. Using the results, we can slightly improve the lower bounds for λ_k for large n-dimensional grids given in [6]

1 Known Results

For 2-pseudosnakes in hypercubes, everything is known:

Theorem 1 Danzer, Klee 1967: *[4] $\lambda_2(Q_2) = 1, \lambda_2(Q_3) = 3/4, \lambda_2(Q_4) = 9/16$, but $\lambda_2(Q_n) = 1/2$ for every $n \geq 5$.*

The highest index k for which $\alpha_k(G)$ is nontrivial is $k = \Delta(G) - 1$. For r-regular graphs and $k = r - 1$, the parameter α_k is closely related to the domination number γ:

[1] E-mail: `eprisner@fc.edu`.

Proposition 1 *For every r-regular graph G, $\alpha_{r-1}(G) = |V(G)| - \gamma(G)$.*

The paper [3] deals with the domination number in hypercubes. The authors showed that $\gamma(Q_5) = 7, \gamma(Q_6) = 12, \gamma(Q_7) = 16$, and that $\gamma(Q_{2^k-1}) = 2^{n-k}$. Moreover they showed $\gamma(Q_n) \leq 2^{n-3}$ for $n \geq 7$.

Corollary 1.1 *$\lambda_3(Q_4) = 3/4$, $\lambda_4(Q_5) = 25/32$, $\lambda_5(Q_6) = 13/16$, $\lambda_6(Q_7) = 7/8$, $\lambda_{n-1}(Q_n) \geq 7/8$ for $n \geq 7$.*

2 Constructions—Lower bounds

We start with a very simple construction: We glue two copies of Q_n, both containing maximum k-pseudosnakes S_1 and S_2 together to obtain a $k+1$-pseudosnake $S_1 \cup S_2$ in Q_{n+1} of the same density.

Proposition 2 *$\lambda_{k+1}(Q_{n+1}) \geq \lambda_k(Q_n)$.*

Theorem 2 *If $k \geq 1$, then $\lambda_{k+1}(Q_{n+2}) \geq \frac{2\lambda_k(Q_n)+1}{4}$.*

Proof. We are replacing each vertex of a C_4 by the hypercube Q_n to obtain a hypercube $Q_{n+2} = Q_4 \times Q_n$. Let A, B, C, D be the four copies of Q_n in Q_{n+2}, consisting of all vertices of the form $(0,0,x), (0,1,x), (1,1,x)$, and $(1,0,x)$, where x are all vertices in Q_n. Let A' and C' denote k-pseudosnakes in A and C, respectively. Let B' respectively D' denote the set of all vertices of B respectively D with even respectively odd coordinate sum. Of course, $B' \cup D'$ is an independent set. Moreover, every vertex in $A \cup C$ is adjacent to exactly one element of $B' \cup D'$. Since no element of A is adjacent to any element of C, this implies that all vertices of $A' \cup C'$ have at most $k+1$ neighbors in $A' \cup C' \cup B' \cup D'$. Each element of $B' \cup D'$ has at most two neighbors in $A' \cup C' \cup B' \cup D'$—one possibly in A and one possibly in C. If $k \geq 1$, this implies that $A' \cup C' \cup B' \cup D'$ is a $k+1$-pseudosnake in Q_{n+2}. If we choose A' and C' as maximum k-pseudosnakes, we get $|A' \cup C' \cup B' \cup D'| = 2\alpha_{k+1}(Q_n) + 2^n$. ▮

In a similar way it is possible to extend k-pseudosnakes in Q_n into $n+m$-pseudosnakes in Q_{n+2m} with density the average between the densities of the original pseudosnake and $\frac{1}{2}$, provided $m \leq k$.

Theorem 3 *For every positive integer $m \leq k$, $\lambda_{k+m}(Q_{n+2m}) \geq \frac{2\lambda_k(Q_n)+1}{4}$.*

Moreover, it can be shown that the k-pseudosnakes generated with these constructions have the property that the complement is also a k-pseudosnake provided the original pseudosnakes used in the construction have also this property. This property is needed in Section 4.

3 Upper Bounds

Since the projection of every k-pseudosnake in Q_{n+1} is also a k-pseudosnake in Q_n, we get

Proposition 3 $\lambda_k(Q_n) \geq \lambda_k(Q_{n+1})$ *for every* n, k.

A useful simple counting argument of [6], applied to hypercubes, yields the following upper bound:
Theorem 4 *[6]* $\lambda_k(Q_n) \geq \frac{n}{2n-k}$.

Therefore there are infinitely many cases where the solution is known:
Corollary 4.2 *For every* $t \geq 2$ $\lambda_{2t}(Q_{3t}) = \frac{3}{4}$.

Theorem 5 $\lambda_3(Q_5) = \frac{5}{8}$.

4 k-pseudosnakes in large grids

Define $\lambda_k^n(\infty, \ldots, \infty)$ to be the limes superior of all λ_ks of n-dimensional grids, where the side lengths go to infinity. In [6] a construction was shown how to put together k-pseudosnakes (whose complements are also k-pseudosnakes) in Q_n to obtain k-pseudosnakes in the cartesian product of n copies of C_6. These pseudosnakes then were used to tile large n-dimensional grids. Using the results above, we obtain (slightly) better lower bounds for the $\lambda_k^n(\infty, \ldots, \infty)$, for which we just give a few examples:

Corollary 5.3 $\lambda_3^6(\infty, \ldots, \infty) \geq 0.50004287$ *(before the bound was 0.5),*

$\lambda_4^6(\infty, \ldots, \infty) \geq 0.50034294$ *(before the bound was 0.50017),*

$\lambda_4^7(\infty, \ldots, \infty) \geq 0.50005716$ *(before the bound was 0.5),*

$\lambda_4^8(\infty, \ldots, \infty) \geq 0.50000476$ *(before the bound was 0.5),*

$\lambda_5^7(\infty, \ldots, \infty) \geq 0.50011431$ *(before the bound was 0.500057),*

References

[1] H.L. Abbott, M. Katchalski, Snakes and pseudo-snakes in powers of complete graphs, Discrete Math. 68 (1988) 1-8.
[2] J.A. Andrews, M.S. Jacobson, On a generalization of chromatic number and two kinds of Ramsey numbers, Ars Combinatoria 23 (1987) 97-102.
[3] S. Arumugam, R. Kala, Domination parameters of hypercubes, J. Indian Math. Soc. (N.S.) 65 (1998) 31-38.
[4] L. Danzer, V. Klee, Lengths of snakes in boxes, *J. Comb. Th. 2* (1967) 258-265.
[5] J.F. Fink, M.S. Jacobson, On n-domination, n-dependence and forbidden subgraphs, Graph Theory with Applications to Algorithms and Computer Science, Proceedings 5th Int. Conf., Kalamazoo/MI (1985) 301-311.
[6] M. Matamala, E. Prisner, I. Rapaport, k-Pseudosnakes in large grids, LATIN 2002, 224-235.

Semi–preemptive routing on a line

Dirk Räbiger[1]

Zentrum für Angewandte Informatik Köln, Arbeitsgruppe Faigle/Schrader, Universität zu Köln, Weyertal 80, 50931 Köln

Abstract

The problem of routing a robot (or vehicle) between n stations in the plane in order to transport objects is well studied, even if the stations are specially arranged, e.g. on a linear track or circle. The robot may use either *all* or *none* of the stations for reloading. We will generalize these concepts of preemptiveness/non–preemptiveness and emancipate the robot by letting it choose $k \leq n$ reload–stations.

Key words: pickup and delivery, dial–a–ride

1 Introduction

A robot is given the task of transporting m objects between n stations in the plane. Each (heterogeneous) object is initially located at one of the stations and has to be moved to its destination. The robot is only strong enough to hold one object at a time. A station can be source and destination for several objects. We focus on the special case when the n stations, given as the set $\mathcal{S}$, are arranged on a line. There are exactly two stations at both ends of the line that have only one neighbor, any other (inner) station has exactly two neighbors. The distance between neighboring stations i, j is given by $l(i,j) \in \mathbb{R}_+$. If two stations are not neighbors their distance will be the sum of the distances over the unique path using only neighboring stations. Every object has a source $s_i \in \mathcal{S}$ and destination $s_j \in \mathcal{S}$ assigned, and we will call this a *request* (s_i, s_j). We will often use object as a synonym for request. The set of m requests is given as $\mathcal{R}$. Every station is source or destination of a request, otherwise any unused station will be removed. The robot starts at one terminal station $s_0 \in \mathcal{S}$ of the line and moves back and forth along the line to pick up objects, transport them, and drop them. We want to find the minimal motion to transport every object from the source to its destination. Typically, one distinguishes between a *non–preemptive* and a *preemptive* version of the problem. In the first case any object must only be dropped at its destination station once it is picked up. The latter case allows the robot to drop the object at any intermediate node and pick it up later. We will call this action a *reload*. Both cases were studied by Atallah,Kosaraju in [1]. A nice overview of closely related problems is given in [3].

[1] E-mail: `raebiger@zaik.de`.

We want to generalize the concepts of preemptiveness/non–preemptiveness and let the robot know a number $k \in \mathbb{N}, k \leq n$ that defines the maximum quantity of *reload–stations* the transport is allowed to use. The reload–stations may be *exogenously* given as a subset $B \subseteq \mathcal{S}$ and the robot is allowed to use every node $s \in B$ for reloads. In a different model the k reload–stations have to be *endogenously* determined and the robot has to find out itself which stations are best for reloading in order to minimize the total travel length. We will only deal with the more interesting endogenous case in this Extended Abstract. The exogenous case can be easily deducted from it. Moreover, we introduce a cost value $\Delta \in \mathbb{R}$ for every reload station installed.

2 The model and its properties

The goal is to construct a directed multigraph $G_T = (\mathcal{S}, E)$ that the robot can take as routing advice, in the sense that it will move according to the edge set E. If E contains an edge (s_i, s_j) then the robot will move to j when it arrives in i. We cannot use any graph for routing, thus G_T needs certain properties which we will specify soon. The node set $\mathcal{S}$ corresponds to the set of stations. To represent this in G_T, two nodes are *neighboring* if their corresponding stations on the line are neighboring. The distance between nodes corresponds to the stations on the line, and we will use the distance function $l : E \to \mathbb{R}_+$. We number all nodes continuously according to their appearance on the line, thus we can say the nodes s_i, s_{i+1} are neighboring for all $i = 0, \ldots, n-2$.

A request $r_1 = (s_a, s_c) \in \mathcal{R}$ *crosses* station b if $a < b < c$ or $c < b < a$. Suppose the robot transports object r_1 and b was declared to be a reload station. The robot picks up r_1 at node s_a and starts moving towards s_c. Along the way it will pass s_b. At this point it may drop r_1 and transport any other object, before it returns to s_b and continues the transportation of r_1 towards s_c.

Given a directed multigraph $G = (V, A)$, we denote by $\delta^-(v)$ $(\delta^+(v))$ the number of incoming (outgoing) edges of $v \in V$. We will now define what kind of graphs the robot needs. It is easy to see that all the following properties are necessary and sufficient in order to describe a feasible routing for the robot.

Definition 1 *A* transport graph $G_T = (\mathcal{S}, E)$ *has the following properties:*

(1) *For every* $(s_i, s_j) \in \mathcal{R}$ *exists a one–to–one sequence* $((s_{x_0}, s_{x_1}), (s_{x_1}, s_{x_2}), \ldots, (s_{x_{p-1}}, s_{x_p}))$ *where* $\forall_{q=0,\ldots,p-1} : (s_{x_q}, s_{x_{q+1}}) \in E$, $\forall_{q=1,\ldots,p-1} : s_{x_q} \in B \subseteq \mathcal{S}$,

such that $$\begin{cases} s_i = s_{x_0} < s_{x_q} < s_{x_{q+1}} < s_{x_p} = s_j & \text{if } i < j, \text{ and} \\ s_i = s_{x_0} > s_{x_q} > s_{x_{q+1}} > s_{x_p} = s_j & \text{if } i > j \end{cases}$$

(2) G_T *uses at most* k *reloads, i.e.* $|B| \leq k$.

(3) G_T *is degree balanced, i.e.* $\forall_{s \in \mathcal{S}} : \delta^+(s) = \delta^-(s)$

(4) G_T *is connected*

The *cost* of such a graph G_T is $l(G_T) = \sum_{e \in E} l(e) + |B|\Delta$. Property 1 demands that the movement of every object has to be performed, but it allows to split the single edge (s_i, s_j) into several smaller

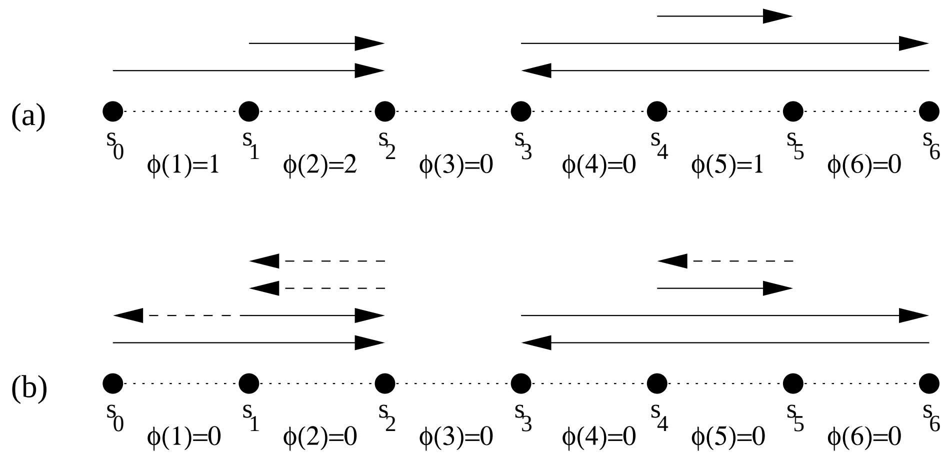

(a)
s_0 s_1 s_2 s_3 s_4 s_5 s_6
$\phi(1)=1$ $\phi(2)=2$ $\phi(3)=0$ $\phi(4)=0$ $\phi(5)=1$ $\phi(6)=0$
(b)
s_0 s_1 s_2 s_3 s_4 s_5 s_6
$\phi(1)=0$ $\phi(2)=0$ $\phi(3)=0$ $\phi(4)=0$ $\phi(5)=0$ $\phi(6)=0$

Figure 2.1. (a) graph with request edges (solid) and flow values (b) same graph with added augmenting edges (dashed)

edges along the unique path from i to j. Note that every request needs its own sequence as the robot can only hold one item at a time. Property 2 permits to split these requests at most k times. Property 3 is known as the *Euler criterium.* Together with the last property it assures that the robot will be able to return to the start node, because every connected component will be strongly connected.

Referring to the line, we call the section between the stations s_i and s_{i+1} the *interval* i, and $l(i) = l(s_i, s_{i+1})$ is the *length of interval* i.

Definition 2 *The* flow $\phi(i)$ *across an interval* i *is defined as*
$\phi(i) = |\{(s_a, s_b) \in E | a \leq i < b\}| - |\{(s_b, s_a) \in E | a \leq i < b\}|$

We know that we do not have to care about the node degrees if and only if we establish a zero flow across all the intervals.

Lemma 1 (Atallah, Kosaraju 1988 [1])
$\forall_{s_i \in \mathcal{S}} : \delta^-(s_i) = \delta^+(s_i) \Leftrightarrow \forall_{i=0,\dots,n-1} : \phi(i) = 0.$

We now want to construct a minimum cost transport graph G_T. Suppose E initially consists of all request edges $(s_i, s_j) \in \mathcal{R}$. Together with the properties of l, one can deduct from Lemma 1 how to minimally augment G_T in order to degree balance all nodes $s_i \in \mathcal{S}$. We do so by adding augmenting edges to E. For any interval i we have to add $\phi(i)$ edges (s_i, s_{i+1}) (resp. (s_{i+1}, s_i)) to E if $\phi(i) < 0$ (resp. > 0). The cost of such an augmentation is $\gamma := \sum_{i=0}^{n-1} l(i)|\phi(i)|$. Figure 2.1 illustrates an example. After adding these edges to E all nodes of G_T are degree balanced, but G_T is not necessarily connected.

Remark 1 *For a given connected component $CC \subseteq \mathcal{S}$ and an arbitrary node $s_i \in CC$, all requests of CC can be independently transported, such that the robot starts and ends in s_i.*

In order to transport all objects of every component the robot must be able to switch from one component to another. The robot has two choices to join connected components. First, it could add more augmenting edges between neighboring nodes of different connected components. Doing so, it will always add two anti–parallel edges (s_i, s_{i+1}), (s_{i+1}, s_i), or otherwise the flow criterium of Lemma 1 will be violated. Alternatively, the robot may use one of the k reload stations to switch from one connected component to another and back again. To calculate the optimal solution out of these alternatives we will construct an auxiliary graph H, whose node set will be the connected components of G_T. We use a known algorithm to construct a spanning tree T for H. In one special case we will need to do some local repair on T in order to use it as direction of how to traverse the connected components of G_T.

3 Endogenous reload stations

Construct an undirected auxiliary graph $H = (C, E^r \dot\cup E^b)$. For every strongly connected component CC_i of G_T, create a supernode $v_i \in C$. The edges are either colored red ($e \in E^r$) or blue ($e \in E^b$). In either case an edge is weighted by $c : E^r \cup E^b \to \mathbb{R}$. Starting with $E^r = E^b = \emptyset$, construct H as follows.

- Add a red edge $(v_i, v_j) \in E^r$ with cost $c(v_i, v_j) = l(a, b)$ to H, if there exist nodes $s_a, s_b \in \mathcal{S}$ which are neighbors, but in different connected components $s_a \in CC_i, s_b \in CC_j, i \neq j$ of G_T.
- Add a blue edge $(v_i, v_j) \in E^b$ with cost $c(v_i, v_j) = \Delta$ to H, if there exists an edge $(s_a, s_c) \in \mathcal{R}$ with $s_a, s_c \in CC_i$ which crosses a node $s_b \in \mathcal{S}, s_b \in CC_j, i \neq j$ in G_T.

We know that H contains a spanning tree on the red edges, because every node $s_i \neq s_0$ has a neighbor s_{i-1} “towards” s_0.

Definition 3 *Let $G = (V, E^r \dot\cup E^b)$ be an undirected graph. A k–*tree *is a spanning tree $T \subseteq E^r \cup E^b$ with $|T \cap E^b| \leq k$.*

Proposition 1 (Gabow, Tarjan 1984 [2]) *Let $G = (V, E = E^r \cup E^b)$ be an undirected graph and $c : E \to \mathbb{R}$ a cost function. If it exists, a minimal cost k–tree $T \subseteq E$ can be calculated in $O(|E| \log |V| + |V| \log |V|)$ time.*

Theorem 1 *Let T be a minimal k–tree of H with cost $c(T) = \sum_{e \in T \cap E^r} c(e) + |T \cap E^b| \Delta$. T can be used to construct an optimal transport graph G^*_T with cost $l(G^*_T) = 2 \sum_{e \in T \cap E^r} c(e) + \beta\Delta + \gamma$ in $O(n \log n)$ time, where γ is the cost to degree balance the initial graph and*

$$\beta = \begin{cases} |T \cap E^b| & if \Delta \geq 0 \\ k & if \Delta < 0 \end{cases}$$

The idea of using the tree T to construct G^*_T is the following, starting with $G^*_T = G_T$. Let CC_0 be the connected component containing the start node s_0. Choose v_0 as root and traverse T using depth-first search. Suppose v_i is the current node and v_j its son. If $(v_i, v_j) \in E^r$ there exist two neighboring nodes $s \in CC_i$ and $t \in CC_j$. Add both edges $(s, t), (t, s)$ to G^*_T. If $(v_i, v_j) \in E^b$ there is a chance that an edge starting and ending in CC_i crosses a node $t \in CC_j$. In this case split the request edge at t and add t to B. Otherwise there is an edge starting and ending in CC_j that crosses a node $t \in CC_i$. We can exploit the structure of the line to show that there must be another blue edge out of an anchestor of v_j that we can use instead. In case $\Delta < 0, |T \cap E^b| < k$, we can choose any node $t \in \mathcal{S} \setminus B$ and add it to B. Repeating this step until $|T \cap E^b| = k$ will improve the quality. If there is a transport graph D with $l(D) < l(G^*_T)$, we can always construct a k–tree T' with $c(T') < c(T)$.

References

[1] M.J. Atallah and S.R. Kosaraju, Efficient solutions to some transportation problems with applications to minimizing robot arm travel, *SIAM Journal Computing.* **17** (1988) 849–869.

[2] H.N. Gabow and R.E. Tarjan, Efficient algorithms for a family of matroid intersection problems. *Journal of Algorithms.* **5** (1984) 80–131.

[3] D.J. Guan, Routing a vehicle of capacity greater than one, *Discrete Applied Mathematics.* **81** (1998) 41–57.

Dynamic programming algorithms for the elementary shortest path problem with resource constraints

Giovanni Righini[1], Matteo Salani[2]

Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano

Abstract

When vehicle routing problems with additional constraints (e.g. capacities or time windows) are solved via column generation and branch-and-price, it is common that the pricing problem requires the computation of a minimum cost constrained path on a graph with costs on the arcs and prizes on the nodes. The pricing problem is usually solved via dynamic programming in two possible ways: requiring elementary paths or allowing paths with cycles. We experimentally compare these two strategies and we evaluate the effectiveness of some algorithmic ideas to improve their performance.

Key words: shortest path, vehicle routing, dynamic programming, column generation

Vehicle routing problems require to compute a set of tours for a fleet of vehicles in charge of providing a given kind of service to a set of customers. Each vehicle starts from a given depot and goes back to it after visiting a subset of customers; the objective is to minimize the total distance traveled. The structure of the vehicle routing problem (VRP) intuitively suggests the use of mathematical programming techniques like Dantzig-Wolfe decomposition and column generation. In a column generation approach the master problem is a set partitioning or set covering problem with a constraint (and a corresponding dual variable) for each customer, while the pricing problem consists of finding a tour with minimum reduced cost. This kind of pricing problem is a shortest path problem with some special characteristics: first, it is formulated on a graph with costs on the arcs and prizes on the nodes, where the prizes correspond to the optimal dual values computed at the last iteration on the restricted linear master problem. This is equivalent to formulate it on a graph with no prizes but with negative cost arcs and possibly negative cost cycles. Therefore the requisite that the solution must be an elementary path does no longer come for free from cost minimization and it must be explicitely enforced. Second, the pricing problem is typically subject to a number of restrictions, due to the limited capacity of the vehicles or to time windows. These constraints are very common in many vehicle routing applications and they are not taken into account at master problem level; they are usually represented as resource constraints, since distances, costs, time, capacities can all be interpreted as resources, that are consumed every time a vehicle travels along

[1] E-mail: `righini@dti.unimi.it`.
[2] E-mail: `salani@dti.unimi.it`.

an arc or visits a customer. Therefore the pricing problem turns to be an elementary shortest path problem with resource constraints (ESPPRC).

The elementary shortest path problem (without resource constraints) has been studied in many textbooks: see for instance the classical reference by Ahuja et al. [1], chapter 5. The shortest path problem with resource constraints has been addressed with methods based on Lagrangean relaxation of the resource constraints, like those of Handler and Zang [10] and Beasley and Christofides [2]; however these methods are effective when the Lagrangean subproblem is easy, that is when arc costs are non-negative.

The ESPPRC is NP-complete [7] and it has been addressed so far by dynamic programming, following the approach of Desrochers et al. [5], who exploited a pulling label-correcting algorithm also described in [6] and [8]. In [5] the authors describe a state-space relaxation such that paths with cycles are allowed. In this way the pricing problem can be solved in pseudo-polynomial time; the drawback is that the lower bound computed at each node of the search tree, when this technique is employed in a branch-and-price framework, can be weaker than it would have been if only elementary paths had been considered. Feillet et al. [8] and Chabrier et al. [3] worked in the direction of forbidding cycles and enhancing the performance of the exact dynamic programming algorithm. They claimed that this choice yields much better bounds at the root node (and sometimes integer solutions) and its cost in terms of increased computing time is tolerable. All these papers mainly refer to the VRP with time windows, even if Feillet et al. also cite some other applications, quoting an unpublished Ph.D. thesis [9].

There is an evident trade-off between the computing time spent at each column generation iteration by the pricing algorithm and the quality of the lower bound obtained, that in turn influences the overall effectiveness of branch-and-price. Our experience with a branch-and-price algorithm for the VRP with simultaneous pick-up and delivery [4] brought to our attention the need of comparing exact dynamic programming versus state-space relaxation on a variety of routing problems and with constraints of different strength and this is the purpose of this paper. Moreover we suggest here some algorithmic ideas to improve the performance of dynamic programming algorithms, both exact and with state-space relaxation, and for each of them we present some experimental measures of effectiveness. In particular we illustrate and discuss bi-directional search, bounded-depth tree search, 2-cycle and k-cycle avoidance, cycle prevention on critical nodes, stabilization techniques and efficient data-structures.

Our purpose is to give a useful contribution to developers of branch-and-price algorithms for routing problems in the implementation of effective pricing algorithms and in the choice of the most suitable dynamic programming algorithm according to the specific constraints of the problem instances at hand. For this reason we also discuss the compatibility of the enhanced dynamic programming algorithms with various branching strategies.

References

[1] Ahuja R.K., Magnanti T.L., Orlin J.B., *Network flows*, Prentice Hall 1993

[2] Beasley J.E., Christofides N., *An algorithm for the resource constrained shortest path problem*, Networks 19 (1989) 379-394

[3] Chabrier A., Danna E., La Pape C., *Coopération entre génération de colonnes avec tournées sans cycle et recherche locale appliquée au routage de véhicules*, Actes JNPC'02 (2002) 83-97

[4] Dell'Amico M., Righini G., Salani M., *A branch-and-price algorithm for the vehicle routing problem with simultaneous pick-up and delivery*, Note del Polo 51, Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano (2003) [submitted to Transportation Science]

[5] Desrochers M., Desrosiers J., Solomon M., *A new optimization algorithm for the vehicle routing problem with time windows*, Operations Research 40 (1992) 342-354

[6] Desrosiers J., Dumas Y., Solomon M., Soumis F., *Time constrained routing and scheduling* in *Network Routing*, Handbooks in Operations Research and Management Science, Ball M.O. et al. eds., Elsevier Science 1995

[7] Dror M., *Note on the complexity of the shortest path models for column generation in VRPTW*, Operations Research 42 (1994) 977-978

[8] Feillet D., Dejax P., Gendreau M., Gueguen C., *An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems*, Laboratoire Informatique d'Avignon, 2003

[9] Gueguen C., *Méthodes de résolution exacte pour les problèmes de tournées de véhicules*, PhD thesis, Laboratoire Productique Logistique, Ecole Centrale Paris (1999)

[10] Handler G.Y., Zang I., *A dual algorithm for the constrained shortest path problem*, Networks 10 (1980) 293-310

The Ramsey Numbers of Paths Versus Kipases

A. N. M. Salman[1], H. J. Broersma[2]

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

Abstract

For two given graphs G and H, the Ramsey number $R(G, H)$ is the smallest positive integer p such that for every graph F on p vertices the following holds: either F contains G as a subgraph or the complement of F contains H as a subgraph. In this paper, we study the Ramsey numbers $R(P_n, \hat{K}_m)$, where P_n is a path on n vertices and $\hat{K}_m$ is the graph obtained from the join of K_1 and P_m. We determine the exact values of $R(P_n, \hat{K}_m)$ for the following values of n and m: $1 \leq n \leq 5$ and $m \geq 3$; $n \geq 6$ and (m is odd, $3 \leq m \leq 2n-1$) or (m is even, $4 \leq m \leq n+1$); $n = 6$ or 7 and $m = 2n-2$ or $m \geq 2n$; $n \geq 8$ and $m = 2n-2$ or $m = 2n$ or $(q \cdot n - 2q + 1 \leq m \leq q \cdot n - q + 2$ with $3 \leq q \leq n-5)$ or $m \geq (n-3)^2$; odd $n \geq 9$ and $(q \cdot n - 3q + 1 \leq m \leq q \cdot n - 2q$ with $3 \leq q \leq (n-3)/2)$ or $(q \cdot n - q - n + 4 \leq m \leq q \cdot n - 2q$ with $(n-1)/2 \leq q \leq n-4)$.

Key words: kipas, path, Ramsey number
AMS Subject Classifications: 05C55, 05D10

1 Introduction

Throughout this paper, all graphs are finite and simple. Let G be such a graph. The graph $\overline{G}$ is the *complement* of G, i.e., the graph obtained from the complete graph on $|V(G)|$ vertices by deleting the edges of G. A *kipas* $\hat{K}_m$ is the graph on $m+1$ vertices obtained from the join of K_1 and P_m. The vertex corresponding to K_1 is called the *hub* of the kipas. Given two graphs G and H, the *Ramsey number* $R(G, H)$ is defined as the smallest positive integer p such that every graph F on p vertices satisfies the following condition: F contains G as a subgraph or $\overline{F}$ contains H as a subgraph.

In 1967 Geréncser and Gyárfás [3] determined all Ramsey numbers for paths versus paths. After that, Ramsey numbers $R(P_n, H)$ for paths versus other graphs H have been investigated in several papers, for example in [5], [1], [6], [4], [2], [7] and [8]. We study Ramsey numbers for paths versus kipases.

[1] E-mail: `a.n.m.salman@math.utwente.nl`.
[2] E-mail: `h.j.broersma@math.utwente.nl`.

2 Main results

We determine the Ramsey numbers $R(P_n, \hat{K}_m)$ for the following values of n and m: $1 \leq n \leq 5$ and $m \geq 3$; $n \geq 6$ and (m is odd, $3 \leq m \leq 2n-1$) or (m is even, $4 \leq m \leq n+1$); $n = 6$ or 7 and $m = 2n-2$ or $m \geq 2n$; $n \geq 8$ and $m = 2n-2$ or $m = 2n$ or $(q \cdot n - 2q + 1 \leq m \leq q \cdot n - q + 2$ with $3 \leq q \leq n-5)$ or $m \geq (n-3)^2$; odd $n \geq 9$ and $(q \cdot n - 3q + 1 \leq m \leq q \cdot n - 2q$ with $3 \leq q \leq (n-3)/2)$ or $(q \cdot n - q - n + 4 \leq m \leq q \cdot n - 2q$ with $(n-1)/2 \leq q \leq n-4)$.

Theorem 1

$$R(P_n, \hat{K}_m) = \begin{cases} 1 & \text{for } n = 1 \text{ and } m \geq 3 \\ m+1 & \text{for either } (n = 2 \text{ and } m \geq 3) \\ & \text{or } (n = 3 \text{ and even } m \geq 4) \\ m+2 & \text{for } (n = 3 \text{ and odd } m \geq 5) \\ 3n-2 & \text{for either } (n = 3 \text{ and } m = 3) \\ & \text{or } (n \geq 4 \text{ and } m \text{ is odd}, 3 \leq m \leq 2n-1) \\ 2n-1 & \text{for } n \geq 4 \text{ and } m \text{ is even}, 4 \leq m \leq n+1. \end{cases}$$

Theorem 1 can be obtained by indicating suitable graphs for providing sharp lower bounds, and using some result in [8] for getting the best upper bounds. We omit the details.

The next lemma plays a key role in our proofs of Lemma 2 and Lemma 3. The proof of this lemma has been given in [7].

Lemma 1 *Let $n \geq 4$ and G be a graph on at least n vertices containing no P_n. Let the paths $P^1, P^2, \ldots, P^k$ in G be chosen in the following way: $\bigcup_{j=1}^{k} V(P^j) = V(G)$, P^1 is a longest path in G, and, if $k > 1$, P^{i+1} is a longest path in $G - \bigcup_{j=1}^{i} V(P^j)$ for $1 \leq i \leq k-1$. Let z be an end vertex of P^k. Then:*

(1) $|V(P^1)| \geq |V(P^2)| \geq \ldots \geq |V(P^k)|$;

(2) If $|V(P^k)| \geq \lfloor n/2 \rfloor$, then $|N(z)| \leq |V(P^k)| - 1$;

(3) If $|V(P^k)| < \lfloor n/2 \rfloor$, then $|N(z)| \leq \lfloor n/2 \rfloor - 1$.

Lemma 2 *If $n \geq 4$ and $m = 2n-2$ or $m \geq 2n$, then*

$$R(P_n, \hat{K}_m) \leq \begin{cases} m+n-1 & \text{for } m = 1 \ mod(n-1) \\ m+n-2 & \text{for other values of } m. \end{cases}$$

Proof. Let G be a graph that contains no P_n and has order

$$|V(G)| = \begin{cases} m+n-1 \text{ for } m = 1 \; mod(n-1) \\ m+n-2 \text{ for other values of } m. \end{cases} \qquad (1)$$

Choose the paths $P^1, \ldots, P^k$ and the vertex z in G as in Lemma 1. Because of (1), not all P^i can have $n-1$ vertices, so $|V(P^k)| \leq n-2$. By Lemma 1, $|N(z)| \leq n-3$. We will use the following result that has been proved in [1]: $R(P_t, C_s) = s + \lfloor t/2 \rfloor - 1$ for $s \geq \lfloor (3t+1)/2 \rfloor$. We distinguish the following cases.

Case 1 $|N(z)| \leq \lfloor n/2 \rfloor - 2$ or n is odd and $|N(z)| = \lfloor n/2 \rfloor - 1$.
Since $|V(G) \setminus N[z]| \geq m + \lfloor n/2 \rfloor - 1$, we find that $\overline{G - N[z]}$ contains a C_m. So, there is a $\hat{K}_m$ in $\overline{G}$ with z as a hub.

Case 2 n is even and $|N(z)| = n/2 - 1$.
Since $|V(G) \setminus N[z]| \geq (m+n-2) - n/2 = m + n/2 - 2$, we find that $\overline{G - N[z]}$ contains a C_{m-1}; denote its vertices by $v_1, v_2, v_3, \ldots, v_{m-1}$ in the order of appearance on the cycle with a fixed orientation. There are $n/2 - 1$ vertices in $U = V(G) \setminus (V(C_{m-1}) \cup N[z])$, say $u_1, u_2, \ldots, u_{n/2-1}$. If some vertex v_i $(i = 1, \ldots, m-1)$ is no neighbor of some vertex u_j $(j = 1, \ldots, n/2-1)$, w.l.o.g. assume $v_{m-1}u_1 \notin E(G)$. Then $\overline{G}$ contains a $\hat{K}_m$ with hub z and its other vertices $v_1, v_2, v_3, \ldots, v_{m-2}, v_{m-1}, u_1$. Now let us assume each of the v_i is adjacent to all u_j in G. For every choice of a subset of $n/2$ vertices from $V(C_{m-1})$, there is a path on $n-1$ vertices in G alternating between the vertices of this subset and the vertices of U, starting and terminating in two arbitrary vertices from the subset. Since G contains no P_n, there are no edges $v_iv_j \in E(G)$ $(i, j \in \{1, \ldots, m-1\})$. This implies that $V(C_{m-1}) \cup \{z\}$ induces a K_m in $\overline{G}$. Since G contains no P_n, no v_i is adjacent to a vertex of $N(z)$. This implies that $\overline{G}$ contains a $K_{m+1} - e$ for some edge zw with $w \in N(z)$, and hence $\overline{G}$ contains a $\hat{K}_m$ with one of the v_i as a hub.

Case 3 Suppose that there is no choice for P^k and z such that one of the former cases applies. Then $|N(w)| \geq \lfloor n/2 \rfloor$ for any end vertex w of a path on $|V(P^k)|$ vertices in $G - \bigcup_{j=1}^{k-1} V(P^j)$. This implies all neighbors of such w are in $V(P^k)$ and $|V(P^k)| \geq \lfloor n/2 \rfloor + 1$. So for the two end vertices z_1 and z_2 of P^k we have that $|N(z_i) \cap V(P^k)| \geq \lfloor n/2 \rfloor \geq |V(P^k)|/2$. By standard arguments in hamiltonian graph theory we obtain a cycle on $|V(P^k)|$ vertices in G. This implies that any vertex of $V(P^k)$ could serve as w. By the assumption of this last case, we conclude that there are no edges in G between $V(P^k)$ and the other vertices. This also implies that all vertices of P^k have degree in $\overline{G}$ at least

$$\begin{cases} m+1 \text{ if } |V(G)| = m+n-1 \\ m \qquad \text{if } |V(G)| = m+n-2. \end{cases}$$

We now turn to P^{k-1} and consider one of its end vertices w. Since $|V(P^{k-1})| \geq |V(P^k)| \geq \lfloor n/2 \rfloor + 1$, similar arguments as in the proof of Lemma 1 show that all neighbors of w are on P^{k-1}. If $|N(w)| < \lfloor n/2 \rfloor$, we get a $\hat{K}_m$ in $\overline{G}$ as in Case 1 and 2. So we may assume $|N(w_i) \cap V(P^{k-1})| \geq \lfloor n/2 \rfloor \geq |V(P^{k-1})|/2$ for both end vertices w_1 and w_2 of P^{k-1}. By standard arguments in hamiltonian graph theory we obtain a cycle on $|V(P^{k-1})|$ vertices in G. This implies that any vertex of $V(P^{k-1})$ could serve as w. By the assumption of this last case, we conclude that there are no edges in G between

$V(P^{k-1})$ and the other vertices. This also implies that all vertices of P^{k-1} have degree in $\overline{G}$ at least

$$\begin{cases} m & \text{if } |V(G)| = m+n-1 \\ m-1 & \text{if } |V(G)| = m+n-2. \end{cases} \tag{2}$$

Repeating the above arguments for $P^{k-2}, \ldots, P^1$ we eventually conclude that all vertices of G have degree in $\overline{G}$ at least as (2).

Now let $|V(P^k)| = \ell$ and $H = \overline{G} - V(P^k)$. If $V(G) = m+n-1$, then in the graph H all vertices have degree at least $m - \ell \geq m/2 + (n-1) - \ell \geq \frac{1}{2}(m + 2n - 2 - \ell - (n-2)) = \frac{1}{2}(m+n-\ell) = \frac{1}{2}(|V(H)|+1)$. If $V(G) = m+n-2$, then in the graph H all vertices have degree at least $m - 1 - \ell \geq m/2 + (n-1) - 1 - \ell \geq \frac{1}{2}(m + 2n - 4 - \ell - (n-2)) = \frac{1}{2}(m+n-2-\ell) = \frac{1}{2}|V(H)|$. Hence, there exists a Hamilton cycle in H. Since $|V(H)| \geq m$ and z is a neighbor of all vertices in H, it is clear that $\overline{G}$ contains a $\hat{K}_m$ with z as a hub. ∎

Corollary 1.1 *If* $(4 \leq n \leq 6$ *and* $m = 2n-2$ *or* $m \geq 2n)$ *or* $(n \geq 7$ *and* $m = 2n-2$ *or* $m = 2n$ *or* $m \geq (n-3)^2)$ *or* $(n \geq 8$ *and* $q \cdot n - 2q + 1 \leq m \leq q \cdot n - q + 2$ *for* $3 \leq q \leq n-5)$, *then*

$$R(P_n, \hat{K}_m) = \begin{cases} m+n-1 & \textit{for } m = 1 \textit{ mod}(n-1) \\ m+n-2 & \textit{for other values of } m. \end{cases}$$

Corollary 1.1 can be obtained by indicating suitable graphs for providing sharp lower bounds, and combining them with the upper bounds from Lemma 2. We omit the details.

Lemma 3 *If odd* $n \geq 7$ *and* $q \cdot n - q + 3 \leq m \leq q \cdot n - 2q + n - 2$ *with* $2 \leq q \leq n-5$, *then* $R(P_n, \hat{K}_m) \leq m+n-3$.

The proof of Lemma 3 is modeled along the lines of the proof of Lemma 2. We omit the details.

Corollary 1.2 *If* $(n = 7$ *and* $m = 15)$ *or* (*odd* $n \geq 9$ *and* $(q \cdot n - 3q + 1 \leq m \leq q \cdot n - 2q$ *with* $3 \leq q \leq (n-3)/2)$ *or* $(q \cdot n - q - n + 4 \leq m \leq q \cdot n - 2q$ *with* $(n-1)/2 \leq q \leq n-4)$), *then* $R(P_n, \hat{K}_m) = m+n-3$.

Proof. For $n = 7$ and $m = 15$, the graph $3K_6$ and for odd $n \geq 9$ and $m = q \cdot n - 2q - j$ with either $(3 \leq q \leq (n-3)/2$ and $0 \leq j \leq q-1)$ or $((n-1)/2 \leq q \leq n-5$ and $0 \leq j \leq n-q-4)$, the graph $(q-j-1)K_{n-2} \cup (j+2)K_{n-3}$ shows that $R(P_n, \hat{K}_m) > m+n-4$. Using Lemma 3, we obtain that $R(P_n, \hat{K}_m) = m+n-3$. ∎

References

[1] R.J. Faudree, S.L. Lawrence,T.D. Parsons and R.H. Schelp, Path-cycle Ramsey numbers, *Discrete Mathematics*, **10** (1974), 269–277.

[2] R.J. Faudree, R.H. Schelp and M. Simonovits, On some Ramsey type problems connected with paths, cycles and trees, *Ars Combinatoria*, **29A** (1990), 97–106.

[3] L. Gerécser and A. Gyárfás, On Ramsey-type problems, *Annales Universitatis Scientiarum Budapestinensis, Eötvös Sect. Math.*, **10** (1967), 167–170.

[4] R. Häggkvist, On the path-complete bipartite Ramsey numbers, *Discrete Mathematics*, **75** (1989), 243–245.

[5] T.D. Parsons, The Ramsey numbers $r(P_m, K_n)$, *Discrete Mathematics*, **6** (1973), 159–162.

[6] T.D. Parsons, Path-star Ramsey numbers, *Journal of Combinatorial Theory*, Series B, **17** (1974), 51–58.

[7] A.N.M. Salman and H.J. Broersma, Path-fan Ramsey numbers, *Submitted to Discrete Applied Mathematics*, (2003).

[8] A.N.M. Salman and H.J. Broersma, On Ramsey numbers for paths versus wheels, *Accepted for Discrete Mathematics*, (2004).

Connections between continuous and combinatorial optimization problems through an extension of the fundamental theorem of Linear Programming

Fabio Tardella[1]

Dipartimento di Matematica, Facoltà di Economia e Commercio, Via del Castro Laurenziano 9, 00161 Roma, Italy

Abstract

We describe a common extension of the fundamental theorem of Linear Programming on the existence of a global minimum in a vertex for lower bounded linear programs, and of the Frank-Wolfe theorem on the existence of the minimum of a lower bounded quadratic function on a polyhedron.

We then show that several known results providing continuous formulations for discrete optimization problems can be easily derived and generalized with our result. These include the Quadratic Programming formulation of the maximum clique problem by Motzkin and Straus and its weighted extension by Gibbons et al., the equivalence between the minimization of a multilinear function on the continuous and discrete unit hypercube by Rosenberg, and a recent continuous polynomial formulation of the maximum independent set problem by Abello et al.

Furthermore, we use our extension of the fundamental theorem of Linear Programming to obtain combinatorial formulations and polynomiality results for some nonlinear problems with simple polyhedral constraints.

1 Introduction

For the solution of several optimization problems it is often advantageous to be able to approach them with different and complementary methods. The most notable example is Linear Programming, where both a continuous and a combinatorial structure are used on the basis of the celebrated *fundamental theorem* that guarantees the existence of a vertex optimal solution. Such result remains valid and is used also for the broader class of (quasi-) concave minimization problems on a polyhedron.

We describe here a further extension of the fundamental theorem of Linear Programming based on the concept of directional (quasi-) concavity, and we show that this result can be used as a general tool for establishing the Frank-Wolfe theorem on the existence of the minimum in lower bounded

[1] E-mail: `fabio.tardella@uniroma1.it`.

quadratic programs as well as new and old connections between continuous and combinatorial optimization problems.

In particular, we describe connections among nonlinear programs with box constraints, pseudo-Boolean optimization and the maximum weight independent set problem on a graph. Furthermore, we establish the equivalence between a nonlinear optimization problem over a simplex and a minimum weight (or maximal) clique problem on a graph. Some special cases of the equivalent formulations presented here have been obtained in the literature [1, 4, 5, 6], and have been exploited to tackle combinatorial optimization problems with nonlinear programming tools. Here we point out that also the opposite direction might be fruitful. Indeed, we show that some nonlinear programming problems can be solved in polynomial time by means of a reduction to an equivalent polynomially solvable combinatorial problem.

2 An extension of the fundamental theorem of Linear Programming

We now describe an extension of the fundamental theorem of Linear Programming based on the concept of directional (quasi-) concavity of a function. We first need to introduce some definitions and notation. The *affine hull* of a subset S of $\mathbb{R}^n$ is the smallest affine manifold containing S. We denote by $\mathrm{lin}(S)$ (with a non-standard notation) the smallest linear subspace of $\mathbb{R}^n$ containing $S - \{x^0\} = \{x - x^0 : x \in S\}$, where x^0 is a point of S. The *relative interior* $\mathrm{rint}\,(S)$ of S is the interior of S for the topology relative to the affine hull of S.

Given a pointed polyhedron P, a point x in P and a direction $d \in \mathbb{R}^n \setminus \{0\}$ we consider the set $P_d(x) = \{z \in P : z = x + \lambda d, \lambda \in \mathbb{R}\}$ obtained intersecting P with the line passing through x with direction d. Note that, since P is pointed, $P_d(x)$ is either a segment or a half-line for every x in P and $d \in \mathbb{R}^n \setminus \{0\}$.

We will use the following conditions:

$$P_d(x) \text{ is bounded and } f \text{ is quasi-concave on } P_d(x); \tag{1}$$

$$f \text{ is concave and bounded below on } P_d(x); \tag{2}$$

$$f \text{ is strictly quasi-concave on } P_d(x). \tag{3}$$

We say that a face F of P satisfies *Property A, B* or *C*, if for every $x \in \mathrm{rint}\, F$ there exists $d \in \mathrm{lin}(F) \setminus \{0\}$, such that:

Property A condition (1) or (2) or (3) holds;

Property B condition (1) or (2) holds;

Property C condition (3) holds.

We let $\mathcal{F}_A$, $\mathcal{F}_B$ and $\mathcal{F}_C$ denote the sets of all faces of P that *do not* satisfy Property A B and C, respectively. By definition, each set $\mathcal{F}_A$, $\mathcal{F}_B$ and $\mathcal{F}_C$ contains all the vertices V_P of P.

Theorem 1 (Existence and location of minima) *We have*

$$\inf_{x \in P} f(x) = \min_{F \in \mathcal{F}_B} \inf_{x \in F} f(x),$$

where the first infimum is attained if and only if the second infimum is attained. Furthermore, if the infimum is attained we have

$$\min_{x \in P} f(x) = \min_{F \in \mathcal{F}_A} \min_{x \in F} f(x) \qquad \text{and} \qquad \arg\min_{x \in P} f(x) \subseteq \bigcup_{F \in \mathcal{F}_C} F.$$

Theorem 1 clearly extends the fundamental theorem of Linear (and Concave) Programming. Furthermore it also extends the Frank-Wolfe theorem [3], which states that a quadratic function that is bounded below on a polyhedron P attains its minimum on P . Indeed, the Frank-Wolfe theorem can be easily obtained from Theorem 1, by observing that when f is quadratic the family of faces $\mathcal{F}_B$ defined above coincides with the family of faces of P where f is strictly convex. Furthermore, by coercivity, the minimum of a strictly convex quadratic function on a closed set always exists.

3 Optimization with box constraints, pseudo-Boolean optimization, and maximum independent set in a graph

A simple class of polyhedra to which Theorem 1 can be easily applied are the rectangles (or *box constraints*) $R = [\ell, u] = \{x \in \mathbb{R}^n : \ell \leq x \leq u\}$. In particular, without much loss in generality, we consider the case of the (continuous) unit hypercube $R = [0,1]^n$, and the corresponding discrete (or Boolean) hypercube $\mathbb{B}^n = \{0,1\}^n$. Let $N = \{1, \dots, n\}$. Given a real-valued function f on R, we define $I = \{i \in N : f$ is quasi-concave with respect to $x_i\}$. Furthermore, for every $K \subseteq I$ we set

$$\begin{aligned} R(K) &= \{x \in R : x_i = \ell_i \text{ for } i \in K \text{ and } x_i = u_i \text{ for } i \in I \setminus K\} \\ \psi(K) &= \min_{x \in R(K)} f(x). \end{aligned}$$

Theorem 2 *If f attains its minimum on R, the following equality holds:*

$$\min_{x \in R} f(x) = \min_{K \subseteq I} \psi(K). \tag{4}$$

Theorem 2 extends a result of Rosenberg for multilinear functions. It can be proved that if f is a submodular function on R, then $\psi(K)$ is a submodular function on the power set 2^I of all subsets of I. Since submodular functions can be minimized in polynomial time on power sets, we obtain the following.

Theorem 3 *If f is a submodular function on R, and $\psi(K)$ can be evaluated in polynomial time for every $K \subseteq I$, then f can be minimized in polynomial time over R.*

Note that any quadratic function of the form $f(x) = \frac{1}{2}x^T Cx + q^T x$, with $c_{ij} \leq 0$ for all $i, j \in N$ satisfies the assumptions of theorem 3 (with $I = N$).

Theorem 2 can also be used to provide a continuous formulation of the weighted maximum independent set problem in a graph. Our formulation includes as a special case the one proposed in [1] for the unweighted case. Let $G = (N, E)$ be a simple undirected graph, where $N = \{1, \dots, n\}$ is the set of nodes and E is the set of edges. A subset $S \subseteq V$ is an *independent set* if there is no edge joining nodes in S. Let w_i denote the weight of node i in N, and $w(S) = \sum_{i \in S} w_i$ the weight of the subset S of nodes. An independent set S has *maximum weight* if $w(S)$ is maximum among all independent sets. Let $\alpha_1, \dots, \alpha_n$ and $\beta_1, \dots, \beta_n$ be quasi-convex real-valued functions on [0,1] satisfying the following properties. For every $i = 1, \dots, n$, $\alpha_i(1) = \beta_i(0) = 0$, $\alpha_i(0) = \beta_i(1) = 1$, and $\alpha_i(x), \beta_i(x) > 0$, for all $x \in (0,1)$. Consider the function $f : [0,1]^n \to \mathbb{R}$ defined by $f(x) = \sum_{i \in N} w_i \alpha_i(x_i) \prod_{(i,j) \in E} \beta_j(x_j)$. Note that for every $x \in \{0,1\}^n$ we have $f(x) = \sum_{i \in N} w_i (1 - x_i) \prod_{(i,j) \in E} x_j$. To every subset $S \subseteq N$ we associate the point $x^S \in \{0,1\}^n$ defined by $x_i^S = 0$ iff $i \in S$ (x^S is the complement of the characteristic vector of S).

Theorem 4 *Let T be a maximum-weight independent set in G. Then*

$$f(x^T) = w(T) = \max_{x \in [0,1]^n} f(x) = \max_{x \in \{0,1\}^n} f(x). \tag{5}$$

4 Optimization over a simplex and the maximum clique problem

We now consider the problem of minimizing a function f over a simplex in $\mathbb{R}^n$. Also in this case we can restrict our attention, without much loss in generality, to a standard situation, i.e., the case where the feasible set is the *standard simplex* $\Delta = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1, x_i \geq 0, i = 1, \dots, n\}$. The special case where f is a quadratic function is already an NP-complete problem, known as the *standard quadratic optimization problem* [2].

Let $d_{ij} = e^i - e^j$, where e^k denotes the k-th unit vector in $\mathbb{R}^n$. Consider the graph $G = (N, E)$, where $N = \{1, \dots, n\}$ and E is the set of all pairs (i, j) such that f is quasi-concave on $\Delta_{d_{ij}}(x)$ for all x in Δ. To every subset Q of N we associate the face $F_Q = \{x \in \Delta : \sum_{x \in Q} x_i = 1\}$ of Δ. Note that the mapping $Q \mapsto F_Q$ is a bijection between the subsets of N and the faces of Δ. We also consider the weight function $W(Q) = \min\{f(x) : x \in F_Q\}$ defined on the subsets Q of N. Recall that a subset Q of N is a *clique* of G if every pair of nodes in Q are joined by an edge in E. A clique is *maximal* if it is not strictly contained in any larger clique. We denote by $\mathcal{C}$ the set of all cliques of G and by $\mathcal{C}^M$ the set of all maximal cliques of G. Given a real-valued function W(Q) on the subsets of N, a *maximum (minimum) weighted clique* is a clique that maximizes (minimizes) $W(Q)$ on the set $\mathcal{C}$ of all cliques of G. Note that there exists a maximum (minimum) weighted clique that is also a maximal clique if the function W is *isotone* (*antitone*), i.e. $W(Q) \leq W(Q')$ ($W(Q) \geq W(Q')$) whenever $Q \subseteq Q'$. In the classical maximum weighted clique problem the function W is modular, i.e., $W(Q) = \sum_{i \in Q} w_i$. The following result shows that the minimization problem on the standard simplex is equivalent to the minimum weighted clique problem on the associated graph. Furthermore, since our weight function is antitone, we can restrict the search for the global minima of f on Δ to those faces F_Q corresponding to maximal cliques Q of G.

Theorem 5 *We have*

$$\min_{x\in\Delta} f(x) = \min\{f(x) : x \in \bigcup_{Q\in\mathcal{C}^M} F_Q\} = \min_{Q\in\mathcal{C}} W(Q) = \min_{Q\in\mathcal{C}^M} W(Q).$$

Theorem 5 shows that the problem of minimizing a function on the standard simplex can be solved efficiently when the clique number of the associated graph is small. Furthermore, if we specialize Theorem 5 to a suitable class of quadratic functions we obtain the quadratic programming formulation of the maximum clique by Motzkin and Straus [5] and its weighted extension by Gibbons et al. [4]. We conclude by mentioning that if $f(x) = \frac{1}{2}x^T Cx + q^T x$, and C is a *Monge matrix* (i.e., for all $i < j$ and $h < k$, we have $c_{ih} + c_{jk} \leq c_{ik} + c_{jh}$) then, we can use Theorem 5 to show that the minimum of f on Δ is attained at one of its n vertices.

References

[1] J. Abello, S. Butenko, P. M. Pardalos, and M. G. C. Resende. Finding independent sets in a graph using continuous multivariable polynomial formulations. *J. Global Optim.*, 21(2):111–137, 2001.

[2] I. M. Bomze. On standard quadratic optimization problems. *J. Global Optim.*, 13(4):369–387, 1998.

[3] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95 – 110, 1956.

[4] L. Gibbons, D. Hearn, M. Ramana, and P. M. Pardalos. A continuous characterization of the maximum clique problem. *Mathematics of Operations Research*, 22(3):754–768, 1997.

[5] T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canadian Journal of Mathematics*, 17:533–540, 1965.

[6] I. G. Rosenberg. $0-1$ optimization and non-linear programming. *Rev. Française Automat. Informat. Recherche Opérationnelle*, 6(V-2):95–97, 1972.

On cliques associated to 3-set packing problems

Luis M. Torres [1]

Escuela Politécnica Nacional, Quito, Ecuador

Given a family $\mathcal{E}$ of n subsets of a ground set V, and a weight function $w : \mathcal{E} \to \mathbb{Q}_+$, the *set packing problem* SSP is the task of finding a subfamily $\mathcal{E}^* \subseteq \mathcal{E}$ of disjoints sets, for which $w(\mathcal{E}^*) := \sum_{E \in \mathcal{E}^*} w(E)$ is maximal. This problem can be formulated as the following integer program:

$$\text{(SSP)} \quad \begin{cases} \min w^T x \\ \text{s.t.} \\ \quad Ax \leq \mathbf{1}, \\ \quad\quad x \in \{0,1\}^n. \end{cases}$$

where $A \in \{0,1\}^{m \times n}$ is a 0/1-matrix whose columns are the incidence vectors of the members from $\mathcal{E}$ (assuming $m := |V|$), and each binary variable x_i corresponds to a set $E_i \in \mathcal{E}$ having weight $w_i := w(E_i)$.

In the κ-set packing problem (κ-SSP), we additionally require all sets in $\mathcal{E}$ to have cardinality less than or equal to κ, or, equivalently, we require the matrix A to have at most κ nonzero entries per column. One can prove that 1-SSP is a trivial problem, and that 2-SSP is equivalent to a matching problem. Meanwhile, 3-SSP is already $\mathcal{NP}$-complete (see e.g. Garey & Johnson [5] for a reduction from 3-SAT).

Following a usual approach in polyhedral combinatorics, we consider in the following the *set packing polytope* $P_I(A)$, defined by the convex hull of all feasible solutions to an instance of κ-SSP, and its *fractional relaxation* $P(A)$:

$$\begin{aligned} P_I(A) &:= \text{conv } \{x \in \{0,1\}^n : Ax \leq \mathbf{1}\}, \\ P(A) &:= \{x \in \mathbb{R}^n_+ : Ax \leq \mathbf{1}\}. \end{aligned}$$

Edmonds [3] suggested for the first time that SSP can also be expressed in graph theoretic terms. Namely, given the integer programming formulation of the problem, we define the *intersection or conflict graph* $\mathbb{G}(A) = (\mathbb{V}, \mathbb{E})$ of A as a graph having one node for each column of A, and an edge between each pair of nodes whose corresponding columns intersect at least in one row. Moreover, weights associated with the columns of A are transmitted to the nodes in $\mathbb{G}$. The set packing problem is then equivalent to the problem of finding a maximum-weight stable set in $\mathbb{G}(A)$. Padberg [7, 8] showed that this fact can be exploited to obtain valid (and even facet-defining) inequalities for

[1] E-mail: `torres@zib.de`.

$P_I(A)$ from certain structures (node-induced subgraphs) in $\mathbb{G}(A)$. During the last three decades, this has been the most successful approach for obtaining results concerning the facetial structure of the set packing polytope (see e.g. Borndörfer [1] for a survey).

Our interest has been to study how the cardinality restriction in the κ-set packing problem is reflected both on the packing polytope and on the conflict graph. In this paper, we are going to consider some results related to one specific structure in the conflict graph of 3-set packing problems. Observe that introducing redundant restrictions of the form $x_j \leq 1$, it is always possible to reduce this problem to the case where all columns of A have *exactly* three ones. Therefore, we call the sets in $\mathcal{E}$ just *triangles*, and the elements of the ground set V just *points*. Similarly, A is the *point-triangle* incidence matrix and the linear inequalities defining the fractional polytope $P(A)$ are the *point constraints.*

Historically, one of the first graph structures studied in the context of the stable set polytope was the *clique*: a subgraph consisting of mutually adjacent nodes. If Q is the set of vertices of a clique in $\mathbb{G}(A)$, then the inequality

$$\sum_{i \in Q} x_i \leq 1$$

is (trivially) valid for $P_I(A)$. Moreover, (as noticed by Fulkerson [4] and Padberg [7]) this inequality defines a facet of the set packing polytope if and only if the clique is maximal with respect to node inclusion. Note that by definition all rows in A correspond to cliques in $\mathbb{G}(A)$, but the converse is generally not true.

Grötschel et al. [6, Section 9.2, page 283] showed that the separation problem for clique inequalities is $\mathcal{NP}$-complete. At the same time, the authors proved that these inequalities belong to a larger class of polynomially separable *orthonormal representation constraints.* Graphs for which the stable set polytope is completely described by clique and nonnegativity inequalities are called *perfect graphs*, and if A is a clique-node incidence matrix of a perfect graph, then $P_I(A) = P(A)$ holds. A matrix having this property is called a *perfect matrix.*

In conflict graphs related to 2-SSP, cliques must stem from one of the two structures showed in Figure 0.1. Here, edges represent the 2-sets from $\mathcal{E}$ and nodes the elements of the ground set V. Observe that the conflict graphs are obtained in this case as the line graphs of the structures showed in the figure. Moreover, remark that the clique inequality associated to the second case is exactly the point inequality corresponding to the point 1 (the center of the star). Hence, the only inequalities which might be violated by points of $P(A)$ come from structures like the first one. Since these structures involve only three variables, it follows that clique inequalities can be separated in polynomial time for 2-SSP.

Unfortunately, the situation is not so easy in 3-SSP, and one can give examples of arbitrarily large cliques that are associated to violated clique inequalities. However, it is still possible to separate clique inequalities in polynomial time. This is a consequence of the following theorem that we have proven:

Theorem 1 (Cliques of Triangles) *Let $\mathcal{Q} \subseteq \mathcal{E}$ be a maximal clique of triangles. Then one of the two following statements hold:*

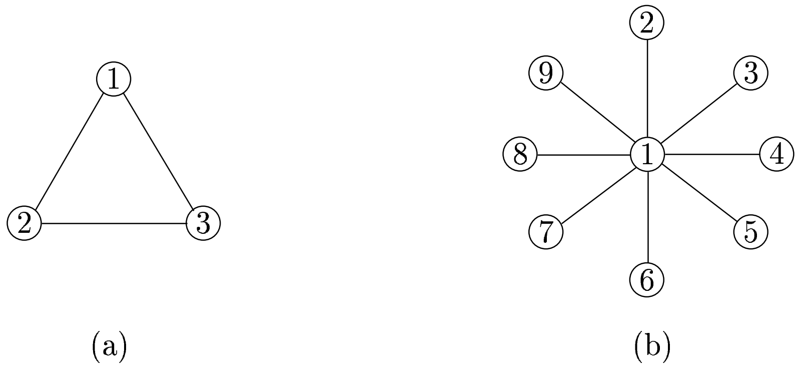


Figure 0.1. Cliques in 2-SSP. The only possible structures giving rise to cliques in the conflict graph are (a) the K_3 and (b) the stars.

(i) There exists a set $S \subseteq V$, with $|S| = 3$ such that

$$\mathcal{Q} = \{E \in \mathcal{E} : |S \cap E| \geq 2\},$$

(ii) there exists a point $v \in V$ with the property

$$|\{E \in \mathcal{Q} : v \notin E\}| \leq 5.$$

Another issue we looked at are fractional vertices in $P(A)$ associated to violated clique inequalities. We have proven that such vertices can have maximal seven nonintegral coordinates and enumerated all possible configurations for the corresponding cliques using a computer. The set consisting of all nonintegral coordinates must be equal to one of the following:

$$\begin{aligned}
X_1 &= \left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right\}, & X_2 &= \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\}, \\
X_3 &= \left\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right\}, & X_4 &= \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}\right\}, \\
X_5 &= \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\}, & X_6 &= \left\{\frac{1}{5}, \frac{1}{5}, \frac{2}{5}, \frac{2}{5}, \frac{3}{5}\right\}, \\
X_7 &= \left\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}\right\}, & X_8 &= \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\}, \\
X_9 &= \left\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right\}.
\end{aligned}$$

There is only one structure associated to vertices of the class X_9. It consists of seven triangles and seven points, configured as shown in the matrix from Figure 0.2. The reader can verify that this structure is combinatorially equivalent to the *Fano plane* depicted in the figure.

Moreover, there are several similarities between this structure and the K_3 considered above when discussing 2-SSP :

(i) Both cliques cannot be further extended.

$$\begin{pmatrix} 1 & 1 & 1 & . & . & . & . \\ 1 & . & . & 1 & 1 & . & . \\ 1 & . & . & . & . & 1 & 1 \\ . & 1 & . & 1 & . & 1 & . \\ . & 1 & . & . & 1 & . & 1 \\ . & . & 1 & 1 & . & . & 1 \\ . & . & 1 & . & 1 & 1 & . \end{pmatrix}$$

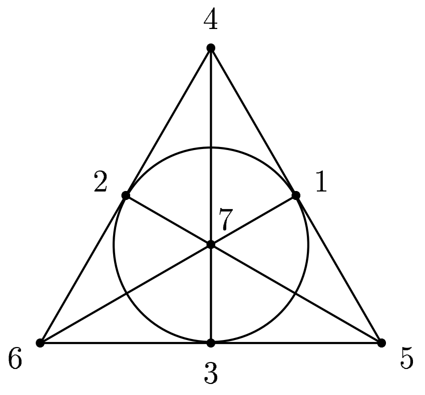


Figure 0.2. Clique of triangles associated to the vertex class X_9. The clique contains 7 triangles and 7 points, configured according to the point-triangle incidence matrix shown. Representing triangles by lines, the structure turns out to be isomorphic to the Fano plane.

(ii) The number of sets is equal to the number of points.

(iii) The cliques are *regular*: Each point appears in exactly κ sets.

(iv) For any two points, there is exactly one set containing both of them.

(v) The cliques are *self-dual* in the sense of hypergraph duality.

For $\kappa \geq 3$, such structures are called *finite projective planes* of order $\kappa - 1$ and one fundamental unresolved conjecture in combinatorics (see Bruck & Ryser [2]) states that they exist only for $\kappa = p^a + 1$, where p is a prime number and $a \in \mathbb{N}$.

References

[1] Borndörfer (1998). *Aspects of Set Packing, Partitioning and Covering.* Ph.D. dissertation, Tech. Univ. of Berlin, Berlin.
[2] Bruck & Ryser (1949). The Nonexistence of Certain Finite Projective Planes. *Canad. J. Math. 1, 88–93.*
[3] Edmonds (1962). Covers and Packings in a Family of Sets. *Bulletin of the Am. Math. Soc. 68, 29–32.*
[4] Fulkerson (1971). Blocking and Anti-Blocking Pairs of Polyhedra. *Math. Prog. 1, 168–194.*
[5] Garey & Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W. H. Freeman and Company.
[6] Grötschel, Lovász & Schrijver(1988). *Geometric Algorithms and Combinatorial Optimization.* Springer Verlag, Berlin.
[7] Padberg (1973a). On the Facial Structure of Set Packing Polyhedra. *Math. Prog. 5, 199–215.*
[8] —— (1973b). Perfect Zero-One Matrices. *Math. Prog. 6, 180–196.*

Subdivision of the hierarchy of H-colorable graph classes by circulant graphs [1]

Akihiro Uejima [a,2] and Hiro Ito [a,3]

[a] *School of Informatics, Kyoto University Yoshida-honmachi, Sakyo-ward, Kyoto 606-8501 Japan*

Abstract

For any integer $p \geq 2$, p-colorable graphs are $\overline{C_{2p+1}}$-colorable and $\overline{C_{2p+1}}$-colorable graphs are $p+1$-colorable, where $\overline{C_{2p+1}}$ is the complement graph of a cycle of order $2p+1$. The converse statements are however incorrect. This paper presents that the above inclusion can be subdivided by a subset of circulant graphs $H(n,k)$ $(n,k \in \boldsymbol{N},\ k \leq \lfloor n/2 \rfloor)$. The subdivided hierarchy of inclusion contains the well-known inclusion of C_{2p+1}-colorable graphs. Moreover, we prove some NP-complete problems for planar $H(n,k)$-colorings, including the $\overline{C_5}$-coloring.

Key words: H-coloring; Color-family; Circulant graphs; time complexity

1 Introduction

For a given graph G with vertex set $V(G)$ and edge set $E(G)$, a *p-coloring* of G is a mapping of $V(G)$ to $\{1,2,\ldots,p\}$ such that no two vertices on the same edge receive the same color. Given graphs G and H, a *homomorphism* f of G to H is an edge preserving mapping of $V(G)$ to $V(H)$, i.e., a mapping f of $V(G)$ to $V(H)$ such that $f(x)$ and $f(y)$ are adjacent vertices of H if x and y are adjacent vertices of G. Such a homomorphism is also called an *H-coloring of G*. For any fixed graph H, *H-coloring problem* is deciding whether there is an H-coloring of a given input graph G. We say that G is *H-colorable* if an H-coloring exists. A complete graph of order n is denoted by K_n. Then, the problem of deciding whether G is K_p-colorable is the problem of deciding whether G is p-colorable. Thus, H-coloring problem is a natural generalization of the traditional graph colorings, and it has been studied in various contexts [2, 5, 6, 7, 8, 9]. The interconnections between homomorphisms and the theory of grammar forms were investigated in [5], and H-coloring problems include *T-colorings* and problems related to *channel assignment problems* [6].
Equivalently, H-coloring problem may be considered as a decision problem related to a class $\mathcal{L}(H) := \{G \mid G \text{ is } H\text{-colorable}\}$. Such classes are called *color-families* and their structure has been an important theme, e.g., Theorems 3 and 4, which will be shown later. Some terms are

[1] This research is supported by The 21st Century COE Program and the Scientific Grant-in-Aid from Ministry of Education, Science, Sports and Culture of Japan.
[2] uejima@lab2.kuis.kyoto-u.ac.jp
[3] itohiro@kuis.kyoto-u.ac.jp

Table 11
Inclusion on $H(n,k)$ and relation with K_p, C_q, and $\overline{C_r}$

K_2		C_9	C_7	C_5		K_3		$\overline{C_7}$	
$H(2,1)$	$\subset$					$H(3,1)$	$\subset \cdots$		
$H(4,2)$	$\subset$			$H(5,2)$	$\subset$	$H(6,2)$	$\subset$	$H(7,2)$	$\subset \cdots$
$H(6,3)$	$\subset$		$H(7,3)$	$\subseteq H(8,3) \subset$		$H(9,3)$	$\subset H(10,3) \subseteq H(11,3) \subset \cdots$		
$H(8,4)$	$\subset$	$H(9,4)$	$\subset$	$H(10,4)$	$\subset H(11,4) \subset$	$H(12,4)$	$\subset H(13,4) \subset$	$H(14,4)$	$\subset H(15,4) \subset \cdots$
$\cdots$	$\cdots$					$\cdots$	$\cdots$		

defined for explaining the statements we show, as follows. We say that two graphs H and H' are *homomorphically equivalent* if H-colorable and H'-colorable are equivalent for any graph. A graph H is a *core* if it does not admit a homomorphism to a proper subgraph. For example, all complete graphs, odd cycles, and complements of odd cycles are cores. It was proven in [9] that every graph H contains a unique (up to isomorphism) subgraph H' which is a core and admits a homomorphism of H to H'; we call H' the *core of* H. For positive integers n and $1 \leq a_1, a_2, \ldots, a_k \leq \lfloor n/2 \rfloor$, the *circulant graph* $(n;\, a_1, a_2, \ldots, a_k)$ has the vertex set $\{0, 1, \ldots, n-1\}$, and two vertices x and y are adjacent if and only if $y - x \pmod n$ or $x - y \pmod n \in \{a_1, a_2, \ldots, a_k\}$.
We mainly prove two theorems for the subclass of circulant graphs $H(n,k) := (n; k, k+1, \ldots, \lfloor n/2 \rfloor)$ for $1 \leq k \leq \lfloor n/2 \rfloor$ as follows. Corollary 1.1 is obtained from Theorem 1. ($X \subset Y$ denotes that X is a proper subset of Y, C_p is a cycle of order p, and $\overline{H}$ is the complement graph of a graph H.)

Theorem 1 *(see Tab. 11) For any integers $p \geq 2$, n, $k \geq 1$,*

- *$\mathcal{L}(H(n,k)) \subseteq \mathcal{L}(H(n+1,k))$,*
- *$\mathcal{L}(H(pk-1,k)) \subset \mathcal{L}(H(pk,k)) \subset \mathcal{L}(H(pk+1,k))$ and K_p is the core of $H(pk,k)$, and*
- *$\mathcal{L}(H((2p+1)k-1,2k)) \subset \mathcal{L}(H((2p+1)k,2k)) \subset \mathcal{L}(H((2p+1)k+1,2k))$ and $\overline{C_{2p+1}}$ is the core of $H((2p+1)k,2k)$.*

Corollary 1.1 *For any integers $p \geq 2$, $\mathcal{L}(H(4p,4)) \subset \mathcal{L}(H(4p+1,4)) \subset \mathcal{L}(H(4p+2,4)) \subset \mathcal{L}(H(4p+3,4)) \subset \mathcal{L}(H(4(p+1),4))$. Moreover, K_p and $\overline{C_{2p+1}}$ are the cores of $H(4p,4)$ and $H(4p+2,4)$, respectively.*

Theorem 2 *Planar $H(8,4)$-coloring problem is in P. For any $9 \leq n \leq 15$, planar $H(n,4)$-coloring problem is NP-complete. Otherwise, it is in $O(1)$ to solve. Planar C_{2p+1}-coloring problem is NP-complete for any integer $p \geq 2$.* ∎

Color-families have the already-known basic inclusions as follows (see Tab. 11). The inclusion in Theorem 1 is a subdivided hierarchy of these inclusions.

Theorem 3 *[5] For any integers $p \geq 1$, $q \geq 2$, $\mathcal{L}(K_p) \subset \mathcal{L}(K_{p+1})$. Moreover, $\mathcal{L}(K_2) \subset \mathcal{L}(C_{2q+1}) \subset \mathcal{L}(K_3) = \mathcal{L}(C_3)$ and $\mathcal{L}(C_{2q+1}) \subset \mathcal{L}(C_{2q-1})$.*

Theorem 4 *[7] For any integer $p \geq 2$, $\mathcal{L}(K_p) \subset \mathcal{L}(\overline{C_{2p+1}}) \subset \mathcal{L}(K_{p+1})$.* ∎

For general graphs, the complexity of H-coloring problem is well-known, the problem is in P if H is bipartite, otherwise it is NP-complete [2]. However, the proofs in [2] cannot be directly extended for planar graphs and it is still open. Needless to say, the problem is in P if H is bipartite. The situation is well-understood for complete graphs: For any fixed $p \geq 4$ or $p = 1$, the K_p-coloring

problem is easy to solve [3], and K_2-coloring problem is in P. K_3-coloring problem is NP-complete [1].
In this paper, we are also interested in studying the time complexity of H-coloring of planar graphs and analyze the threshold graph H such that planar H-coloring problem changes from P to NP-complete preserving a inclusion of color-families. For this purpose, subdividing the inclusion of $\mathcal{L}(H)$, e.g., the inclusions in Theorems 3 and 4 into the one in Theorem 1, is not only elegant but also useful for analyzing such a threshold graph H.
For the inclusion in Theorem 4, the most part of the complexity of planar $\overline{C_{2p+1}}$-coloring problems are solved. For any $p \geq 4$, it is in P since $\mathcal{L}(K_4) \subset \mathcal{L}(\overline{C_{2p+1}})$. For $p = 3$, we presented that $\overline{C_7}$-coloring problem is NP-complete [8]. The case $p = 2$ is only open (note that it is shown in Theorem 2.). We make the time complexity of planar $H(n,4)$-coloring problems clear for the new inclusion in Corollary 1.1 and show that $\mathcal{L}(K_2) \subset \mathcal{L}(C_{2p+1})$, $|\mathcal{L}(C_{2p+1}) - \mathcal{L}(K_2)|$ is boundlessly small, and planar C_{2p+1}-coloring problem is NP-complete for positive infinity p in Theorem 2.
We introduce some terms and notations as follows. Consider a graph G and a vertex subset $V' \subseteq V(G)$. A graph $(V', \{(x,y)|x,y \in V', (x,y) \in E(G)\})$ is called an *induced subgraph* of G by V'. An induced subgraph of G by $V(G) - V'$ is denoted by $G - V'$. If $(x,y) \in E(G)$ (resp., $(x,y) \notin E(G)$) for any vertex pair $x, y \in V'$, V' is called a *clique* (resp., an *independent set*). For a graph G, the maximum order of its cliques is denoted by $\omega(G)$. An *elementary homomorphism* in a graph H consists of contracting two non-adjacent vertices x and y in H. A graph H' is a *morphic image* of a graph H if it is obtained from H by finitely many elementary homomorphisms. H is also considered to be a morphic image of itself.

Theorem 5 *[5] A graph G is H-colorable if and only if a morphic image G' of G is isomorphic to a subgraph of H.*

2 Proof of Theorem 1

Theorem 1 can be directly obtained by the following lemmas.

Lemma 1 *For any integers $n \geq 2$, $1 \leq k \leq \lfloor n/2 \rfloor$, $\mathcal{L}(H(n,k)) \subseteq \mathcal{L}(H(n+1,k))$.*

Proof. Let x be a vertex of $V(H(n+1),k)$. $H(n,k)$ is a proper subgraph of $H(n+1,k) - \{x\}$. Thus, $H(n,k)$ is $H(n+1,k)$-colorable, so all $H(n,k)$-colorable graphs are $H(n+1,k)$-colorable. ∎

Lemma 2 *For any integers $n, r \geq 2$, $1 \leq k \leq \lfloor n/2 \rfloor$, $H(rn,rk)$ is not a core. That is, $H(n,k)$ and $H(rn,rk)$ are homomorphically equivalent and $H(n,k)$ is a proper subgraph of $H(rn,rk)$.*

Proof. A induced subgraph of $H(rn,rk)$ by $\{0, r, 2r, \dots, (n-1)r\}$ is $H(n,k)$. Thus, $H(n,k)$ is $H(rn,rk)$-colorable.
On the other hand, $V_i := \{ri, ri+1, ri+2, \dots, r(i+1)-1\}$ for $i \in \{0,1,\dots,n-1\}$ (see Fig. 2.1(a)). Then, each V_i is a independent set. There is no adjacent vertex-pair between V_i and $V_{i\pm j}$ if $j \in \{1,2,\dots,k-1\}$, otherwise there is at least one adjacent vertex-pair between them. Hence, a morphic image of $H(rn,rk)$ constructed by contracting each V_i is $H(n,k)$. From Theorem 5, $H(rn,rk)$ is $H(n,k)$-colorable. ∎

Lemma 3 *For any integers $p \geq 2$, $1 \leq k \leq \lfloor n/2 \rfloor$, $\mathcal{L}(H(pk-1,k)) \subset \mathcal{L}(H(pk,k)) \subset \mathcal{L}(H(pk+1,k))$ and $\mathcal{L}(H((2p+1)k-1,2k)) \subset \mathcal{L}(H((2p+1)k,2k)) \subset \mathcal{L}(H((2p+1)k+1,2k))$.*

Proof. We show a sketch of proof about $\mathcal{L}(H(pk-1,k)) \subset \mathcal{L}(H(pk,k)) \subset \mathcal{L}(H(pk+1,k))$ (the remaining proof is omitted due to limitations of space).
The case $\mathcal{L}(H(pk-1,k)) \subset \mathcal{L}(H(pk,k))$: From Lemma 2, a core of $H(pk,k)$ is K_p. $\omega(H(pk-1,k)) = \lfloor (pk-1)/k \rfloor = p-1 < p$, hence $H(pk,k)$ is not $H(pk-1,k)$-colorable.
The case $\mathcal{L}(H(pk,k)) \subset \mathcal{L}(H(pk+1,k))$: Assume that $H(pk+1,k)$ is K_p-colorable ($V(K_p) := \{0,1,\dots,p-1\}$). Then, a induced subgraph of $H(pk+1,k)$ by $\{0,k,2k,\dots,(p-1)k\}$ is K_p. Thus, we can assume that a homomorphism f of $H(pk+1,k)$ of $H(pk,k)$ such that $f(x) = \frac{x}{k}$, $x \in \{0,k,\dots,(p-1)/k\}$ without loss of generality. The one by $\{0,k,2k,\dots,(p-2)k,(p-1)k+1\}$ is also K_p, so $f((p-1)k+1) = p-1$ (see Fig. 2.1). After $pk+1$ steps of same operation, $f(0) = p-1$, it contradicts the assumption $f(0) = 0$. ∎

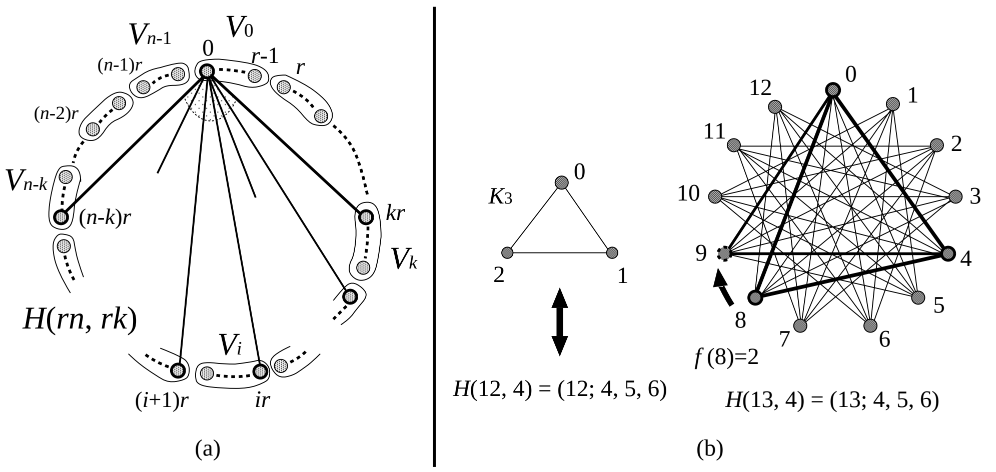


Figure 2.1. Sketch of proofs

3 Proof of Theorem 2

For this proof, we can reduce Planar 3SAT [4] to the problems (the proofs are omitted). The reductions are similar to the reduction used to prove that planar 3-coloring problem is NP-complete [1].

4 Concluding remarks

This paper was interested in studying the time complexity of H-coloring of planar graphs. We mainly showed a new inclusion of $H(n,k)$-colorable graphs and the time complexity of some planar $H(n,k)$-coloring problems.

References

[1] M. R. Garey, D. S. Johnson, & L. Stockmeyer, "Some simplified NP-complete graph problems," Theoretical Computer Science, **1**, pp. 237-267, 1976.

[2] P. Hell, & J. Nesetril, "On the Complexity of H-Coloring," Journal of Combinatorial Theory, **B 48**, pp. 92-110, 1990.

[3] T. R. Jensen, & B. Toft, *Graph Coloring Problems*, John Wiley & Sons, New York, 1995.

[4] D. Lichtenstein, "Planar formulae and their uses," SIAM J. Comput., **11**, no. 2, pp. 329-343, 1982.

[5] H. A. Maurer, A. Salomaa, & D. Wood, "Colorings and interpretations : A connection between graphs and grammar forms," Discrete Applied Mathematics, **3**, pp. 119-135, 1981.

[6] F. S. Roberts, "T-colorings of graphs: recent results and open problems," Discrete Math., **93**, pp. 229–245, 1991.

[7] A. Uejima, and H. Ito, "On H-coloring problems with H expressed by complements of cycles, bipartite graphs, and chordal graphs," IEICE Transactions, vol. E85-A, no. 5, pp. 1026–1030, 2002.

[8] A. Uejima, H. Ito, and T. Tsukiji, "$\overline{C_7}$-coloring problem," IEICE Transactions, 2004 (to appear).

[9] E. Welzl, "Color families are dense," Theoret. Comput. Sci., **17**, pp. 29-41, 1970.

To be or not to be Yutsis

D. Van Dyck,[1] V. Fack[2]

Ghent University, Department of Applied Mathematics & Computer Science, Research Group Combinatorial Algorithms and Algorithmic Graph Theory, Krijgslaan 281-S9, 9000 Ghent, Belgium

1 Introduction

A binary coupling tree on $n+1$ leaves is an unordered binary tree in which each leaf has a distinct label. A Yutsis graph of order n consists of two binary coupling trees on $n+1$ leaves, in which the unique leaf edges with the same label are identified. In addition both root nodes are connected by an additional edge. The leaf nodes themselves disappear from the graph. Figure 1.1 shows an example. The graph thus obtained is cubic and has $2n$ nodes and $3n$ edges. Moreover it has the property that it contains an edge-cut on $n+2$ edges that separates the graph into two trees of equal size, which we call a *defining cut*. The trees are called *defining trees*.

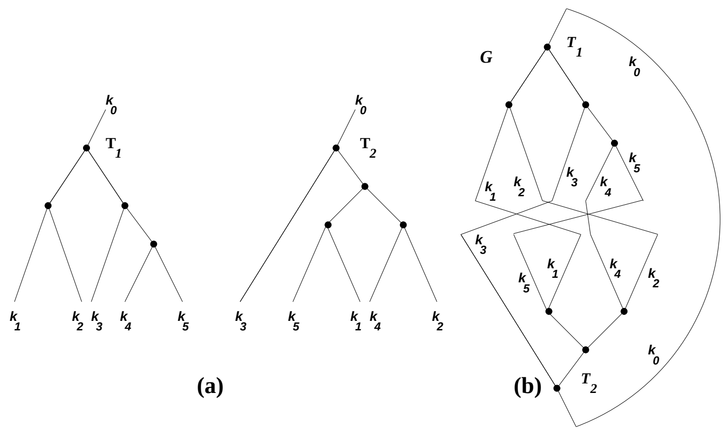

k_0
T_1
k_1 k_2 k_3 k_4 k_5
k_0
T_2
k_3 k_5 k_1 k_4 k_2
(a)
G
T_1
k_0
k_1 k_2 k_3 k_4 k_5
k_3 k_5 k_1 k_4 k_2
k_0
(b)
T_2

Figure 1.1. (a) Two binary coupling trees T_1 and T_2, and (b) the corresponding Yutsis graph of order 4

Yutsis graphs appear in the context of quantum theory of angular momenta, where they represent a $3nj$-coefficient. The binary coupling trees correspond to the coupling schemes in the bra/kets of the $3nj$-coefficient [BL81, YLV62]. Yutsis graphs are used to calculate a summation formula for such a $3nj$-coefficient [VF04].

[1] E-mail: `Dries.VanDyck@UGent.be`.
[2] E-mail: `Veerle.Fack@UGent.be`.

So far no better method is known to determine whether a cubic graph is Yutsis than searching for a defining tree (or cut). In this article we will tackle the decision problem whether a given cubic graph is Yutsis or not. For the quantum theory of angular momenta, we are interested in obtaining large testcases by generating large cubic graphs at random and filter out those graphs which are not Yutsis. In addition we would like to identify the non Yutsis graphs and study their structure. Yutsis graphs cannot contain a bridge by construction, so from now on we only consider bridgeless cubic graphs.

2 The decision problem is NP-Complete

We prove the stronger result that deciding whether a given cubic graph is Yutsis or not is NP-complete when restricted to the subclass of cubic polyhedra, i.e. 3-connected planar cubic graphs. This of course implies the weaker result for general cubic graphs.

Lemma 1 *A cubic planar graph is Yutsis iff its dual planar graph is Hamiltonian.*

Chvátal and Wigderson proved, independent of each other, that finding a Hamiltonian cycle is a NP-complete problem, even when restricted to triangulations [Chv85, Wig82]. We prove that the Yutsis decision problem is NP-complete by showing that the decision problem whether a given triangulation contains a Hamiltonian cycle can be transformed in polynomial time to the Yutsis decision problem and that a solution for the Yutsis decision problem, by means of a defining tree, can be checked in polynomial time.

Theorem 1 *The decision problem whether a given cubic graph is Yutsis or not is NP-complete for cubic polyhedra, i.e. cubic 3-connected planar graphs.*

3 A local search approximation algorithm

In order to perform local search for a given problem, one has to define the set of problem instances $\mathcal{O}$, a cost function f and a neighbourhood function N. Let, for the following definitions, $G = (V, E)$ be a cubic graph with $2n$ nodes and $3n$ edges.

Definition 1 *We define the set of problem instances $\mathcal{O}(G)$ for G as the set of $\binom{2n}{n}$ possible subsets of V of size n.*

Let from this point on $V_1 \in \mathcal{O}(G)$ and $\overline{V}_1 = V \setminus V_1$.

Definition 2 *We define a neighbourhood function N as follows:*

$$N : V_1 \mapsto N(V_1) = \{V_1 \setminus \{v\} \cup \{\overline{v}\} | v \in V_1 \wedge \overline{v} \notin V_1\}\,.$$

n	#Yutsis	$r=1$	$r=2$	$r=3$	$r=4$	$r=n$
6	80	76	80			
7	475	424	463	471	475	
8	3 836	3 355	3 673	3 775	3 804	3 832
9	39 555	33 299	37 728	38 795	39 166	39 521
10	495 045	412 231	469 052	483 781	489 379	494 584
11	7 159 696	5 857 400	6 749 785	6 849 959	7 066 865	7 153 345

Table 12
The number of Yutsis graphs recognized using r restarts. The second column shows the number of Yutsis graphs on $2n$ nodes.

Define an *edge cut* to be the set of edges going from one partition to the other in a partioning $[V_1, \overline{V}_1]$ of the nodeset V. Counting edges, it is easily shown that every edge cut on $n+2$ edges separating connected components is a defining cut. This idea is the basis for our cost function f:

Definition 3 *A cost function f can be defined as follows:*

$$f : V_1 \mapsto |n+2-k| + c_1 + \overline{c}_1 - 2,$$

with k the number of edges on the cut $[V_1, \overline{V}_1]$, and c_1 (respectively $\overline{c}_1$) the number of components of V_1 (respectively $\overline{V}_1$).

This function evaluates to 0 i.f.f. $[V_1, \overline{V}_1]$ is a defining cut. Call f^* the global minimum of $f(V_1)$ for $V_1 \in \mathcal{O}(G)$ and f_{min} the absolute minimum of f over all $\mathcal{O}(G)$ for all cubic graphs G on $2n$ nodes. Clearly $f^* = f_{min} = 0$ for a given cubic graph G i.f.f. G is Yutsis.

We will show that the time complexity of the basic algorithm is $O(n^4)$.

The probability for success is highly influenced by the choice of the initial problem instance $V_1 \in \mathcal{O}(G)$. Since the shape of an initial solution depends on the used labeling, it is hard to obtain a good initial V_1 in acceptable time. For this reason we choose an initial problem instance at random and provide an option to restart the algorithm with a new random V_1 when the local search ends in a local optimum $\hat{o}$ s.t. $f(\hat{o}) \neq f_{min}$. This option makes the algorithm behave like a Monte-Carlo algorithm: it is possible that we get a wrong negative answer, i.e. the algorithm *can* conclude that a given Yutsis graph is not Yutsis. We can lower this probability by augmenting the number of restarts r with a new random V_1, which is a typical property of a Monte-Carlo algorithm.

Running the algorithm on all Yutsis graphs unpto $n = 11$ and on sets of larger randomly generated cubic graphs we obtained the results shown in Table 12 and Table 13. As one can see, this approach is quite succesfull considering its cost.

n	#Yutsis	$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = n$
10	995	869	964	986	987	995
20	2 000	1 493	1 842	1 939	1 965	2 000
30	2 999	2 080	2 659	2 874	2 942	2 999
50	500	299	412	442	483	500
100	1 000	455	696	828	888	1 000
200	2 000	586	971	-	-	-

Table 13
The number of Yutsis graphs recognized using r restarts for $100n$ ($n \leq 30$) or $10n$ randomly generated graphs ($n > 30$). The second column shows the number of Yutsis graphs in these sets.

n	Yutsis/cubic	n	Yutsis/ cubic	n	Yutsis/ cubic
5	18/ 18	8	3 836/ 3 874	11	7 159 696/ 7 187 627
6	80/ 81	9	39 555/ 39 866	12	116 040 456/ 116 349 635
7	475/ 480	10	495 045/497 818	13	2 068 782 009/2 072 540 352

Table 14
Cubic (bridgeless) graphs vs. Yutsis graphs

4 Which graphs are not Yutsis?

Using a fast exhaustive algorithm [VF03c], we obtained the number of Yutsis graphs within the set of bridgeless cubic graphs, shown in Table 14. Clearly non Yutsis graphs are rare and the results even propose that asymptotically allmost all cubic graphs are Yutsis. Nevertheless non Yutsis graphs of arbitrary large girth do exist, even 3-connected ones:

Theorem 2 *For every $g > 2$ there exists a 3-connected non Yutsis graph with girth g.*

The proof is constructive and based on the existence of cubic cages of arbitrary large girth [ES63] and the fact that all cages are 3-connected [FHR97]. In [Jae74] Jaegar proved for the planar case that all cyclically 4-connected cubic graphs have a Hamiltonian cycle in their planar dual and thus are Yutsis. In the same article Jaeger conjectures that all cyclically 4-connected cubic graphs are Yutsis. This conjecture is still open.

In addition we present some graph classes which are not Yutsis.

References

[BL81] L.C. Biedenharn and J.D. Louck, “Coupling of n angular momenta: recoupling theory”, in: *The Racah-Wigner Algebra in Quantum Theory, Encyclopedia of Mathematics and its Appli-*

cations, Vol. 9, pp. 435–481 (Addison-Wesley, 1981).

[YLV62] A.P. Yutsis, I.B. Levinson and V.V. Vanagas, *Mathematical Apparatus of the Theory of Angular Momentum*, (Israel Program for Scientific Translation, Jerusalem, 1962).

[VF04] D. Van Dyck, V. Fack, "On the Reduction of Yutsis Graphs", accepted by *Discrete Mathematics* (2004).

[Chv85] V. Chvátal, "Hamiltonian cycles", in: E. L. .Lawler, J. .K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, *The Traveling Salesman Problem*, pp. 403–429 (John Wiley, 1985).

[Wig82] A. Wigderson, "The complexity of the hamiltonian circuit problem for maximal planar graphs", *Technical Report Computer Science Department Princeton University* **298** (1982).

[VF03c] D. Van Dyck, V. Fack, "A fast algorithm for filtering Yutsis graphs", submitted to *Journal of Graph Algorithms and Applications* (2003).

[ES63] P. Erdos, H. Sachs, "Reguläre graphen gegebener Taillenweite mit minimaler Knotenzahl.", *Wiss. Z. Uni. Halle (Math. Nat.)* **12** (1963) 251–257.

[FHR97] H. L. Fu, K. C. Huang, C. A. Rodger, "Connectivity of Cages", *Journal of Graph Theory* **24** (1997) 187–191.

[Jae74] F. Jaeger, "On vertex-induced forests in cubic graphs", Proceedings 5th Southeastern Conference, Congressus Numerantium (1974) 501–512.

The complexity of arc-colorings for directed hypergraphs

Andrea Vietri [1]

Dipartimento Me.Mo.Mat., Università di Roma1, Via A. Scarpa 16, 00161 Roma, Italia

The notion of *directed hypergraph* is not univocally defined in the literature. In the present abstract we are referring to [1, 2].

Definition 1 *A directed hypergraph $H = (V(H), E(H))$ consists of a set V, the nodes, together with a set $E \subseteq \mathcal{P}(V) \times V$. Each element $\varepsilon = (A, z)$ of E is a hyperarc (or simply an arc). A and z are respectively the tail and the head of ε. If $v \in V$, the degree of v is $\delta(v) := \max(|\{(A,z) \in E : v \in A\}|, |\{(A,z) \in E : v = z\}|)$. The degree of H is $\Delta(H) := \max_{v \in V}(\delta(v))$.*

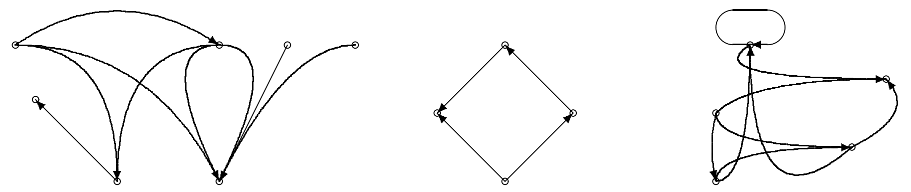


Figure 0.1. Directed hypergraphs

Computer science has often invoked directed hypergraphs, in connection with database theory and functional dependencies among attributes [1], as well as with artificial intelligence, deductive databases, diagnostic, logic [2].

The following notion of arc-coloring for directed hypergraphs has been introduced in [11]. In the case of digraphs, such coloring coincides with the arc-coloring defined and studied in [6, 10].

Definition 2 *An arc-coloring of H is a map $\gamma : E(H) \to \mathbf{N}$ such that*

i) $(\ (A,x) \in E, (B,y) \in E, A \cap B \neq \emptyset\ ) \Rightarrow \gamma(A,x) \neq \gamma(B,y)$

ii) $(\ (C,z) \in E, (D,z) \in E\ ) \Rightarrow \gamma(C,z) \neq \gamma(D,z)$

provided $(A,x) \neq (B,y)$ and $(C,z) \neq (D,z)$.

If k colors are enough for coloring the arcs of H, then H is said k-colorable. The (directed) chromatic index of H, denoted by $q(H)$, is the least number k such that H is k-colorable.

[1] E-mail: `vietri@dmmm.uniroma1.it`, `http://www.dmmm.uniroma1.it/~vietri`.

A k-coloring of H can be interpreted as a partition of $E(H)$ into k classes, each class consisting of one or more (disjoint) directed *paths* which yield no fragmentation of flow (both in input and in output) at any node.

Proposition 1 *An optimal arc-coloring of a digraph can be performed in polynomial time with respect to its size.*

Hint of the proof. A digraph D is equivalent to a bipartite graph G_D whose vertices are partitioned into two copies of the nodes of D. Now a classical result proved by Gabow [7] guarantees the existence of a polynomial algorithm for optimally coloring the edges of a bipartite graph.

Proposition 2 *There exists a polynomial algorithm which decides the 2-colorability of a given directed hypergraph of degree 2.*

Hint of the proof. A polynomial reduction can be easily performed from this problem to the 2-coloring problem for the vertices of a suitable non-directed graph.

Definition 3 **(I)** *A directed hypergraph H is an interval (directed) hypergraph if there exists a linear ordering $\leq$ of its nodes, such that every tail of H is a closed interval with respect to $\leq$.* **(II)** *H is non-overlapping if there exists no pair of arcs of H sharing the same head and some node of the tails.*

Theorem 1 *Let H be a non-overlapping, interval hypergraph of degree 2. Then, there exists a polynomial algorithm which evaluates $q(H)$.*

Proof. Using the inequality $q(H) \leq 2\Delta(H) - 1$, which is satisfied by any non-overlapping, interval hypergraph [11], it turns out that evaluating $q(H)$ is equivalent to deciding whether H is 2-colorable or not, and Proposition 2 ensures that such question can be answered in polynomial time.

From the viewpoint of arc-coloring, non-overlapping hypergraphs play a basic role. Indeed, the following holds [11].

Theorem 2 *Every overlapping hypergraph H can be transformed in polynomial time into a non-overlapping hypergraph H' such that $q(H) = q(H')$. (...)*

Hint of the proof. See Figure 0.2.

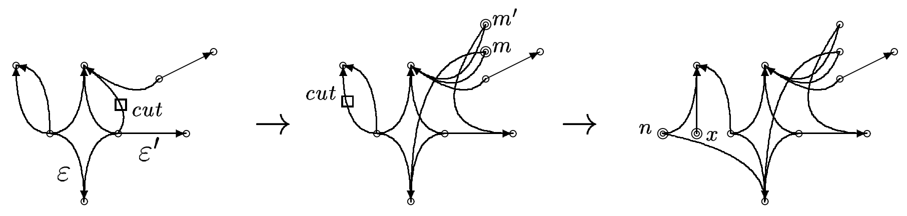
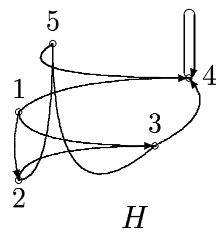


Figure 0.2. From an overlapping hypergraph to a non-overlapping one

Furthermore, non-overlapping hypergraphs can be easily represented by certain labelled adjacency matrices, namely *walls* [11, 12, 13, 14]. Such rephrasing may speed up proofs. In Figure 0.3 we show

how to associate a non-overlapping hypergraph H with the corresponding wall P_H. In particular, all the equally labelled squares of P_H are related to the same arc of H. Notice that the nodes of H need to be indexed, and that the representation depends on the given indexing. Also notice that if H is a digraph, P_H reduces to the transposed adjacency matrix of H.



$$E(H) = \{(\{1\},2), (\{1,2\},3), (\{1,5\},4), (\{3\},4), (\{4\},4), (\{2,3\},5)\}$$

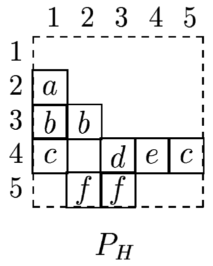


Figure 0.3. An indexed non-overlapping hypergraph and the related wall

In most cases the arc-coloring of directed hypergraphs provides NP-complete problems.

Theorem 3 *For all integers k, d such that $k \geq d \geq 2$ and $k \geq 3$, the k-colorability of the arcs of a given non-overlapping hypergraph having degree d is an NP-complete problem.*

Hint of the proof. In [9] a polynomial reduction is provided from the $3 - SAT$ problem to the edge-coloring problem of cubic graphs. Using this remarkable result, a polynomial reduction from the latter problem to the arc-coloring problem in the claim is then performed, as follows. In the first step we reduce the edge-coloring problem to the 3-colorability of non-overlapping hypergraphs of degree 2. Subsequently, we reduce the latter problem to the k-colorability of non-overlapping hypergraphs of degree 2 with $k \geq 4$, by recursion, as sketched in Figure 0.4 (we use the representation in terms of walls).

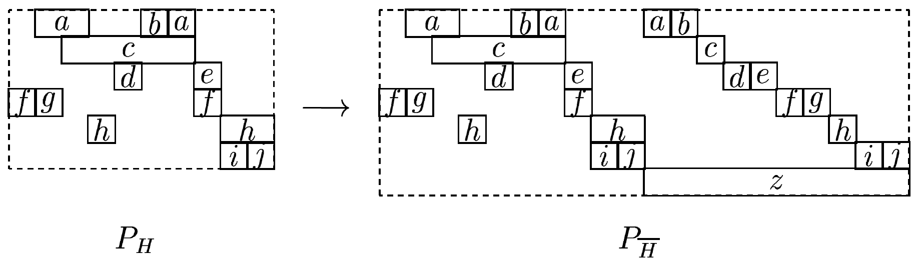


Figure 0.4. $\Delta(H) = \Delta(\overline{H}) = 2$, $q(H) \leq k \Leftrightarrow q(\overline{H}) \leq k+1$

Finally, a third reduction is performed from the above problem to the general problem.

It is clear that NP-completeness still holds if the non-overlap hypothesis is dropped.

References

[1] G. Ausiello, A. D'Atri and D. Saccà, *Minimal Representation of Directed Hypergraphs*, SIAM J. Computing **15** (1986), pp. 418-431.

[2] G. Ausiello, P.G. Franciosa and D. Frigioni, *Directed Hypergraphs: Problems, Algorithmic Results, and a Novel Decremental Approach*, Lecture Notes in Theor. Comp. Sci. **2202** (2001), pp. 312-327.

[3] C. Berge, *Hypergraphs*, North Holland, Amsterdam, 1989.

[4] B. Bollobás, *Modern graph theory*, GTM Vol. 184, Springer, New York, 1998.

[5] D.P. Bovet and P. Crescenzi, *Introduction to the Theory of Complexity*, Prentice Hall, UK, 1994.

[6] M. Espona and O. Serra, *Cayley Digraphs based on the De Bruijn networks*, SIAM J. Discr. Math. **11** (1998), pp. 305-317.

[7] H. Gabow, *Using Euler partitions to edge color bipartite multigraphs*, Internat. J. Comput. Inform. Sci. **5** (1976), pp. 345-355.

[8] G. Gallo, G. Longo, S. Nguyen and S. Pallottino, *Directed Hypergraphs and Applications*, Discr. Appl. Math. **42** (1993), pp. 177-201.

[9] I. Holyer, *The NP-completeness of edge-coloring*, SIAM J. Computing **10** (1981), pp. 718-720.

[10] O. Serra and M.A. Fiol, *Arc-colored line digraphs and their groups*, in Graph Theory, Combinatorics, Algorithms, and Applications, Proc. Appl. Math. **54** (1991), Y. Alavi et al. eds., SIAM.

[11] A. Vietri, *Arc-coloring of directed hypergraphs and chromatic number of walls*, submitted to Discr. Appl. Math..

[12] A. Vietri, *Chromatic number of regular walls*, to appear in Ars Comb..

[13] A. Vietri, *Coherent walls of maximal class*, submitted to SIAM J. Discr. Math..

[14] A. Vietri, *The complexity of arc-colorings for directed hypergraphs*, submitted to Discr. Appl. Math..

Fast and Simple Algorithms for Weighted Perfect Matching

Mirjam Wattenhofer [1], Roger Wattenhofer [2]

Department of Computer Science, ETH Zurich, Switzerland

Abstract

We present two fast and simple combinatorial approximation algorithms for constructing a minimum-weighted perfect matching on complete graphs whose cost functions satisfy the triangle inequality. The first algorithm runs in $O(n^2 \log n)$ time and is at most a factor $\log n$ worse than an optimal solution. In the second algorithm, the average time until a node is matched is $O(n^2)$ and the approximation ratio is $\log^2 n$.

Key words: weighted matching, approximation algorithms

1 Introduction and Motivation

Let $G = K(V)$ be a complete graph of an even number of $|V| = n$ nodes and a non-negative cost function $w : E \to \mathbb{R}_+$ satisfying the triangle inequality. A *perfect matching* of G is a subset $M \subset E$ such that no two edges in M are adjacent and each node is incident to one edge in M. The *weight* $w(M)$ of a matching M is the sum of the weights of its edges. The problem is to find a perfect matching of minimum weight *(MM)*.

This problem can be solved in polynomial time by the algorithm of Edmonds [Edm65]. Though its running time was improved from $O(n^4)$ to $O(n(m+n \log n))$ by [Gab90] it is still too time-consuming for many applications. Hence, much effort was done to find good approximation algorithms for MM which are faster than the exact algorithm (e.g. [GW92], [VA99]). Besides its running time, Edmonds' algorithm has another disadvantage: By making use of the linear programming dual of the MM problem viewed as a linear program, the algorithm and its analysis are difficult to understand, implement, or teach. Unfortunately, most of the other exact or approximation algorithms for MM are based on Edmonds' algorithm or a linear programming approach. This almost inevitably leads to a less intuitive understanding of the algorithms and/or their analysis.

[1] E-mail: `mirjam.wattenhofer@inf.ethz.ch`.
[2] E-mail: `wattenhofer@inf.ethz.ch`.

In this paper we will in a first step introduce a simple, purely combinatorial algorithm for MM, which yields a $2\log n$ approximation in $O(n^2 \log n)$ time. In a next step, we are able to reduce the average time until a node is matched to $O(n^2)$ by losing a factor of $\log n$ in the approximation. Given that the only other simple MM algorithm we are aware of, the greedy algorithm, has an approximation ratio of $\Omega(n^{0.58})$ [RT81] and that the fastest approximation algorithms need time at least $O(n^2\sqrt{n\alpha(m,n)\log n})$[GT91], respectively $O(n^2 \log n)$ [GW92], also on average, we believe that our results are of two-fold interest. On the one hand the algorithms and their analysis are intuitive, and could be used as a stepping stone to explain more intricate algorithms. On the other hand, the approximation ratio and running time of our algorithms slightly improve the currently best known results for polylogarithmic approximations. Altogether, due to their simplicity and efficiency, we hope that our algorithms or derivations thereof will be applicable in the distributed, online, and/or mobile computing setting.

2 The Algorithms

2.1 Description

The main algorithm, Algorithm 1 (Perfect Matching), takes as input an undirected graph $K(V)$ with edge costs $w_e \geq 0$ satisfying the triangle inequality and outputs a perfect matching M. The basic structure of the algorithm involves maintaining a forest F of edges which is initially empty. In each iteration φ of the algorithm's while-loop a set of edges is selected connecting distinct connected components of F. Define an *active component* to be any connected component C of F with an odd number of nodes, else call the component *inactive.* The while-loop terminates when all connected components C of F are inactive. The approximation properties of the algorithm will follow from the way we choose the edges in each iteration φ and which algorithm we call as a subroutine to compute the matching on the forest F. First to the edge-selection step: In each iteration the algorithm connects every active component to the *nearest* other active component, where nearest is meant in the sense of minimizing the weight of the edges to be added (see Figure A.1). Now to the matching-subroutines: Algorithm 2 (Idle Match) computes the matching on the forest F after the termination of the while-loop of Algorithm 1, whereas Algorithm 3 (Instant Match) matches the nodes in an ongoing fashion in each iteration of the while-loop. Thus, decreasing the average time until a node is matched significantly but naturally losing some accuracy.

2.2 Analysis

In this section we prove the approximation properties of the Perfect Matching Algorithm (Algorithm 1) in connection with the Idle Match Algorithm (Algorithm 2), respectively the Instant Match Algorithm (Algorithm 3). By using the Idle Matching Algorithm the matching we compute comes within a factor of $\log n$ of optimal. Whereas, for Algorithm 3 we prove an approximation ratio of $\log^2 n$. In the next section we show that the running time of the algorithms is $n^2 \log n$, but that for the Instant Match Algorithm the average time until a node is matched is $O(n^2)$.

Lemma 1 *The while-loop of the Perfect Matching Algorithm is executed at most* $\log n$ *times.*

Algorithm 1 Perfect Matching

```
Input: an undirected graph $G = K(V)$ with edge costs $w_e \geq 0$
Output: a matching $M$
1: (* depending on preferences either call line 11 or line 14 but not both to compute
   matching *)
2: $F \leftarrow \emptyset$, $M \leftarrow \emptyset$, $\varphi = 0$,
3: $\mathcal{C}_\varphi \leftarrow \{\{v\} : v \in V\}$
4: while ($\exists$ active component $C \in \mathcal{C}_\varphi$) do
5:    $E_{\mathcal{C}_\varphi} \leftarrow \emptyset$
6:    for each active component $C \in \mathcal{C}_\varphi$ do
7:       find a path $P$ from $C$ to an active component $C'$, $C \neq C'$, that minimizes $w(E_C)$,
         where $E_C = P - F$
8:       $E_{\mathcal{C}_\varphi} \leftarrow E_{\mathcal{C}_\varphi} \dot\cup E_C$
9:    end for
10:   $F \leftarrow F \cup E_{\mathcal{C}_\varphi}$
11:   $\boldsymbol{M} \leftarrow$ Algorithm 3 with Input $(\boldsymbol{F}, \boldsymbol{M}, \boldsymbol{\varphi})$
12:   $\varphi \leftarrow \varphi + 1$
13:   update $\mathcal{C}_\varphi$
14: end while;
15: $\boldsymbol{M} \leftarrow$ Algorithm 2 with Input $\boldsymbol{F}$
```

Algorithm 2 Idle Match

```
Input: a forest $F$
Output: a perfect matching $M$
1: duplicate every edge of each component of $F$ and shortcut to obtain a collection of cycles
   (see Figure A.2) (* the cycles have even length since all components are inactive *)
2: keep the best matching $M$ out of the two matchings defined by every cycle
```

Algorithm 3 Instant Match

```
Input: a forest $F$, a partial matching $M$, $\varphi$
Output: a partial matching $M$
1: if ($\varphi = 0$) then
2:    duplicate every edge of each component of $F$ and shortcut to obtain a collection of
      (maybe odd-length) cycles
3:    for every odd cycle fix an arbitrary node to remain unmatched
4:    keep the best matching out of the two (partial) matchings defined by every cycle (and
      fixed node)
5: else
6:    match free nodes in every component, s.t. the paths between matched nodes are
      disjoint (* See Lemma 3 for how to construct such paths. *)
7: end if
8: update $M$
```

Proof. At least three active components must be connected to each other such that the new component is active again. Therefore, in each iteration the number of active components decreases by a factor of at least three and the logarithmic number of executions follows. □

The incidence vector of the forest F produced by the Perfect Matching Algorithm is a feasible solution for following integer linear programm *(IP)*:

$$\begin{array}{llll} (IP) & \text{Min} & \sum_{e\in E} w_e x_e & \\ & \text{subject to:} & x(\delta(S)) \geq |S| \, mod \, 2 & \emptyset \neq S \subset V \\ & & x_e \in \{0,1\} & e \in E, \end{array}$$

where $\delta(S)$ denotes the set of edges having exactly one endpoint in S and $x(F) = \sum_{e\in F} x_e$. The optimal solution F^* to *(IP)* is obviously of less weight than is the optimal solution M^* of the minimum-weighted perfect matching problem.

Lemma 2 *The weight of the set of edges added in each execution of the while-loop of the Perfect Matching Algorithm is at most the weight of the optimal forest F^*: $w(E_{\mathcal{C}_\varphi}) \leq w(F^*)$.*

Proof. We will argument about an arbitrary iteration φ. Construct a graph H by considering the inactive and active components of this iteration as nodes in H. We conceptually divide the edges of H into *red* and *blue* edges. The edges $e \in F^* \cap \delta(C)$ for all $C \in \mathcal{C}_\varphi$ are the *red* edges of H. The edges $e \in E_{\mathcal{C}_\varphi}$ added in this iteration to F are the *blue* edges[3]. Keeping in mind that the incidence vector of edges of F^* must be a feasible solution for (IP) we know that for each node v in H associated with an active component (a so called *active node*) there is a path P_v^r, consisting of red edges only, which connects this node to another active node in H. By the way we chose $E_{\mathcal{C}_\varphi}$ we know that there is also a blue path P_v^b connecting v to another active node in H which is at most as heavy as the corresponding red path: $w(P_v^b) \leq w(P_v^r)$. Consequently, $\sum_{v\in V_a} w(P_v^b) \leq \sum_{v\in V_a} w(P_v^r)$, where V_a is the set of nodes in H associated with active components. Since, $\sum_{v\in V_a} w(P_v^r) \leq 2w(F^*)$ and $2w(E_{\mathcal{C}_\varphi}) = \sum_{v\in V_a} w(P_v^b)$ the lemma follows. (The factors of two arise because each path is counted twice, once for each endpoint.) □

Corollary 0.1 *The weight of the forest F at the termination of the while-loop of the Perfect Matching Algorithm is at most $\log n \cdot w(F^*)$.*

Proof. The Corollary follows immediately from Lemma 1 and Lemma 2. □

Theorem 1 *The matching computed by the Idle Matching Algorithm is a $\log n$-approximation of the minimum-weighted perfect matching M^*.*

Proof. We compute an Euler tour with shortcuts on each component and choose the better out of the two possible matchings for each tour. Since the weight of an Euler tour is at most twice the weight of the component, the weight of the better matching is at most the weight of the component. All together the matching has weight at most $w(F)$, which is by Corollary 0.1 at most $\log n \cdot w(F^*)$. Since $w(F^*) \leq w(M^*)$ we can deduce the theorem. □

[3] Note that some edges may be red as well as blue.

Before we analyze the Instant Match Algorithm (Algorithm 3) we need following helper lemma.

Lemma 3 *Given a tree $T = (V, E)$ and a set of marked nodes $S = \{v_1, v_2, \ldots\} \subseteq V$. Let $S_p = \{(v_i, v_j), \ldots\} \subset \binom{S}{2}$ be a disjoint pairing of the nodes in S and for each pair (v_i, v_j), p_{ij} the path connecting v_i to v_j. Then it is always possible to construct S_p in such a way that the paths p_{ij} are mutually edge-disjoint.*

Proof. We prove the lemma by giving a construction for S_p. Beginning from the leaves, pair v_i with v_j if they have a common ancestor v_a and there is no un-paired node v_k which has a closer common ancestor v_b with either v_i or v_j, where closer is meant in the sense of the length of the path between the nodes. This construction leads to all desired properties of the set S_p. □

Lemma 4 *The weight of the edges added to the matching M at the end of the Instant Match Algorithm is at most the weight of the input forest F.*

Proof. If $\varphi = 0$ we basically do the same as in the Idle Match Algorithm, except for some cycles being of odd length, which does not have any implications on the weight of the matching. Hence, the weight of the edges added to the matching is at most the weight of the components of F, which is $w(F)$. For $\varphi > 0$ we know by Lemma 3 that we can match the free nodes (except for one in each active component) in a component of F such that the paths between matched nodes are disjoint. Furthermore, by the triangle inequality we know that connecting the nodes directly is at most as expensive as connecting them via other nodes. Consequently, for each component of F the weight of the edges added to the matching is at most the weight of the component itself and the lemma follows. □

Theorem 2 *The matching computed by the Instant Match Algorithm is a $\log^2 n$-approximation of the minimum-weighted perfect matching M^*.*

Proof. By Lemma 2 the weight of the forest in iteration φ is at most

$$w(F) = w(\mathcal{C}_\varphi) = w(\mathcal{C}_{\varphi-1}) + w(E_{\mathcal{C}_{\varphi-1}}) \leq w(\mathcal{C}_{\varphi-1}) + w(F^*) \leq \varphi \cdot w(F^*).$$

By Lemma 4 we have:

$$\begin{aligned} w(M) &\leq \sum_{\varphi=1}^{\log n} w(\mathcal{C}_\varphi) \leq \sum_{\varphi=1}^{\log n} \varphi \cdot w(F^*) \\ &= \frac{1}{2}(\log^2 n + \log n) \cdot w(F^*) \leq \log^2 n \cdot w(M^*) \end{aligned} \quad .$$

□

2.3 Implementing the Algorithms

We now turn to the problem of implementing the algorithms efficiently. Both, the Idle Match Algorithm and the Instant Match Algorithm, have a running time of $O(n)$. Thus, the critical step of the implementation is the while-loop of the Perfect Matching Algorithm. More specific, in each iteration the crucial part is to find the paths P and to merge the components in the update of $\mathcal{C}_\varphi$. If we maintain the components C as a union-find structure of nodes, merging will take at most $O(n\alpha(n,n))$ time, α being the inverse Ackermann function [Tar75]. By using a generalized Voronoi diagram the time to find the shortest path for all active components in an iteration is $O(n^2)$ [SPR80]. Since the while-loop is executed at most $\log n$ times, after $O(n^2 \log n)$ time steps, the Perfect Matching Algorithm terminates, independent of whether we use the Idle Match or the Instant Match Algorithm as a subroutine.

In the following we will prove that the average time until a node is matched decreases by a factor of magnitude $\log n$ if we use the Instant Match Algorithm.

Theorem 3 *After on average at most two iterations of the while-loop of the Perfect Matching Algorithm and using the Instant Match Algorithm as a subroutine a node is matched.*

Proof. After the first iteration at least $\frac{2}{3}$ of all nodes are matched, since at most one node remains unmatched in each active component. By the same reasoning, after iteration φ at least $(1-(\frac{1}{3})^\varphi)\cdot n$ nodes are matched. For the average matching time we get:
$E[\#\text{ iterations until a node is matched}] = \sum_{\varphi=1}^{\log n}(1-(\frac{1}{3})^\varphi)(1-(\frac{1}{3})^{\varphi-1})\cdot\varphi = \sum_{\varphi=1}^{\log n} 2(\frac{1}{3})^\varphi \cdot i \leq \frac{3}{2}$.
□

References

[Edm65] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[Gab90] H.N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. *in Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 434–443, 1990.

[GT91] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph matching problems. *Journal of the ACM*, 38(4):815–853, 1991.

[GW92] M.X. Goemans and D.P. Williamson. A general approximation technique for constrained forest problems. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1992.

[RT81] E.M. Reingold and R.E. Tarjan. On a greedy heuristic for complete matching. *SIAM Journal on Computing*, 10:676–681, 1981.

[SPR80] K.J. Suppowit, D.A. Plaisted, and E.M. Reingold. Heuristics for weighted perfect matching. *in Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 398–419, 1980.

[Tar75] R.E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22:215–225, 1975.

[VA99] K.R. Varadarajan and P.K. Agarwal. Approximation algorithms for bipartite and non-bipartite matching in the plane. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1999.

A APPENDIX

A.1 Visualization of Matching Algorithms

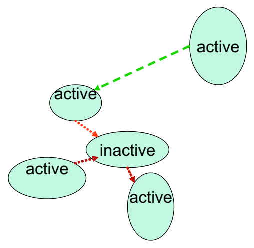


Figure A.1. Connecting Active Components

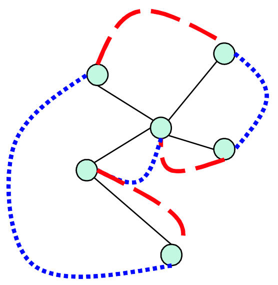


Figure A.2. Duplicating and Shortcutting

Heavy cycles in k-connected weighted graphs

Shenggui Zhang[1], Bing Chen

Department of Applied Mathematics, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, P.R. China

Rongzu Yu

School of Arts and Science, Air Force Engineering University, Xi'an, Shaanxi 710043, P.R. China

Abstract

A weighted graph is one in which every edge e is assigned a nonnegative number $w(e)$, called the weight of e. The weight of a cycle is defined as the sum of the weights of its edges. The weighted degree of a vertex is the sum of the weights of the edges incident with it. In this paper, we prove that: Let G be a k-connected weighted graph where $k \geq 2$. Then G contains either a Hamilton cycle or a cycle of weight at least $2m/(k+1)$, if G satisfies the following conditions: (1) The weighted degree sum of any $k+1$ independent vertices is at least m; (2) In each induced claw, each induced modified claw and each induced P_4 of G, all edges have the same weight. This generalizes an early result of Enomoto *et al.* on the existence of heavy cycles in k-connected weighted graphs.

Key words: heavy cycle, weighted degree (sum), induced claw (modified claw, P_4)
AMS Subject Classification (1991): 05C45 05C38 05C35

A Terminology and notation

We use Bondy and Murty [5] for terminology and notation not defined here and consider finite simple graphs only.

Let $G = (V, E)$ be a simple graph. G is called a *weighted graph* if each edge e is assigned a nonnegative number $w(e)$, called the *weight* of e. For a subgraph H of G, $V(H)$ and $E(H)$ denote the sets of vertices and edges of H, respectively. The *weight* of H is defined by $w(H) = \sum_{e \in E(H)} w(e)$. For a vertex $v \in V$, $N_H(v)$ denotes the set, and $d_H(v)$ the number, of vertices in H that are adjacent to v. We define the *weighted degree* of v in H by $d^w_H(v) = \sum_{h \in N_H(v)} w(vh)$. When no confusion occurs, we will denote $N_G(v)$, $d_G(v)$ and $d^w_G(v)$ by $N(v)$, $d(v)$ and $d^w(v)$, respectively.

[1] E-mail: `shengguizhang@hotmail.com`.

An unweighted graph can be regarded as a weighted graph in which each edge e is assigned weight $w(e) = 1$. Thus, in an unweighted graph, $d^w(v) = d(v)$ for every vertex v, and the weight of a subgraph is simply the number of its edges.

The number of vertices in a maximum independent set of G is denoted by $\alpha(G)$. If G is noncomplete, then for a positive integer $k \leq \alpha(G)$ we denote by $\sigma_k(G)$ the minimum value of the degree sum of any k independent vertices, and by $\sigma_k^w(G)$ the minimum value of the weighted degree sum of any k independent vertices. If G is complete, then both $\sigma_k(G)$ and $\sigma_k^w(G)$ are defined as ∞.

We call the graph $K_{1,3}$ a *claw*, and the graph obtained by joining a pendant edge to some vertex of a triangle a *modified claw*.

B Results

There have been many results on the existence of long paths and cycles in unweighted graphs. In [3] and [4], Bondy and Fan generalized several classical theorems of Dirac and of Erdös and Gallai on paths and cycles to weighted graphs. A weighted generalization of Ore's theorem was obtained by Bondy *et al.* [2]. In [11], it was shown that if one wants to generalize Fan's theorem on the existence of long cycles to weighted graphs some extra conditions cannot be avoided. By adding two extra conditions, the author gave a weighted generalization of Fan's theorem.

Among the many results on cycles in unweighted graphs, the following generalization of Ore's theorem is well-known.

Theorem 1 (Fournier and Fraisse [8]) *Let G be a k-connected graph where $2 \leq k < \alpha(G)$, such that $\sigma_{k+1}(G) \geq m$. Then G contains either a Hamilton cycle or a cycle of length at least $2m/(k+1)$.*

A natural question is whether Theorem A also admits an analogous generalization for weighted graphs. This leads to the following problem.

Problem 1 *Let G be a k-connected weighted graph where $2 \leq k < \alpha(G)$, such that $\sigma_{k+1}^w(G) \geq m$. Is it true that G contains either a Hamilton cycle or a cycle of weight at least $2m/(k+1)$?*

It seems very difficult to settle this problem, even for the case $k = 2$. Motivated by the result in [11], Zhang *et al.* [10] proved that the answer to Problem 1 in the case $k = 2$ is positive with two extra conditions.

Theorem 2 (Zhang *et al.* [10]) *Let G be a 2-connected weighted graph which satisfies the following conditions: (1) $\sigma_3^w(G) \geq m$; (2) $w(xz) = w(yz)$ for every vertex $z \in N(x) \cap N(y)$ with $d(x, y) = 2$; (3) In every triangle T of G, either all edges of T have different weights or all edges of T have the same weight. Then G contains either a Hamilton cycle or a cycle of weight at least $2m/3$.*

In [7], after giving a characterization of the graphs satisfying Conditions (2) and (3) of Theorem 1, Enomoto *et al.* proved that the answer to Problem 1 is positive for any $k \geq 2$ with these two extra conditions.

Theorem 3 (Enomoto *et al.* [7]) *Let G be a k-connected weighted graph where $k \geq 2$. Suppose that G satisfies the following conditions:* (1) *$\sigma_{k+1}^w(G) \geq m$;* (2) *$w(xz) = w(yz)$ for every vertex $z \in N(x) \cap N(y)$ with $d(x,y) = 2$;* (3) *In every triangle T of G, either all edges of T have different weights or all edges of T have the same weight. Then G contains either a Hamilton cycle or a cycle of weight at least $2m/(k+1)$.*

On the other hand, Fujisawa [9] gave a so-called claw condition for the existence of heavy cycles in weighted graphs, which is a weighted generalization of a result of Bedrossian *et al.* [1].

Theorem 4 (Fujisawa [9]) *Let G be a 2-connected weighted graph which satisfies the following conditions:* (1) *For each induced claw and each induced modified claw of G, all its nonadjacent pair of vertices x and y satisfy $max\{d^w(x), d^w(y)\} \geq s/2$;* (2) *For each induced claw and each induced modified claw of G, all of its edges have the same weight. Then G contains either a Hamilton cycle or a cycle of weight at least s.*

A result similar to this theorem was obtained by Chen and Zhang [6]. It also generalizes Theorem 1.

Theorem 5 (Chen and Zhang [6]) *Let G be a 2-connected weighted graph which satisfies the following conditions:* (1) *$\sigma_3^w(G) \geq m$;* (2) *For each induced claw and each induced modified claw of G, all of its edges have the same weight. Then G contains either a Hamilton cycle or a cycle of weight at least $2m/3$.*

Thus, we have the following question: Can Conditions (2) and (3) in Theorem 2 be weakened by Condition (2) of Theorem 3?

Problem 2 *Let G be a k-connected weighted graph where $2 \leq k < \alpha(G)$. Suppose that G satisfies the following conditions:* (1) *$\sigma_{k+1}^w(G) \geq m$;* (2) *For each induced claw and each induced modified claw of G, all of its edges have the same weight. Is it true that G contains either a Hamilton cycle or a cycle of weight at least $2m/(k+1)$?*

In this paper, we give a partial answer to this problem. Our result is a generalization of Theorem 2.

Theorem 6 *Let G be a k-connected weighted graph where $k \geq 2$. Suppose that G satisfies the following conditions:* (1) *$\sigma_{k+1}^w(G) \geq m$;* (2) *In each induced claw, each induced modified claw and each induced P_4 of G, all of its edges have the same weight. Then G contains either a Hamilton cycle or a cycle of weight at least $2m/(k+1)$.*

References

[1] Bedrossian, P., Chen, G., Schelp, R.H.: A generalization of Fan's condition for hamiltonicity, pancyclicity, and hamiltonian connectedness. Discrete Math. **115**, 39-50 (1993)

[2] Bondy, J.A., Broersma, H.J., Heuvel, J. van den, Veldman, H.J.: Heavy cycles in weighted graphs. Discuss. Math. Graph Theory. **22**, 7-15 (2002)

[3] Bondy, J.A., Fan, G.: Optimal paths and cycles in weighted graphs. Ann. Discrete Math. **41**, 53-69 (1989)

[4] Bondy, J.A., Fan, G.: Cycles in weighted graphs. Combinatorica. **11**, 191-205 (1991)

[5] Bondy, J.A., Murty, U.S.R.: Graph Theory with Applications. New York, North-Holland 1979

[6] Chen, B, Zhang, S.: A new σ_3 type condition for heavy cycles in weighted graphs. (submitted)

[7] Enomoto, H, Fujisawa, J., Ota, K.: A σ_k type condition for heavy cycles in weighted graphs. Preprint (2003)

[8] Fournier, I., Fraisse, P.: On a conjecture of Bondy. J. Combin. Theory Ser. B. **39**, 17-26 (1985)

[9] Fujisawa, J.: Claw conditions for heavy cycles in weighted graphs. Preprint (2003)

[10] Zhang, S., Broersma, H.J., Li, X.: A σ_3 type condition for heavy cycles in weighted graphs. Discuss. Math. Graph Theory. **21**, 159-166 (2001)

[11] Zhang, S., Broersma, H.J., Li, X., Wang, L.: A Fan type condition for heavy cycles in weighted graphs. Graphs Combin. **18**, 193-200 (2002)

Behzad-Vizing conjecture and Cartesian product graphs

Blaž Zmazek[1] Janez Žerovnik[2]

FME, University of Maribor, Smetanova 17, SI-2000 Maribor, Slovenia

IMFM, Jadranska 19, SI-1111 Ljubljana, Slovenia

Abstract

We prove the following theorem: if the Behzad-Vizing conjecture is true for graphs G and H, then is it true for the Cartesian product $G\Box H$.

Key words: Cartesian graph product, chromatic number, total chromatic number, Vizing conjecture.

A Introduction

Graph products were first defined by Sabidussi [1] and Vizing [2]. A lot of work was done on various topics related to graph products, but on the other hand there are still many questions open. For a very recent survey, see [3].

Here we consider the total chromatic number on the Cartesian product graphs. It is well known and easy to see that the chromatic number of the Cartesian product is the maximum of the chromatic numbers of the factors. For the chromatic index (edge–chromatic number), the Cartesian product is of class I (type I) if at least one of the factors is of class I [4]. In this paper we give a corresponding result for the total chromatic number. Total chromatic numbers of Cartesian products of a path and a star, a cycle and a star, a path and a cycle and (in certain cases) of two cycles are given in [5] (see also [6]).

The *Cartesian product* $G\Box H$ of two graphs G and H is the graph with vertex set $V(G) \times V(H)$, in which the vertex (a, b) is adjacent to the vertex (c, d) whenever $a = c$ and b is adjacent to d, or $b = d$ and a is adjacent to c. The G- and H- *layers* are the induced subgraphs in $G\Box H$ on vertex sets $G_u = \{(x, u) \mid x \in V(G)\}$ and $H_v = \{(v, x) \mid x \in V(H)\}$, respectively. An element of a graph G is either a vertex or an edge of G. In a *proper total coloring*, any two elements that are either adjacent or incident are assigned different colors. The minimum number of colors needed for a proper total coloring is the *total chromatic number* of G, denoted by $\chi''(G)$. The maximum vertex degree in G is denoted by $\Delta(G)$.

[1] E-mail: `blaz.zmazek@uni-mb.si`.
[2] E-mail: `janez.zerovnik@uni-lj.si`.

Total coloring was introduced by Behzad [7, 8] and Vizing [9], both of whom conjectured that for any graph G, the following inequality holds:

$$\Delta(G)+1 \leq \chi''(G) \leq \Delta(G)+2.$$

The lower bound is clearly best possible. Graphs that need only $\Delta(G)+1$ colors are called graphs of *type I.* Examples include the cycles C_{3k}, paths and complete graphs with odd number of vertices K_{2k+1}. Graphs that need at least $\Delta(G)+2$ colors are called graphs of *type II.* Examples include the cycles C_{3k+1} and C_{3k+2}, complete graphs with even number of vertices K_{2k}, and the Cartesian product $K_2 \Box C_5$. For a survey on total colorings see [10].

Our key result is the following

Theorem 1 *Let G and H be arbitrary graphs such that $\Delta(G) \leq \Delta(H)$. Then*

$$\chi''(G \Box H) \leq \Delta(G)+\chi''(H).$$

Clearly, if for H the Behzad-Vizing conjecture is true, then $\chi''(G \Box H) \leq \Delta(G)+\chi''(H) \leq \Delta(G)+\Delta(H)+2$, hence the conjecture is true for $G \Box H$.

Corollary 1.1 *If the Behzad-Vizing conjecture is true for graphs G and H, then it is true for the Cartesian product $G \Box H$.*

Furthermore, from above result it easily follows:

Corollary 1.2 *If the factor with largest vertex degree is of type I, then the product is also of type I.*

Corollary 1.3 $\chi''(G \Box H) \leq \min\{\Delta(G)+\chi''(H), \Delta(H)+\chi''(G)\}$ *provided* $\Delta(G)=\Delta(H)$. *In particular, the product of two cycles $C_{3k} \Box C_n$ is of type I.*

Corollary 1.4 *Any graph product $P_m \Box S_n$ or $P_n \Box C_m$, $n>2$ is of type I, where S_m is a star (graph with one universal vertex and m vertices of degree 1).*

However, there are examples in which one factor is of type I and the other factor is of type II, and the Cartesian product is in some cases of type I and in other cases of type II. Furthermore, there are examples where the product of two type II factors is of type I. See examples below.

Proof of Theorem 1. Let G and H be arbitrary graphs such that $\Delta(G) \leq \Delta(H)$. Then we will prove that

$$\chi''(G \Box H) \leq \Delta(G)+\chi''(H).$$

We prove the assertion by constructing the total coloring.

First color edges of G using colors -1,-2,...,-($\Delta(G)$ +1). Color edges of all layers G_u in the same way. (Hence vertices, and edges of layers H_v, have not yet been colored.) Clearly, there is at least one color at each vertex $v \in V(G)$ which is not used for any edge incident to v. Select a color not used on any edge incident to v and denote it $free(v)$.

Let I_0, I_1,...,$I_{\chi(G)-1}$ be a decomposition of $V(G)$ into independent sets. Recall that $\chi(G) \leq \Delta(G) + 1 \leq \Delta(H) + 1 \leq \chi''(H)$.

Let T_0, T_1,...,$T_{\chi''(H)-1}$ be a decomposition of elements of H into independent sets of some proper total coloring. (Hence $T_i \subseteq V(H) \cup E(H)$.) Using this decomposition we will now temporarily assign colors to elements of $V(G) \times (V(H) \cup E(H))$, in order to complete the total coloring of $G \Box H$ by coloring the vertices, and the edges of layers H_v, as follows: Element of $I_i \times T_j$ receives color $(i + j) \bmod \chi''(H)$. It is straightforward to see that colors 0,1,2,...,$\chi''(H) - 1$ are used and that this is a proper coloring.

Finally, for each vertex $v \in V(G)$, replace color 0 in layer H_v by the color $free(v)$. Observe that this does not introduce any violation of the proper coloring. We omit the details.

Summarizing we see that $\Delta(G) + 1 + \chi''(H)$ colors were used, but after recoloring we have a proper total coloring with $\Delta(G) + \chi''(H)$ colors as claimed. □

Various examples will be given showing the existence of

- type II Cartesian graph product of type II graphs;
- type I Cartesian graph product of type II graphs;
- type II Cartesian graph products of type II graphs;
- Type I Cartesian graph product of type I and type II graphs;
- etc.

References

[1] G.Sabidussi, Graph Multiplication, *Math. Z.* **72** (1960) 446–457.
[2] V.G.Vizing, The Cartesian product of graphs, *Vyc. Sis.* **9** (1963) 30–43.
[3] W.Imrich and S.Klavžar, Product Graphs: Structure and Recognition, *John Wiley & Sons*, New York, 2000 (in print).
[4] E.S.Mahmoodian, On edge-colorability of Cartesian products of graphs, *Canad. Math. Bull.* **24** (1981) 107–108.
[5] M.A.Seoud, A.E.I. Abd el Maqsoud, R.J.Wilson and J.Williams, Total colourings of Cartesian products, *Int. J. Math. Educ. Sci. Technol.* **28** (1997) 481–487.
[6] M.A.Seoud, Total chromatic numbers, *Appl. Math. Lett.* **5** (1992) 37–39.
[7] M.Behzad, Graphs and their chromatic numbers, *Ph.D. Thesis, Michigan State University*, East Lansing, 1965.

[8] M.Behzad, The total chromatic number, *Combinatorial Math. and its Applications (Proc. Conf., Oxford 1969)* pp 1-8, Academic Press, London, 1971.

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

[10] H.P.Yap, Total colourings of graphs, Lecture Notes in Mathematics 1623, *Springer*, Berlin 1996.

Author Index

Printed by:
Centro Stampa, Dipartimento di Elettronica e Informazione
Politecnico di Milano, Milano, Italy
29 April 2004