Welcome to the

12th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2013).

CTW 2013 takes place at the University of Twente, Enschede, Netherlands, from May 21 to May 23, 2013.

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

As it was the case with previous CTWs, we will edit a special edition of *Discrete Applied Mathematics* for CTW 2013. Hereby, we invite all participants to submit full-length papers related to the topics of the workshop.

Program Committee.

- Ulrich Faigle (University of Cologne, Germany)
- Johann L. Hurink (University of Twente, Enschede, Netherlands, co-chair)
- Renato de Leone (Università degli Studi di Camerino, Italy)
- Leo Liberti (École Polytechnique, Paris, France)
- Bodo Manthey (University of Twente, Enschede, Netherlands, co-chair)
- Gaia Nicosia (Università degli studi Roma Tre, Italy)
- Andrea Pacifici (Università degli Studi di Roma "Tor Vergata", Italy)
- Stefan Pickl (Universität der Bundeswehr München, Germany)
- Giovanni Righini (Università degli Studi di Milano, Italy)
- Rainer Schrader (University of Cologne, Germany)
- Rüdiger Schultz (University Duisburg-Essen, Germany)

Organizing Committee.

- Kamiel Cornelissen
- Ruben Hoeksma
- Johann L. Hurink
- Bodo Manthey

We thank Marjo Mulder for her help. We gratefully acknowledge the financial support from the Centre for Telematics and Information Technology (CTIT) of the University of Twente, Paragon Decision Technology, ORTEC, and Stichting Universiteitsfonds Twente.

Johann Hurink *Bodo Manthey*

CTIT ORTEC AIMMS THE MODELING SYSTEM 3 UF stichting universiteitsfonds twente

List of Abstracts

Nair M. M. de Abreu, Maria A. A. de Freitas, Renata R. Del-Vecchio
Simultaneously integral graphs on three associated matrices 7

Stephan Dominique Andres, Winfried Hochstättler
Perfect digraphs and a strong perfect digraph theorem 11

Danilo Artigas, Simone Dantas, Mitre C. Dourado, Jayme L. Szwarcfiter
Geodetic sets and periphery 15

D. Bakarcic, G. Di Piazza, I. Méndez-Díaz, P. Zabala
An IP based heuristic algorithm for the vehicle and crew scheduling pick-up and delivery problem with time windows 19

Karl Bringmann, Benjamin Doerr, Adrian Neumann, Jakub Sliacan
Online checkpointing with improved worst-case guarantees 23

Tobias Brunsch, Kamiel Cornelissen, Bodo Manthey, Heiko Röglin
Smoothed analysis of the successive shortest path algorithm 27

Christoph Buchheim, Laura Klein
The spanning tree problem with one quadratic term 31

Christina Büsing, Fabio D'Andreagiovanni, Annie Raymond
Robust optimization under multiband uncertainty 35

Eglantine Camby, Oliver Schaudt
Connected dominating set in graphs without long paths and cycles 39

Márcia R. Cerioli, Daniel F. D. Posner
Total $L(2,1)$-coloring of graphs 43

Sourav Chakraborty, Akshay Kamath, Rameshwar Pratap
Testing uniformity of stationary distribution 47

Yonah Cherniavsky, Avraham Goldstein, Vadim E. Levit
Balanced Abelian group valued functions on directed graphs: Extended abstract . . 51

Hebert Coelho, Luerbio Faria, Sylvain Gravier, Sulamita Klein
An oriented 8-coloring for acyclic oriented graphs with maximum degree 3 55

Stefano Coniglio
Bound-optimal cutting planes 59

Fernanda Couto, Luerbio Faria, Sulamita Klein, Loana T. Nogueira, Fábio Protti
On specifying boundary conditions for the graph sandwich problem 63

Jean-François Couturier, Mathieu Liedloff
A tight bound on the number of minimal dominating sets in split graph 67

Radu Curticapean, Marvin Künnemann
A quantization framework for smoothed analysis on Euclidean optimization problems 71

Elias Dahlhaus
Linear time and almost linear time cases for minimal elimination orderings 75

S. Dantas, C. M. H. de Figueiredo, G. Mazzuoccolo, M. Preissmann, V. F. dos Santos, D. Sasaki
On total coloring and equitable total coloring of cubic graphs with large girth . . . 79

Ekrem Duman, Ahmet Altun
Routing ATM loading vehicles 85

Michael Etscheid
Performance guarantees for scheduling algorithms under perturbed machine speeds 89

Ulrich Faigle, Alexander Schönhuth
Observation and evolution of finite-dimensional Markov systems 93

Philipp von Falkenhausen, Tobias Harks
Optimal cost sharing for capacitated facility location games 99

Ángel Felipe Ortega, M. Teresa Ortuño Sánchez, Gregorio Tirado Domínguez, Giovanni Righini
Exact and heuristic algorithms for the green vehicle routing problem 103

Mirjam Friesen, Dirk Oliver Theis
Fooling-sets and rank in nonzero characteristic 107

Giulia Galbiati, Stefano Gualandi
Coloring of paths into forests 113

Valentin Garnero, Ignasi Sau, Dimitrios M. Thilikos
A linear kernel for planar red-blue dominating set 117

Ismael González Yero, Amaurys Rondón Aguilar
The double projection method for some domination related parameters in Cartesian product graphs 121

Ruben Hoeksma, Marc Uetz
Two-dimensional optimal mechanism design for a single machine scheduling problem 125

Olivier Hudry
Application of the descent with mutations (DWM) metaheuristic to the computation of a median equivalence relation 129

Sebastiaan J. C. Joosten, Hans Zantema
Relaxation of 3-partition instances 133

K. Karam, D. Sasaki
Semi blowup and blowup snarks and Berge-Fulkerson Conjecture 137

Roland Kaschek, Alexander Krumpholz
An extension of the Collatz function 141

Imran Khaliq, Gulshad Imran
Constructing strategies in subclasses of McNaughton games 145

Philipp Klodt, Anke van Zuylen
Toward a precise integrality gap for triangle-free 2-matchings 151

Monique Laurent, Zhao Sun
Handelman's hierarchy for the maximum stable set problem 155

Maciej Liśkiewicz, Martin R. Schuster
A new upper bound for the traveling salesman problem in cubic graphs 159

Ovidiu Listes
Manufacturing process flexibility with robust optimization using AIMMS 163

Dmitrii Lozovanu, Stefan Pickl
Optimal paths in networks with rated transition time costs 165

Ján Maňuch, Murray Patterson, Roland Wittler, Cedric Chauve, Eric Tannier
Linearization of ancestral multichromosomal genomes 169

Isabel Méndez-Díaz, Federico Pousa, Paula Zabala
A branch-and-cut algorithm for the angular TSP 175

Xavier Molinero, Fabián Riquelme, Maria Serna
Star-shaped mediation in influence games 179

Haiko Müller, Samuel Wilson
Characterising subclasses of perfect graphs with respect to partial orders related to edge contraction 183

Andrea Munaro
The VC-dimension of graphs with respect to k-connected subgraphs 187

S. Pirzada, Muhammad Ali Khan, E. Sampathkumar
Coloring of signed graphs 191

Alain Quilliot, Djamal Rebaine
Approximation results for the linear ordering problem on interval graphs 197

Fabio Roda
Hazmat transportation problem: instance size reduction through centrality erosion . 201

Uéverton dos Santos Souza, Fábio Protti, Maise Dantas da Silva
Parameterized and/or graph solution 205

Thatchaphol Saranurak
Finding the colors of the secret in Mastermind 209

Eckhard Steffen
1-factors and circuits of cubic graphs 213

Lara Turner, Matthias Ehrgott, Horst W. Hamacher
On the generality of the greedy algorithm for solving matroid problems 217

Sven de Vries
Graph products for faster separation of 1-wheel inequalities 219

Ayumi Igarashi, Yoshitsugu Yamamoto
Computational complexity of the average covering tree value 223

Simultaneously integral graphs on three associated matrices

Nair M. M. de Abreu[1], Maria A. A. de Freitas[1], and Renata R. Del-Vecchio[2]

[1]Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil
[2]Universidade Federal Fluminense, Niteroi, Brazil

In this article, we construct graphs that are simultaneously integral, Laplacian integral and signless Laplacian integral. The graphs with this property, known in the literature, are regular or bipartite, unless by a few exceptions. We obtain infinite families of such graphs, that are neither regular nor bipartite, from join of regular graphs.

1 Introduction

Let $G = (V, E)$ be a simple graph on n vertices and $D(G) = diag(d_1, \ldots, d_n)$ be the diagonal matrix of its vertex degrees. Let $A(G)$, $L(G) = A(G) - D(G)$ and $Q(G) = A(G) + D(G)$ be the *adjacency*, *Laplacian* and the *signless Laplacian* matrices of G, respectively. For $M(G) = A(G)$, $L(G)$ or $Q(G)$, let $P_M(G, x)$ be the characteristic polynomial of $M(G)$ and $Sp_M(G)$ the spectrum of $M(G)$. Since 1974, when Harary and Schwenk posed the question *Which graphs have integral spectra?*[5], the search for graphs whose adjacency eigenvalues or Laplacian eigenvalues are all integers (here called *A-integral graphs* and *L-integral graphs*, respectively) has been done. More recently, *Q-integral graphs* (graphs whose signless Laplacian spectrum consists entirely of integers) were introduced in the literature [2, 6, 7, 8, 3].

If G is a r-regular graph,

$$P_A(G, x) \;=\; P_Q(G, x + r) \text{ and } P_L(G, x) \;=\; (-1)^n P_Q(G, 2r - x).$$

So, for regular graphs, these three concepts coincide. If G is a bipartite graph, $P_L(G, x) = P_Q(G, x)$; so, L-integral bipartite graphs and Q-integral bipartite graphs are the same.

A graph is called ALQ-integral graph when it is simultaneously A, L and Q-integral. Among all 172 connected Q-integral graphs up to 10 vertices, there are 42 ALQ-integral graphs, but only one of them is not regular and not bipartite [7]. Our aim is to show how to construct infinite families of ALQ-integral graphs, none of them regular or bipartite.

Firstly, in Section 2, we give ALQ-integrality conditions for join of regular graphs and we build some infinite families of ALQ-integral graphs, from complete graphs, complete bipartite graphs and cycles. In Section 3 we present ALQ-integral infinite families of *complete split* graphs, *multiple complete split-like* graphs and *multiple extended split-like* graphs. All those families were defined in [4].

2 *ALQ*-integral graphs obtained by join of regular graphs

Recall that the *join* of graphs G_1 and G_2 is the graph $G_1 \vee G_2$ obtained from $G_1 \cup G_2$ by joining each vertex of G_1 with every vertex of G_2. For $i = 1, 2$, let G_i be a r_i-regular graph on n_i vertices. Note that if $r_1 \neq r_2$ and $r_1 - r_2 \neq n_1 - n_2$, the graph $G_1 \vee G_2$ is not regular, nor bipartite. It is well known that if G_1 and G_2 are Laplacian integral graphs then $G_1 \vee G_2$ is also a Laplacian integral graph. In the case that G_1 and G_2 are regular graphs, the characteristic polynomial of the matrix $A(G_1 \vee G_2)$ is given in [1]. Under the same conditions, the characteristic polynomial of the matrix $Q(G_1 \vee G_2)$ is obtained in [3].

As a consequence of these two results, we are able to provide a ALQ-integrality condition for join of regular graphs.

Proposition 2.1. *For $i = 1, 2$, let G_i be a r_i-regular graph on n_i vertices. The graph $G_1 \vee G_2$ is ALQ-integral if and only if G_1 and G_2 are ALQ-integral and $(r_1 - r_2)^2 + 4n_1n_2$ and $((2r_1 - n_1) - (2r_2 - n_2))^2 + 4n_1n_2$ are perfect squares.*

Proposition 2.2. *Let G_1 and G_2 be regular ALQ-integral graphs of same degree. The graph $G_1 \vee G_2$ is ALQ-integral if and only if $|G_1|\,|G_2|$ is a perfect square.*

Using the previous proposition, we present two distinct infinite families of ALQ-integral graphs, in the corollary below.

Corollary 2.3. *Let $j, n \in \mathbb{N}$.*

1. *The graph $K_j \vee nK_j$ is ALQ-integral if and only if n is a perfect square;*
2. *If $n = \frac{j(j+1)}{2}$, the graph $K_{j,j} \vee nK_{j+1}$ is ALQ-integral.*

Remark 2.4. *The particular case $j = 2$ of item 1 in Corollary 2.3 was proved in [8].*

It is known that C_n is ALQ-integral only for $n = 3, 4$ or 6. As C_3 coincides with K_3, the construction of ALQ-integral graphs from join of copies of C_3 have been examined in the corollary above. In the following corollary we present other ALQ-integral graphs, constructed from ALQ-cycles.

Corollary 2.5. *Let $n, p, q \in \mathbb{N}$.*

1. *The graph $C_j \vee nC_j$, for $j = 4$ or 6 is ALQ-integral if and only if n is a perfect square;*
2. *If $n = 3^{(2q+1)}$, the graphs $C_3 \vee nC_4$ and $C_4 \vee nC_3$ are ALQ-integral;*
3. *If $n = 2^{(2p+1)}$, the graphs $C_3 \vee nC_6$ and $C_6 \vee nC_3$ are ALQ-integral;*
4. *If $n = 2^{(2p+1)}3^{(2q+1)}$, the graphs $C_4 \vee nC_6$ and $C_6 \vee nC_4$ are ALQ-integral.*

3 Split and split-like *ALQ*-integral graphs

In 2002, Hansen *et al* [4] characterized integral graphs in the classes of *complete split* graphs, *multiple complete split-like* graphs and *multiple extended split-like* graphs. In [3] all signless Laplacian integral graphs in these classes were characterized. We remember the definitions of the three classes that we need for next results.

Definition 3.1. *For a, b, $n \in \mathbb{N}$, we have the following classes of graphs:*

- *the complete split graph $CS_b^a \cong \overline{K_a} \vee K_b$;*
- *the multiple complete split-like graph $MCS_{b,n}^a \cong \overline{K_a} \vee (nK_b)$;*
- *the multiple extended complete split-like graph $MECS_{b,n}^a \cong \overline{K_a} \vee (n(K_b \times K_2))$.*

Applying Proposition 2.1, we are able to characterize the ALQ-integral graphs in the classes above and, in each one, we obtain infinite families of such graphs. Our task in each case, in order to build ALQ-integral graphs, is to solve a system of non linear diophantine equations.

Proposition 3.2. *For a, $b \in \mathbb{N}$, the complete split graph CS_b^a is ALQ-integral if and only if $(b-1)^2 + 4ab$ and $(a+b-2)^2 + 4ab$ are perfect squares. Moreover, for $j_k \in \mathbb{N}$, if one of the conditions below holds, CS_b^a is Q-integral:*

1. *$a = 3j_k - 2$, $b = 2j_k$ and $j_k \in \mathbb{N}$ satisfies*

$$\begin{aligned} j_{k+1} &= 127j_k + 24m_k - 45 \\ m_{k+1} &= 672j_k + 127m_k - 240, \end{aligned} \tag{1}$$

   with $(j_0, m_0) = (1, 3)$ or $(10, 51)$;

2. *$a = 3j_k$, $b = 2j_k - 1$ and $j_k \in \mathbb{N}$ satisfies*

$$\begin{aligned} j_{k+1} &= 127j_k + 484m_k - 45 \\ m_{k+1} &= 336j_k + 127m_k - 120, \end{aligned} \tag{2}$$

   with $(j_0, m_0) = (3, 14)$.

Example 3.3. *Figure 1 shows the ALQ-integral complete split graph $G = CS_5^9$: $Sp_A(G) = (9, 0^8, -1^4, -5)$, $Sp_L(G) = (14^5, 5^8, 0)$ and $Sp_Q(G) = (20, 12^4, 5^8, 2)$, where exponents denote multiplicities.*

Proposition 3.4. *For a, b, $n \in \mathbb{N}$, the multiple complete split-like graph $MCS_{b,n}^a$ is ALQ-integral if and only if $(b-1)^2 + 4abn$ and $(a + 2(b-1) - nb)^2 + 4abn$ is a perfect square. Moreover, for $j \in \mathbb{N}$, if one of the conditions below holds, $MCS_{b,n}^a$ is Q-integral:*

1. *$a = n$, $b = 1$;*
2. *$n \geq 2$, $a = (n-1)b + 1$;*
3. *$n = 2$, $a = j_k$, $b = 3(j_k + 1)$ and $j_k \in \mathbb{N}$ satisfies*

$$\begin{aligned} j_{k+1} &= 23j_k + 4m_k + 12 \\ m_{k+1} &= 132j_k + 23m_k + 72, \end{aligned} \tag{3}$$

   with $(j_0, m_0) = (4, 26)$ or $(20, 118)$;

4. *$n = 3$, $a = j_k$, $b = 2(j_k + 2)$ and $j_k \in \mathbb{N}$ satisfies*

$$\begin{aligned} j_{k+1} &= 127j_k + 24m_k + 135 \\ m_{k+1} &= 672j_k + 127m_k + 720, \end{aligned} \tag{4}$$

   with $(j_0, m_0) = (0, \pm 3)$ or $(12, \pm 69)$.

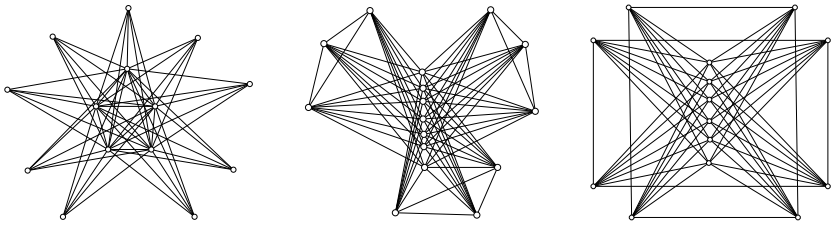


Figure 1: CS_5^9 ,$MCS_{3,3}^7$ and $MECS_{2,2}^6$

Example 3.5. *The multiple complete split-like graph $G = MCS_{3,3}^7$, depicted in Figure 1, is ALQ-integral: $Sp_A(G) = (9, 2^2, 0^6, -1^6, -7)$, $Sp_L(G) = (16, 10^6, 9^6, 7^2, 0)$ and $Sp_Q(G) = (18, 11^2, 9^6, 8^6, 2)$.*

Proposition 3.6. *For a, b, $n \in \mathbb{N}$, the multiple extended complete split-like graph $MECS_{b,n}^a$ is ALQ-integral if and only if $b^2 + 8abn$ and $(a + 2b(n-1))^2 + 8ab$ are perfect squares. Moreover, for $a = (2n-1)b$, $MECS_{b,n}^a$ is ALQ-integral.*

Example 3.7. *Figure 1 shows the ALQ-integral multiple extended complete split-like graph $G = MECS_{2,2}^6$: $Sp_A(G) = (8, 2, 0^9, -2^2, -6)$, $Sp_L(G) = (14, 10^2, 8^9, 6, 0)$ and $Sp_Q(G) = (16, 10, 8^9, 6^2, 2)$.*

Remark 3.8. *In analogy to the result obtained in the Proposition 3.2, we investigate the graphs of type $\overline{K_a} \vee C_4$, and $\overline{K_a} \vee C_6$, for $a \in \mathbb{N}$. For each case, instead of infinite families, we found only one graph: $\overline{K_2} \vee C_4$, and $\overline{K_4} \vee C_6$, respectively, which are both regular and non-bipartite.*

Acknowledgement: The authors are indebted to CNPq(the Brazilian Council for Scientific and Technological Development) for all the support received for this research.

References

[1] Cvetković, D., M. Doob, H. Sachs, "Spectra of graphs-Theory and application," Deutscher Verlag der Wissenschaften-Academic Press, Berlin-New York, 1980; second edition 1982; third edition, Johann Ambrosius Barth Verlag, Heidelberg-Leipzig, 1995.

[2] Cvetković, D., P. Rowlinson and S. Simić, *Signless Laplacian of finite graphs*, Linear Algebra and its Applications **423** (2007), 155–171.

[3] Freitas, M.A.A., N.M.M. de Abreu, R.R. Del-Vecchio and S.Jurkiewicz, *Infinite families of Q-integral graphs*, Linear Algebra Appl. **432** (2010), 2353–2360.

[4] Hansen,P. H. Melot and D, Stevanović, *Integral complete split graphs*, Univ. Beograd, Publ. Elektrotehn. Fak. Ser. Mat. **13** (2002), 89–95.

[5] Harary, F. and A.J. Schwenk, *Which graphs have integral spectra?*, em Bari, R., Harary, F.(Eds), "Graphs and Combinatorics," Springer, Berlim, 1974, 45–51.

[6] Simić, S. and Z. Stanić, *Q-integral graphs with edge-degrees at most five*, Discrete Math. **308** (2008), 4625–4634.

[7] Stanić, Z. *There are exactly 172 connected Q-integral graphs up to 10 vertices*, Novi Sad J. Math. **37** n. 2 (2007), 193–205.

[8] Stanić, Z., *Some results on Q-integral graphs*, Ars Combinatoria **90** (2009), 321–335.

Perfect digraphs and a strong perfect digraph theorem

Stephan Dominique Andres[1] and Winfried Hochstättler[1]

[1]Fakultät für Mathematik und Informatik, FernUniversität in Hagen, Universitätsstr. 1, 58084 Hagen, Germany

1 Introduction

Replacing the chromatic number by the dichromatic number introduced by Neumann-Lara [6] we generalize the notion of perfectness of a graph to digraphs. We give a characterization of perfect digraphs using the notion of perfect graphs. Applying the Strong Perfect Graph Theorem [3], this yields a characterization of perfect digraphs by a set of forbidden induced subdigraphs. Furthermore, modifying a recent proof of Bang-Jensen et al. [1] we show that the recognition of perfect digraphs is co-$\mathcal{NP}$-complete.

2 A strong perfect digraph theorem

First we fix some notation. We only consider digraphs without loops. The *clique number* $\omega(D)$ of a digraph D is the size of the largest bidirectionally complete subdigraph of D. The *dichromatic number* $\chi(D)$ of D is the smallest cardinality $|C|$ of a colour set C, so that it is possible to assign a colour from C to each vertex of D such that for every colour $c \in C$ the subdigraph induced by the vertices coloured with c is acyclic, i.e. it does not contain a directed cycle. The clique number is an obvious lower bound for the dichromatic number. D is called *perfect* if, for any induced subdigraph H of D, $\chi(H) = \omega(H)$.

An (undirected) graph $G = (V, E)$ can be considered as the symmetric digraph $D_G = (V, A)$ with $A = \{(v, w), (w, v) \mid vw \in E\}$. In the following, we will not distinguish between G and D_G. In this way, the dichromatic number of a graph G is its chromatic number $\chi(G)$, the clique number of G is its usual clique number $\omega(G)$, and G is perfect as a digraph if and only if G is perfect as a graph. For us, an *edge* vw in a digraph $D = (V, A)$ is the set $\{(v, w), (w, v)\} \subseteq A$ of two antiparallel arcs, and a *single arc* in D is an arc $(v, w) \in A$ with $(w, v) \notin A$. The *oriented part* $O(D)$ of a digraph $D = (V, A)$ is the digraph (V, A_1) where A_1 is the set of all single arcs of D, and the *symmetric part* $S(D)$ of D is the digraph (V, A_2) where A_2 is the union of all edges of D. Obviously, $S(D)$ is a graph, and by definition we have

Lemma 2.1. *For any digraph D, $\omega(D) = \omega(S(D))$.*

An *odd hole* is an undirected cycle C_n with an odd number $n \geq 5$ of vertices. An *odd antihole* is the complement of an odd hole (without loops). A *filled odd hole/antihole* is a digraph H, so that $S(H)$ is an odd hole/antihole. For $n \geq 3$, the directed cycle on n vertices is denoted by $\vec{C}_n$. Furthermore, for a digraph $D = (V, A)$ and $V' \subseteq V$, by $D[V']$ we denote the subdigraph of D induced by the vertices of V'.

Theorem 2.2. *A digraph $D = (V, A)$ is perfect if and only if $S(D)$ is perfect and D does not contain any directed cycle $\vec{C}_n$ with $n \geq 3$ as induced subdigraph.*

Proof. Proof. Assume $S(D)$ is not perfect. Then there is an induced subgraph $H = (V', E')$ of $S(D)$ with $\omega(H) < \chi(H)$. Since $S(D[V']) = H$, we conclude by Lemma 2.1,

$$\omega(D[V']) = \omega(S(D[V'])) = \omega(H) < \chi(H) = \chi(S(D[V'])) \leq \chi(D[V']),$$

therefore D is not perfect. If D contains a directed cycle $\vec{C}_n$ with $n \geq 3$ as induced subdigraph, then D is obviously not perfect, since $\omega(\vec{C}_n) = 1 < 2 = \chi(\vec{C}_n)$.

Now assume that $S(D)$ is perfect but D is not perfect. It suffices to show that D contains an induced directed cycle of length at least 3. Let $H = (V', A')$ be an induced subdigraph of D such that $\omega(H) < \chi(H)$. Then there is a proper colouring of $S(H) = S(D)[V']$ with $\omega(S(H))$ colours, i.e., by Lemma 2.1, with $\omega(H)$ colours. This cannot be a feasible colouring for the digraph H. Hence there is a (not necessarily induced) monochromatic directed cycle $\vec{C}_n$ with $n \geq 3$ in $O(H)$. Let C be such a cycle of minimal length. C cannot have a chord that is an edge vw, since both terminal vertices v and w of vw are coloured in distinct colours. By minimality, C does not have a chord that is a single arc. Therefore, C is an induced directed cycle (of length at least 3) in H, and thus in D. □

Corollary 2.3. *If D is a perfect digraph, then any feasible colouring of $S(D)$ is also a feasible colouring for D.*

By the Strong Perfect Graph Theorem [3] and Theorem 2.2 we obtain:

Corollary 2.4. *A digraph $D = (V, A)$ is perfect if and only if it does neither contain a filled odd hole, nor a filled odd antihole, nor a directed cycle $\vec{C}_n$ with $n \geq 3$ as induced subdigraph.*

3 Some complexity issues

Corollary 2.3 and the fact that k-colouring of perfect graphs is in $\mathcal{P}$ (see [4]) implies the following.

Corollary 3.1. *k-colouring of perfect digraphs is in $\mathcal{P}$ for any $k \geq 1$.*

To test whether D does not contain an induced directed cycle $\vec{C}_n$, $n \geq 3$, is a co-$\mathcal{NP}$-complete problem by a recent result of Bang-Jensen et al. ([1], Theorem 11). The proof of Bang-Jensen et al. can be easily modified to prove the following.

Theorem 3.2. *The recognition of perfect digraphs is co-$\mathcal{NP}$-complete.*

The preceding result can be obtained by a reduction of 3-SAT to recognition of non-perfect digraphs. This result is in contrast to the result of Chudnovsky et al. [2] which, together with the Strong Perfect Graph Theorem [3], states that the recognition of perfect graphs is in $\mathcal{P}$.

Note that the perfectness of digraphs does not behave as well as the perfectness of graphs in a second aspect: there is no analogon to Lovasz' Weak Perfect Graph Theorem [5]. A digraph may be perfect but its complement may be not perfect. An easy instance of this type is the directed 4-cycle $\vec{C}_4$, which is not perfect, and its complement H, which is perfect.

References

[1] J. Bang-Jensen, F. Havet, N. Trotignon. Finding an induced subdivision of a digraph. Manuscript, submitted for publication, 2010.

[2] M. Chudnovsky, G. Cornuéjols, X. Liu, P. Seymour, K. Vušković. Reognizing Berge graphs. In **Combinatorica** 25:143–186, 2005.

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

[4] M. Grötschel, L. Lovász, A. Schrijver. Geometric algorithms and combinatorial optimization. Second corrected edition, Springer-Verlag, Berlin Heidelberg New York, (1993).

[5] L. Lovász. Normal hypergraphs and the perfect graph conjecture. In **Discrete Math.** 2:253–267, 1972.

[6] V. Neumann-Lara. The dichromatic number of a digraph. In **J. Combin. Theory B** 33:265–270, 1982.

Geodetic Sets and Periphery

Danilo Artigas[1], Simone Dantas[2], Mitre C. Dourado[3,4], and Jayme L. Szwarcfiter[3,4,5]

[1]Instituto de Ciência e Tecnologia, Universidade Federal Fluminense, Brazil
[2]Instituto de Matemática e Estatística, Universidade Federal Fluminense, Brazil
[3]Instituto de Matemática, Universidade Federal do Rio de Janeiro, Brazil
[4]NCE, Universidade Federal do Rio de Janeiro, Brazil
[5]COPPE-Sistemas, Universidade Federal do Rio de Janeiro, Brazil

Let $G = (V, E)$ be a finite, simple and connected graph, and $S \subseteq V$. The geodetic closed interval $I[S]$ is the set of all vertices lying on a shortest path between any pair of vertices of S. The set S is *geodetic* if $I[S] = V$. Let $S \subseteq V$, its monophonic closed interval $J[S]$ is the set of all vertices lying on an induced path between any pair of vertices of S. The set S is *monophonic* if $J[S] = V$. The eccentricity of a vertex v is the number of edges in the greatest shortest path between v and any vertex w of G. The contour $Ct(G)$ of G is the set formed by vertices v such that no neighbor of v has eccentricity greater than v. The diameter $diam(G)$ of G is the maximum eccentricity of the vertices in V. The periphery of G is the set of vertices whose eccentricity is equal to the diameter of G. We consider the problem of determining whether the periphery of a graph is geodetic. First, we establish a relation between the diameter and the geodeticity of the periphery of a graph. We show that the periphery is geodetic for graphs with diameter $k = 2$ and that it is not necessarily geodetic for $k \geq 3$. For $k = 3$, we characterize the graphs whose periphery is not geodetic. Similar results do not extend for graphs with diameter 4. These results lead us to solve the problem for classes of graphs like cographs, chordal, split and threshold graphs. We also consider the monophonic convexity and describe similar results as those for the geodesic convexity.

1 Introduction

Recent works points out the increasing importance of establishing a parallel between concepts of discrete and continuous mathematic like distance and convexity. Some of the early papers that generalized the Euclidean concepts of convex sets to graph theory are [9, 10, 7]. But, convexity in graphs was also studied under different aspects like geodetic sets and hull number [4, 8]. For general information about convexity see [11].

Let $G = (V, E)$ be a graph with vertex set V and edge set E, where $|V| = n$ and $|E| = m$. In this work, all graphs are finite, simple and connected. We say that $G[S]$ is the subgraph of G *induced* by S.

A *geodesic* between v and w in G is a shortest path between v and w in the graph. The *geodetic interval* $I[v,w]$ is the set of all vertices lying on a geodesic between v and w. Given a set S, $I[S] = \bigcup_{u,v \in S} I[u,v]$. If $I[S] = S$, then S is a *g-convex set*. If $I[S] = V$, then S is *geodetic*.

Analogously, we present definitions for the monophonic convexity. The *monophonic interval* $J[v,w]$ is the set of all vertices lying on an induced path between v and w. Given a set S, $J[S] = \bigcup_{u,v \in S} J[u,v]$. If $J[S] = S$, then S is a *m-convex set*. If $J[S] = V$, then S is *monophonic*.

The *length* of a path P between two vertices v and w, denoted by $|P|$, is the number of edges in P. The *distance* in $G = (V,E)$ between v and w, denoted by $d_G(v,w)$, is the length of a geodesic between v and w in G. The *eccentricity* of $v \in V$, denoted by $ecc_G(v)$ is the largest distance from v to any other vertex in G, i.e., $ecc(v) = max\{d_G(v,w) | w \in V\}$. The *diameter* of G, $diam(G)$, is equal to $max\{d_G(v,w) |\ v,w \in V\}$. The *radius* of G, $rad(G)$, is equal to $min\{ecc_G(v) |\ v \in V\}$. For simplicity, we omit G from the notation above. For basic concepts in graph theory see [2].

A vertex v of G such that no neighbor of v has an eccentricity greater than v is called *contour* vertex of G. The *contour* $Ct(G)$ of G is the set formed by all the contour vertices of G. The *periphery* of G is the set of vertices whose eccentricity is equal to the diameter of G.

Contour and periphery sets are well known subjects in the literature. Some examples are [3, 4, 5, 6]. The problem of determining whether $Ct(G)$ is a geodetic set was studied in [1]. It was shown that there exists a strict relation between the diameter of a graph and the geodeticity of its contour. It was proved that if the $diam(G) \leq 4$ then $Ct(G)$ is geodetic and that $Ct(G)$ is not necessarily geodetic if $diam(G) > 4$. Some graph classes were also considered and it was shown that: the contour of a cochordal graph is geodetic; the contour of a planar graph G is not necessarily geodetic if $diam(G) \geq 5$; the contour of a bipartite graph G is not necessarily geodetic if $diam(G) \geq 8$; and the contour of a parity graph G is not necessarily geodetic.

In this work we extend these results to the periphery of a graph G. We show that if $diam(G) \leq 2$ then $Per(G)$ is geodetic, and $Per(G)$ is not necessarily geodetic if $diam(G) > 2$. Particularly, we characterize the graphs G with $diam(G) = 3$ such that $Per(G)$ is not geodetic. These results lead us to solve the problem for classes of graphs like cographs, chordal, split and threshold graphs. Finally, we consider the problem of determining whether $Per(G)$ is a monophonic set. Some proofs will be omitted due to space limits.

2 Periphery × geodetic and monophonic sets

The next results establish limits to the relation between the periphery $Per(G)$ and the diameter of a graph G.

Theorem 2.1. *If $diam(G) \leq 2$, then $Per(G)$ is a geodetic set.*

Proof. The case where $diam(G) = 1$ is trivial. Consider $diam(G) = 2$. The vertices v such that $ecc(v) = 1$ are adjacent to two non-adjacent vertices w_1, w_2 such that $ecc(w_1) = ecc(w_2) = 2$. Hence $v \in I[w_1, w_2]$. □

This result corresponds to a new proof of the result of [4] for the case of graphs with diameter less than or equal to 2.

The above bound is tight since, for each $k \geq 3$ we can generate a graph G with diameter k such that $Per(G)$ is not geodetic. The graph $G = (V,E)$ depicted in Figure 1(a) is a

graph with diameter 3 such that $Per(G) = \{a, e\}$, and $I[Per(G)] \neq V$ because $c \notin I[a, e]$. In Figure 1(b) we construct a more general graph $H = (V', E')$, with diameter $k \geq 3$, such that $I[Per(H)] \neq V'$.

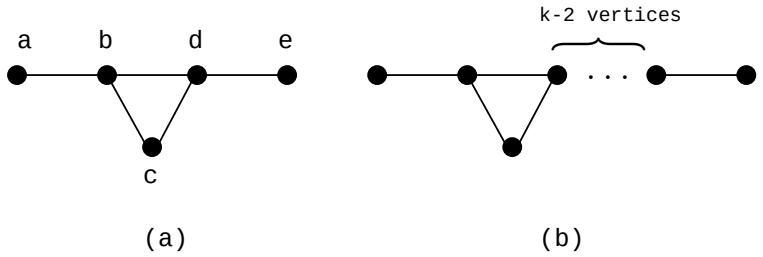


Figure 1: (a) Graph G, with diameter 3, such that $Per(G)$ is not geodetic; (b) Graph H, with diameter k, such that $Per(H)$ is not geodetic.

Consequently,

Lemma 2.2. *For every $k \geq 3$, there exists a graph $G = (V, E)$ with $diam(G) = k$ such that $Per(G)$ is not a geodetic set.*

Graphs G with $diam(G) = 3$ such that $Per(G)$ is not geodetic can be characterized as follows.

Theorem 2.3. *Let $G = (V, E)$ be a graph such that $diam(G) = 3$. Then $Per(G)$ is geodetic if and only if there does not exist a contour vertex $w \in V$ such that $ecc(w) = 2$.*

We observe that Theorem 2.3 could not be generalized for graphs with diameter 4. For example, in Figure 1(a), if we subdivide the edges bc, bd and cd then we obtain a graph G with $diam(G) = 4$ with a contour vertex of eccentricity 3 such that $I[Per(G)]$ is geodetic.

Similar arguments used in this section could be applied to monophonic convexity. Hence, we have the following results.

Theorem 2.4. *If $diam(G) \leq 2$, then $Per(G)$ is a monophonic set.*

Lemma 2.5. *For every $k \geq 3$, there exists a graph $G = (V, E)$ such that $Per(G)$ is not a monophonic set.*

3 Periphery × graph classes

Theorem 2.1 states that for graphs with diameter equal to 2, $Per(G)$ is a geodetic set. Therefore we can establish the following corollaries.

Corollary 3.1. *If G is a cograph, then $Per(G)$ is geodetic.*

Corollary 3.2. *If G is a threshold graph, then $Per(G)$ is geodetic.*

The graph of Figure 1(a), whose periphery is not geodetic, is a split graph and the graph of Figure 1(b) is a chordal graph. In [3], the authors proved that the contour of a chordal graph is geodetic. Hence this result shows a graph class for which the contour is a geodetic set but the periphery is not necessarily geodetic. Particularly, we conclude with the next corollary.

Corollary 3.3. *For every $k \geq 3$, there exists a chordal graph $G = (V, E)$ such that $Per(G)$ is not geodetic.*

Remark 3.4. *If $T = (V, E)$ is a tree with $diam(G) = k, k \geq 3$, then $Per(T)$ is geodetic if and only if $ecc(v) = diam(T)$ for all leaves v of T.*

4 Conclusion

We have considered the problem of deciding whether the periphery of a graph is a geodetic or a monophonic set by establishing limits for the diameter of a graph. We have also characterized graphs with $diam(G) = 3$ such that $Per(G)$ is not geodetic. We show that the same conditions do not generalize for graphs with diameter equal to 4.

In addition, we have described graphs, for which the contour is geodetic and the periphery is not a geodetic set.

References

[1] D. Artigas, S. Dantas, M.C. Dourado, J.L. Szwarcfiter, and S. Yamaguchi. On the contour of graphs. *Discrete Applied Mathematics*, 2013. http://dx.doi.org/10.1016/j.dam.2012.12.024, in press.

[2] J. A. Bondy and U. S. R. Murty. *Graph Theory*. Springer, 2008.

[3] J. Cáceres, C. Hernando, M. Mora, I. M. Pelayo, M. L. Puertas, and C. Seara. Geodeticity of the contour of chordal graphs. *Discrete Applied Mathematics*, 156:1132–1142, 2008.

[4] J. Cáceres, M. C. Hernando, M. Mora, I. M. Pelayo, M. L. Puertas, and C. Seara. On geodetic sets formed by boundary vertices. *Discrete Mathematics*, 306(2):188–198, 2006.

[5] J. Cáceres, A. Márquez, O. R. Oellermann, and M. L. Puertas. Rebuilding convex sets in graphs. *Discrete Mathematics*, 297:26–37, 2005.

[6] G. Chartrand, D. Erwin, G.L. Johns, and P. Zhang. Boundary vertices in graphs. *Discrete Mathematics*, 263:25–34, 2003.

[7] V. D. Chepoi and V. P. Soltan. Conditions for invariance of set diameters under d-convexification in a graph. *Cybernetics and Systems Analysis*, 19(6):750–756, 1983.

[8] M. C. Dourado, J. G. Gimbel, F. Protti, J. L. Szwarcfiter, and J. Kratochvíl. On the computation of the hull number of a graph. *Discrete Mathematics*, 309:5668–5674, 2009.

[9] M. Farber and R. E. Jamison. Convexity in graphs and hypergraphs. *SIAM J. Algebraic Discrete Methods*, 7:433–444, 1986.

[10] F. Harary and J. Nieminen. Convexity in graphs. *Journal of Differential Geometry*, 16:185–190, 1981.

[11] M. J. L. Van de Vel. *Theory of Convex Structures*. North-Holland, Amsterdam, 1993.

An IP based heuristic algorithm for the Vehicle and Crew Scheduling Pick-up and Delivery Problem with Time Windows

D. Bakarcic[1], G. Di Piazza[1], I. Méndez-Díaz[1], and P. Zabala[1,2]

[1]Department of Computer Science, University of Buenos Aires, Buenos Aires, Argentina
[2]Consejo Nacional de Investigaciones Científicas y Técnicas

1 Introduction

In this paper we address the Vehicle and Crew Scheduling Pick-up and Delivery Problem with Time Windows (VCSPDPTW herein): a real life problem involving the planning of the pick-up and delivery of a set of merchandise requests according to certain time restrictions and based on a set of available vehicles. In addition, the allocation of crews to the vehicles which perform each task must also be scheduled. The VCSPDPTW combines features present in three well studied problems: the *Vehicle Routing Problem*, the *Crew Scheduling Problem* and the *Pick-up and Delivery Problem with Time Windows*[1]. As far as we know, this particular problem has not been addressed in the literature before.

In the VCSPDPTW, there is a set of merchandise requests P which must be accomplished by a set of available vehicles V. Each request consists of a transportation task from a pick-up to a drop-off location, being each of these locations potentially different for each request. There are also time restrictions associated with both the pick-up and the drop-off of each request. Each of these events has a time window in which it must be carried out, and a start date indicating that the event cannot be accomplished before that date. A request delivered past the drop-off start date carries a penalty which is a function of the delay. Each vehicle can transport every request in P, but can only do so one at a time. The vehicles can only stop at a specific set of locations L, either to carry out an event or to make a swap in its crew. At any given time, each vehicle's crew is composed of at most two drivers of the set D. Each driver can only travel between locations using the available vehicles, and the vehicles can only be driven by drivers in D. In addition, certain work regulations regarding the drivers' work shifts, such as the amount of working hours or consecutive working days, must be complied.

The objective is to provide a planning, that is, a set of *routes* for the vehicles and a *schedule* for the crew, over a given time horizon of T days, in order to ensure that all requests are accomplished at minimum operation costs. These costs are only associated with the vehicle routes and are composed of the total travel distance, the penalty in the delay of drop-offs, and a special penalty associated with travelling without a load. The proposed solution involves stating the problem as an integer programming model in combination with a column generation approach, in which we generate the columns associated with the vehicle routes. The column generation subproblem is solved by searching optimal paths in a resource constrained network.

This work was partially supported by UBACYT 20020100100666, PICT 2010-304 and 2011-817

2 Model

In the proposed formulation we have five kind of variables, all of which are binary. Let R be the set of all feasible routes, variables y_r indicate if a route $r \in R$ is part of the solution. Variables X_{dki} indicate if a driver $d \in D$ gets on a vehicle in location $k \in L$ at moment $i \in M$, where M denotes the set of all the time moments included in the time horizon. The cardinality of M equals $T \times K$, where K represents the granularity in which a day is divided (e.g. for hours, $K = 24$). Similarly, variables W_{dki} indicate if a driver d gets off a vehicle in location k at moment i. The set of variables Y_{dki} indicate if a driver d is resting in location k at moment i, and the set of variables Z_{dj} specify if a driver d is off duty on day j.

The objective function minimizes the cost of the routes which are part of the solution:

$$min \sum_{r \in R} c_r y_r \qquad subject\ to$$

1. $\sum_{r \in RP_p} y_r = 1 \qquad \forall p \in P$, where RP_p is the set of routes which accomplish p
2. $\sum_{r \in R} y_r \leq |V|$
3. $\sum_{r \in RSL_k} y_r \leq |VSL_k| \qquad \forall k \in LVI$
4. $Y_{dki} = Y_{dk(i-1)} - X_{dki} + W_{dki} \qquad \forall d \in D, k \in L, 1 \leq i < K \times T$
5. $Y_{dk0} = 1 - X_{dk0} \qquad \forall d \in D$, where k is the initial location of driver d
6. $\sum_{k \in L}(Y_{dki} + X_{dki}) \leq 1 \qquad \forall d \in D, 1 \leq i < K \times T$
7. $\sum_{r \in D1_{ki}} y_r + 2\sum_{r \in D2_{ki}} y_r = \sum_{d \in D} W_{dki} \qquad \forall k \in L, 1 \leq i < K \times T$
8. $\sum_{r \in P1_{ki}} y_r + 2\sum_{r \in P2_{ki}} y_r = \sum_{d \in D} X_{dki} \qquad \forall k \in L, 0 \leq i < K \times T$
9. $\sum_{k \in L} \sum_{j=i}^{i+K-1} Y_{dkj} \geq K/2 \qquad \forall d \in D, 0 \leq i < K \times T - (K-1)$
10. $\sum_{i=j}^{j+6} Z_{di} \geq 1 \qquad \forall d \in D, 0 \leq j \leq T - 7$
11. $KZ_{dj} \leq \sum_{k \in L} \sum_{i=K \times j}^{K \times j+K-1} Y_{dki} \qquad \forall d \in D, 0 \leq j < T$

Constraints (1) establish that all requests are satisfied by exactly one route. Constraint (2) establishes that the amount of routes must not exceed the number of available vehicles. Constraints (3) ensure that for every location where there is initially at least one vehicle (LVI), the amount of routes which start on that location (RSL_k) is less or equal than the amount of available vehicles on that location (VSL_k). Constraints (4), (5) and (6) establish the relation between the resting and working hours of the drivers.

Let $D1_{ki}$, $D2_{ki}$, $P1_{ki}$, $P2_{ki}$ be the sets of routes that drop or pick one or two drivers respectively in location k at moment i. Constraints (7) and (8) establish that the amount of drivers who are dropped or picked by a vehicle in a certain location at a given time must match the amount of drivers who are getting off/on a vehicle in that location at that time. Constraints (9), (10) and (11) ensure that the drivers' working regulations are complied.

The proposed model has an exponential number of variables and therefore cannot be formulated explicitly. To address this issue we use the column generation technique to solve the LP relaxation (usually called *master problem*), where the generated columns correspond to the route variables y_r whilst the rest of the variables are considered explicitly.

3 Column generation approach

The main idea behind our approach is to start with a restricted set of columns, obtaining as result a restricted master problem, and iteratively add columns with negative reduced costs until the master problem is solved to optimality or certain stopping criteria is met. The initial restricted set of columns is provided by a greedy heuristic algorithm that generates a set of routes which represent a feasible solution. Once the column generation stage is complete, we solve the resulting model using a Branch & Cut algorithm. It is important to notice that with this approach we only generate columns on the root node of the decision tree and therefore the integer solution may not be optimal.

To solve the column generation subproblem, we need to generate a route whose associated column has a negative reduced cost. In order to do this, we designed the *route graph*, a network in which a directed path from source node s to sink node t represents a route, that is, a valid sequence of actions preformed by a vehicle (pick-up request p, drop a driver at location k, etc). The validity of a path is enforced by verifying certain constraints regarding the resources accumulated by it, which are the time and the accomplished requests. These constraints ensure that a path does not exceed the time horizon, accomplishes each request at most once and satisfies the time windows restrictions. The constraints regarding the vehicles' crew are not verified by resources in the route graph, hence the graph structure must ensure that the drivers' limit is not exceeded. Crew swaps are modeled by two set of nodes per location which represent the arrival and departure of the vehicle and how many drivers get off/on the vehicle respectively. Edges connecting the arrival with the departure only exist if the resulting crew has at least one and no more than two drivers. Regarding path costs, the dual costs obtained from solving the LP relaxation are introduced as part of the edge costs in the route graph. In this way, the cost of a path between s and t (route) matches the reduced cost of its associated column, therefore, the subproblem of column generation becomes searching for a path with a negative cost in the route graph. For this purpose, we use a dynamic programming algorithm, based on the one proposed in [2], which solves the *Resource Constrained Shortest Path Problem* (RCSPP) over the route graph. If the cost of the resulting path is negative, the column can be added to the formulation. Otherwise, the optimal solution of the LP relaxation has been reached. When either this happens or the specified time or number of generated columns is exceeded, the column generation stage is terminated. See [3] for an in-depth description of the route graph and the subproblem resolution approach.

Since the RCSPP is NP-hard, we developed two versions of the algorithm: an exact version and an heuristic one. The latter redefines the notion of path dominance of the former by relaxing some of the conditions. This allows the heuristic to discard a greater number of paths than the exact algorithm, making it faster. Being that in each iteration of the column generation process we need a negative reduced cost column, and not necessarily the one with the minimum cost, we can first run the heuristic and, if that fails, run the exact algorithm. Additionally, and with the objective of speeding the column generation process, we added extra constraints to the routes structure, such as limiting the amount of visited locations and forbidding duplicate visits. This reduces the search space for both algorithms.

To improve the chances of finding an integer solution, we considered adding more than one column in each iteration. We proposed two alternatives. The first one is to add not only the minimum but all negative reduced cost columns found. The second one is to generate a set of columns which complement the already generated ones. A route r complements a set of routes S by making drivers available on locations and moments in which the routes in S require them.

4 Computational experience

We conducted experiments considering two different kind of instances: small sized and big sized. The size of an instance is given by its number of locations and requests. All instances were randomly generated using real geographical information. Algorithms were coded in C++ using the optimizer and the default B&C algorithm provided by CPLEX 12.2. The small instances tests were run on a laptop with an Intel i3 2.13 GHz and 4 GB of RAM whilst the big instances were run on an Intel i7 3.40 GHz and 16 GB of RAM. Since there are no previous experiments on the VCSPDPTW, we report the improvement percentage (%Imp.) between the initial solution, provided by our greedy algorithm, and the resulting solution after the column generation process and B&C optimization.

The table below shows the results for the small instances using four different configurations for the column generation algorithm. The selected configurations consider constrained (C) vs. unconstrained (U) route generation and minimum reduced cost column (MRC) vs. all negative reduced cost columns (ANRC) aggregation. We also run tests considering the alternative complementary route generation but the results showed that it did not contribute to improve the solution so they are not included here.

Inst.	#L	#P	Conf. U-ANRC			Conf. C-ANRC			Conf. U-MRC			Conf. C-MRC		
			% Imp.	% Gap	Time	% Imp.	% Gap	Time	% Imp.	% Gap	Time	% Imp.	% Gap	Time
small-1	6	10	0	0	608	35,93	0	173	0	3	***	35,93	0	79
small-2	6	6	0	0	623	0	27	***	0	0	620	14,66	37	***
small-3	6	8	0	34	***	0	41	***	0	0	616	42,26	17	***
small-4	6	10	0	15	***	0	52	***	0	18	***	0	0	613
small-5	7	12	0	12	***	10,37	0	710	0	0	614	53,33	0	587

A cell filled with (***) means that the algorithm could not solve the instance to optimality within 900 seconds. This time includes both the time used by the column generation process and the B&C optimization. Based on the results showed in the above table, we can conclude that constrained route generation brings better results. We cannot draw any conclusion regarding the use of ANRC columns. The table below shows the results for the big instances using four different configurations considering less constrained (LC) vs. more constrained (MC) column generation and MRC vs. ANRC column aggregation. In these tests, the time limit was set to 72000 seconds (8 hours for column generation and 4 hours for B&C).

Inst.	#L	#P	Conf. MC-MRC			Conf. MC-ANRC			Conf. LC-MRC			Conf. LC-ANRC		
			% Imp.	% Gap	Time	% Imp.	% Gap	Time	% Imp.	% Gap	Time	% Imp.	% Gap	Time
big-1	10	16	0	0	28803	0	0	28803	0	0	28803	1,64	0	29070
big-2	10	16	0	0	28802	0	0	28803	0	45,22	***	0	12,11	***
big-3	10	16	0	0	28805	8,36	0	28806	0	0	28988	0	22,11	***

We consider that our approach produces promissory computational results. As future research, and aiming to obtain a Branch & Price algorithm, it would be interesting to obtain a deeper insight on the structure of the problem in order to derive a robust branching rule that preserves the structure of the pricing subproblem.

References

[1] B. Golden and S. Raghavan and E. Wasil (Editors). *The Vehicle Routing Problem: Latest Advances and New Challenges - Springer*, 2010.

[2] Faramroze G. Engineer and George L. Nemhauser and Martin W. P. Savelsbergh. *Shortest Path Based Column Generation on Large Networks with Many Resource Constraints*, 2008.

[3] D. Bakarcic and G. Di Piazza. *Ruteo de vehículos y asignación de conductores: un enfoque combinado*, 2012.

Online Checkpointing with Improved Worst-Case Guarantees

Karl Bringmann[1], Benjamin Doerr[1], Adrian Neumann[1], and Jakub Sliacan[1]

[1]Max Planck Institute for Informatics, Saarbrücken, Germany

In the online checkpointing problem, the task is to continuously maintain a set of k checkpoints that allow to rewind an ongoing computation faster than by a full restart. The only operation allowed is to remove an old checkpoint and to store the current state instead. Our aim are checkpoint placement strategies that minimize rewinding cost, i.e., such that at all times T when requested to rewind to some time $t \leq T$ the number of computation steps that need to be redone to get to t from a checkpoint before t is as small as possible. In particular, we want that the closest checkpoint earlier than t is not further away from t than p_k times the ideal distance $T/(k+1)$, where p_k is a small constant.

Improving over earlier work showing $1 + 1/k \leq p_k \leq 2$, we show that p_k can be chosen less than 2 uniformly for all k. More precisely, we show the uniform bound $p_k \leq 1.7$ for all k, and present algorithms with asymptotic performance $p_k \leq 1.59 + o(1)$ valid for all k and $p_k \leq \ln(4) + o(1) \leq 1.39 + o(1)$ valid for k being a power of two. For small values of k, we show how to use a linear programming approach to compute good checkpointing algorithms. This gives performances of less than 1.53 for $k \leq 10$.

One the more theoretical side, we show the first lower bound that is asymptotically more than one, namely $p_k \geq 1.30 - o(1)$. We also show that optimal algorithms (yielding the infimum performance) exist for all k.

1 Introduction

Checkpointing means storing intermediate states of a long sequence of computations. This allows reverting the system into a previous state much faster, since only the computations from the preceding checkpoint have to be redone. Checkpointing is one of the fundamental techniques in computer science. Classic results date back to the seventies, e.g. [4] and the references therein. More recent topics are checkpointing in distributed [3] systems and sensor networks [5].

Checkpointing usually involves doing a trade-off between the speed-up of reversions to previous states and the costs incurred by setting checkpoints (time, memory). Much of the classic literature studies checkpointing with the focus of recovering from immediately detectable faults. Consequently, only reversions to the most recent checkpoint are needed. On the negative side, setting checkpoints is expensive, because the whole system state has to be copied to

secondary memory. In such a scenario, the central question is how often to set a checkpoint such that the expected time (under a stochastic failure model) spent on setting checkpoints and redoing computations from the last checkpoint is minimized.

In this work, we will regard a checkpointing problem, where the cost of setting checkpoints is small compared to the cost of regular computation and checkpoints can be kept in main memory. In this scenario, the memory used by *stored* checkpoints is the bottleneck. Applications of this type arise for example in data compression [2].

The first to provide a framework independent of a particular application were Ahlroth, Pottonen and Schumacher [1]. They do not make assumptions on which reversion will be requested, but simply investigate how checkpoints can be set in an online fashion such that at all times their distribution is balanced over the computation history.

They assume that the system is able to store up to k checkpoints (plus a free checkpoint at time 0). At any point in time, a checkpoint may be discarded and replaced by the current state as new checkpoint, while ignoring the cost of this operation.

Each set of checkpoints, together with the current state and the state at time 0, partitions the time from the process start to the current time T into $k+1$ disjoint intervals. Clearly, without further problem-specific information, an ideal set of checkpoints would lead to all these intervals having identical length. Of course, this is not possible due to the restriction that new checkpoints can only be set on the current time. As performance measure for a checkpointing algorithm, Ahlroth et al. mainly regard the *maximum gap ratio*, that is, the ratio of the longest vs. the shortest interval (ignoring the last interval), maximized over all current times T. They show that there is a simple algorithm achieving a performance of two, and that not much improvement is possible for general k. They show a lower bound of $2^{1-1/\lceil (k+1)/2 \rceil} = 2(1-o(1))$. For small values of k, namely $k = 2, 3, 4$, and 5, better upper bounds of approximately $1.414, 1.618, 1.755$, and 1.755, respectively, were shown.

In this work, we regard a different, and, as we find, more natural performance measure. Recall that the cost of reverting to a particular state is basically the cost of redoing the computation from the preceding checkpoint to the desired point in time. Our aim is then to keep the length of the longest interval small (at all times). As a performance measure, we compare the length of the longest interval to the length $T/(k+1)$ of a (at time T) optimal partition into equal length intervals. The *quality* $q(A,T)$ of an algorithm A at time $T \geq t_k$is calculated as

$$q(A,T) := (k+1)\bar{\ell}_T/T,$$

where $\bar{\ell}_T$ denotes the length of the longest interval at time T. The *maximum distance performance* (or simply performance) $\mathrm{Perf}(A)$ is then the supremum over the quality over all times T, i.e.,

$$\mathrm{Perf}(A) := \sup_{T \geq t_k} q(A,T).$$

This performance measure was suggested in [1], where it is remarked that an upper bound of β for the gap-ratio implies an upper bound of $\beta(1+\frac{1}{k})$ for the maximum distance performance. Moreover for all k an upper bound of 2 and a lower bound of $1+\frac{1}{k}$ for the performance is presented. For small $k \leq 5$, stronger upper bounds were shown.

Our work substantially improves both upper and lower bounds. In particular we show that both are bounded away from 1, respectively 2, by a constant.

2 Results

2.1 New Checkpointing Algorithms

We consider a class of algorithms that remove old checkpoints in a periodic pattern, e.g., removing checkpoints at odd positions ordered from oldest to newest and starting again from the beginning once the most recent checkpoint is removed. They place new checkpoints such that after one period the intervals are a scaled copy of the initial intervals. We call such algorithms *cyclic*. Cyclic algorithms are particularly easy to analyze, as their performance can be found by looking at just a single period. We establish the first upper bounds that are asymptotically bounded away from 2 by a constant, as well as improved upper bounds for select values of k.

For any fixed k, we show how to find good cyclic algorithms by solving a series of linear feasibility problems. For a fixed pattern P of length n of checkpoint removals and a given scaling factor, we can set up a system of inequalities for the $k+n$ time points when a checkpoint is placed that is satisfiable if positions exist that yield a performance of at most λ when checkpoints are removed according to P. Binary search over λ together with a linear search for the scaling factor yields a good algorithm for P. For $k < 8$ we can exhaustively try all patterns up to length k. By searching for good patterns via randomized local search we can find good algorithms even for larger k. In experiments we find upper bounds of 1.53 and 1.54 for $k = 3$, respectively $k = 4$ as well as bounds of at most 1.49 for $k \in [5, 59]$.

For large k we construct a simple algorithm Linear with a performance of $1.59 + O(k^{-1})$. It places its checkpoints at times $t_i = (i/k)^\alpha$. In the analysis we optimize over α and choose $\alpha = 1.302$. In one period, Linear removes all checkpoints at odd positions, starting with the oldest. We analyze the performance by bounding the size of the intervals created by deleting an old checkpoint and storing a new one. It turns out that the former are always larger than the latter. Their scaled size can be bounded as

$$\begin{aligned}(k+1)\frac{t_{2(i-k)} - t_{2(i-k)-2}}{t_i} &= (k+1)\frac{(2(i-k))^\alpha - (2(i-k)-2)^\alpha}{i^\alpha} \\ &\leq (k+1)2^\alpha \frac{\alpha(i-k)^{\alpha-1}}{i^\alpha},\end{aligned}$$

where we used again $(x+1)^c - x^c \leq c(x+1)^c$. An easy computation shows that $(i-k)^{\alpha-1}/i^\alpha$ is maximized at $i = \alpha k$ over $k < i \leq 2k$. Hence, we can upper bound this quality by

$$\leq \Big(1 + \frac{1}{k}\Big)2^\alpha \frac{\alpha(\alpha-1)^{\alpha-1}}{\alpha^\alpha} = 2^\alpha \Big(1 - \frac{1}{\alpha}\Big)^{\alpha-1} + O(k^{-1}).$$

For $\alpha = 1.302$ this is $1.586 + O(k^{-1})$.

For k being a power of two we show an even better upper bound. We give an algorithm Binary with performance,

$$\mathrm{Perf}(\textsc{Binary}) = \ln(4) + \frac{0.05}{\lg(k/4)} + O(k^{-1}) \quad \approx 1.39.$$

Unlike Linear, that places its checkpoints on a polynomial, Binary places checkpoints on $\lg k$ exponential curves. Again all checkpoints at odd positions are removed within one period, but the order is more involved. Recursively, checkpoints from the right half of the interval $[t_1, t_k]$ are removed twice as often as those from the right half of the interval $[t_1, t_{k/2}]$.

Experimentally we verify that both LINEAR and BINARY yield very good performance already for moderate values of k. Combining experimental and asymptotic bounds, we can give a uniform upper bound of 1.7 for all k.

2.2 Lower Bound

We also present the first lower bound that is larger than 1 be constant, namely we prove

$$\mathrm{Perf}(A) \geq 1.3 - O(k^{-1}),$$

for any checkpointing algorithm A. This bound leaves room for only 6% improvement over the algorithm BINARY. Our proof works by assuming that A can reshuffle new checkpoints at will, and only the intervals created by deleting old checkpoints are fixed. We bound $p := \mathrm{Perf}(A)$ by analyzing A over the first $k/(2p)$ steps. We bound the size of intervals created by deletion, intervals created by insertion and intervals that remain untouched throughout. These bounds allow us to set up inequalities for p to obtain $p \geq 1.3 - O(k^{-1})$.

2.3 Existence of Optimal Algorithms

On the more theoretical side, we establish the seemingly simple result that algorithms OPT_k exist such that

$$\mathrm{Perf}(\mathrm{OPT}_k) = \inf_A \mathrm{Perf}(A),$$

where A runs over all algorithms that use k checkpoints. Hence, the infimum can be replaced by a minimum. Surprisingly, this is trivial not at all.

Our proof works by showing that initial positions with optimal quality exist and then combining a sequence of algorithms that each do a small number of quality preserving steps.

References

[1] L. Ahlroth, O. Pottonen, and A. Schumacher. Approximately uniform online checkpointing with bounded memory. *Algorithmica*, to appear.

[2] M. Bern, D. H. Greene, A. Raghunathan, and M. Sudan. Online algorithms for locating checkpoints. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, STOC '90, pages 359–368. ACM, 1990.

[3] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message-passing systems. *ACM Comput. Surv.*, 34:375–408, 2002.

[4] E. Gelenbe. On the optimum checkpoint interval. *J. ACM*, 26:259–270, 1979.

[5] F. Österlind, A. Dunkels, T. Voigt, N. Tsiftes, J. Eriksson, and N. Finne. Sensornet checkpointing: Enabling repeatability in testbeds and realism in simulations. In U. Roedig and C. J. Sreenan, editors, *Wireless Sensor Networks*, volume 5432 of *Lecture Notes in Computer Science*, pages 343–357. Springer, 2009.

Smoothed Analysis of the Successive Shortest Path Algorithm*

Tobias Brunsch[1], Kamiel Cornelissen[2], Bodo Manthey[2], and Heiko Röglin[1]

[1]University of Bonn, Department of Computer Science, Germany.
[2]University of Twente, Department of Applied Mathematics, Enschede, The Netherlands.

The minimum-cost flow problem is a classic problem in combinatorial optimization with various applications. Several pseudo-polynomial, polynomial, and strongly polynomial algorithms have been developed in the past decades, and it seems that both the problem and the algorithms are well understood. However, some of the algorithms' running times observed in empirical studies contrast the running times obtained by worst-case analysis not only in the order of magnitude but also in the ranking when compared to each other. For example, the Successive Shortest Path (SSP) algorithm, which has an exponential worst-case running time, seems to outperform the strongly polynomial Minimum-Mean Cycle Canceling algorithm. To explain this discrepancy, we study the SSP algorithm in the framework of smoothed analysis and establish a bound of $O(mn\phi(m + n\log n))$ for its smoothed running time. This shows that worst-case instances for the SSP algorithm are not robust and unlikely to be encountered in practice.

1 Introduction

Flow problems have gained a lot of attention in the second half of the twentieth century to model, for example, transportation and communication networks [1]. Plenty of pseudo-polynomial, polynomial, and strongly polynomial algorithms have been developed for the minimum-cost flow problem over the last fifty years. The fastest known strongly polynomial algorithm up to now is the Enhanced Capacity Scaling algorithm due to Orlin [10] and it has a running time of $O(m\log(n)(m + n\log n))$. For an extensive overview of minimum-cost flow algorithms we suggest the book of Ahuja, Magnanti, and Orlin [1].

Zadeh [14] showed that the Successive Shortest Path (SSP) algorithm has an exponential worst-case running time. Contrary to this, the worst-case running time of the strongly polynomial Minimum-Mean Cycle Canceling (MMCC) algorithm is $O(m^2n^2\min\{\log(nC), m\})$ [11]. Here, C denotes the maximum edge cost. However, the notions of pseudo-polynomial, polynomial, and strongly polynomial algorithms always refer to worst-case running times, which do not always resemble the algorithms' behavior on real-life instances. Algorithms with large

*This research was supported by ERC Starting Grant 306465 (BeyondWorstCase) and NWO grant 613.001.023. It was presented at SODA 2013.

worst-case running times do not inevitably perform poorly in practice. An experimental study of Király and Kovács [7] indeed observes running time behaviors significantly deviating from what the worst-case running times indicate. The MMCC algorithm is completely outperformed by the SSP algorithm. In this paper, we explain why the SSP algorithm comes off so well by applying the framework of smoothed analysis.

Smoothed analysis was introduced by Spielman and Teng [12] to explain why the simplex method is efficient in practice despite its exponential worst-case running time. In the original model, an adversary chooses an arbitrary instance which is subsequently slightly perturbed at random. In this way, pathological instances no longer dominate the analysis. Good smoothed bounds usually indicate good behavior in practice because in practice inputs are often subject to a small amount of random noise. For instance, this random noise can stem from measurement errors, numerical imprecision, or rounding errors. It can also model influences that cannot be quantified exactly but for which there is no reason to believe that they are adversarial. Since its invention, smoothed analysis has been successfully applied in a variety of contexts. We refer to [9] for a recent survey.

We follow a more general model of smoothed analysis due to Beier and Vöcking [2]. In this model, the adversary is even allowed to specify the probability distribution of the random noise. The power of the adversary is only limited by the *smoothing parameter* ϕ. In particular, in our input model the adversary does not fix the edge costs $c_e \in [0,1]$ for each edge e, but he specifies probability density functions $f_e\colon [0,1] \to [0,\phi]$ according to which the costs c_e are randomly drawn independently of each other. If $\phi = 1$, then the adversary has no choice but to specify a uniform distribution on the interval $[0,1]$ for each edge cost. In this case, our analysis becomes an average-case analysis. On the other hand, if ϕ becomes large, then the analysis approaches a worst-case analysis since the adversary can specify small intervals I_e of length $1/\phi$ (that contain the worst-case costs) for each edge e from which the costs c_e are drawn uniformly.

As in the worst-case analysis, the network graph $G = (V, E)$, the edge capacities $u(e) \in \mathbb{R}^+$, and the balance values $b(v) \in \mathbb{R}$ of the nodes – indicating how much of a resource a node requires ($b(v) < 0$) or offers ($b(v) > 0$) – are chosen adversarially. We define the smoothed running time of an algorithm as the worst expected running time the adversary can achieve and we prove the following theorem.

Theorem 1. *The smoothed running time of the SSP algorithm is* $O(mn\phi(m + n\log n))$.

If ϕ is a constant – which seems to be a reasonable assumption if it models, for example, measurement errors – then the smoothed bound simplifies to $O(mn(m + n\log n))$. Hence, it is unlikely to encounter instances on which the SSP algorithm requires an exponential amount of time. Still, this bound is worse than the bound $O(m\log(n)(m + n\log n))$ of Orlin's Enhanced Capacity Scaling algorithm, but this coincides with practical observations.

In practice, an instance of the minimum-cost flow problem is usually first transformed to an equivalent instance with only one *source* (a node with positive balance value) s and one *sink* (a node with negative balance value) t. The SSP algorithm then starts with the empty flow f_0. In each iteration i, it computes the shortest path P_i from the source s to the sink t in the residual network and maximally augments the flow along P_i to obtain a new flow f_i. The algorithm terminates when no $s - t$ path is present in the residual network.

Theorem 2. *In any round* i*, flow* f_i *is a minimum-cost* b_i*-flow for the balance function* b_i *defined by* $b_i(s) = -b_i(t) = |f_i|$ *and* $b_i(v) = 0$ *for* $v \notin \{s,t\}$.

Theorem 2 is due to Jewell [6], Iri [5], and Busacker and Gowen [4]. As a consequence, no residual network G_{f_i} contains a directed cycle with negative total costs. Otherwise, we could augment along such a cycle to obtain a b_i-flow f' with smaller costs than f_i.

2 Terminology and Notation

Consider the run of the SSP algorithm on the flow network G. We denote the set $\{f_0, f_1, \ldots\}$ of all flows encountered by the SSP algorithm by $\mathcal{F}_0(G)$. Furthermore, we set $\mathcal{F}(G) = \mathcal{F}_0(G) \setminus \{f_0\}$. (We omit the parameter G if it is clear from the context.)

By f_0, we denote the empty flow, i.e., the flow that assigns 0 to all edges e. Let f_{i-1} and f_i be two consecutive flows encountered by the SSP algorithm and let P_i be the shortest path in the residual network $G_{f_{i-1}}$, i.e., the SSP algorithm augments along P_i to increase flow f_{i-1} to obtain flow f_i. We call P_i the *next path* of f_{i-1} and the *previous path* of f_i. To distinguish between the original network G and some residual network G_f in the remainder of this paper, we refer to the edges in the residual network as *arcs*, whereas we refer to the edges in the original network as *edges*.

For a given arc e in a residual network G_f, we denote by e_0 the corresponding edge in the original network G, i.e., $e_0 = e$ if $e \in E$ (i.e. e is a forward arc) and $e_0 = e^{-1}$ if $e \notin E$ (i.e. e is a backward arc). An arc e is called *empty* (with respect to some residual network G_f) if e belongs to G_f, but e^{-1} does not. Empty arcs e are either forward arcs that do not carry flow or backward arcs whose corresponding edge e_0 carries as much flow as possible.

3 Outline of Our Approach

Our analysis of the SSP algorithm is based on the following idea: We identify a flow $f_i \in \mathcal{F}_0$ with a real number by mapping f_i to the length ℓ_i of the previous path P_i of f_i. The flow f_0 is identified with $\ell_0 = 0$. In this way, we obtain a sequence $L = (\ell_0, \ell_1, \ldots)$ of real numbers. We show that this sequence is strictly monotonically increasing with a probability of 1. Since all costs are drawn from the interval $[0, 1]$, each element of L is from the interval $[0, n]$. To count the number of elements of L, we partition the interval $[0, n]$ into small subintervals of length ε and sum up the number of elements of L in these intervals. By linearity of expectation, this approach carries over to the expected number of elements of L. If ε is very small, then – with sufficiently high probability – each interval contains at most one element. Thus, it suffices to bound the probability that an element of L falls into some interval $(d, d + \varepsilon]$.

For this, assume that there is an integer i such that $\ell_i \in (d, d + \varepsilon]$. By the previous assumption that for any interval of length ε there is at most one path whose length is within this interval, we obtain that $\ell_{i-1} \leq d$. We show that the augmenting path P_i uses an empty arc e. Moreover, we will see that we can reconstruct flow f_{i-1} without knowing the cost of edge e_0 that corresponds to arc e in the original network. Hence, we do not have to reveal c_{e_0} for this. However, the length of P_i, which equals ℓ_i, depends linearly on c_{e_0}, and the coefficient is $+1$ or -1. Consequently, the probability that ℓ_i falls into the interval $(d, d + \varepsilon]$ is bounded by $\varepsilon\phi$, as the probability density of c_{e_0} is bounded by ϕ. Since the arc e is not always the same, we have to apply a union bound over all $2m$ possible arcs. Summing up over all n/ε intervals the expected number of flows encountered by the SSP algorithm can be bounded by roughly $(n/\varepsilon) \cdot 2m \cdot \varepsilon\phi = 2mn\phi$.

There are some parallels to the analysis of the smoothed number of Pareto-optimal solutions in bicriteria linear optimization problems by Beier and Vöcking [3], although we have only one objective function. In this context, we would call f_i the loser, f_{i-1} the winner, and the difference $\ell_i - d$ the loser gap. Beier and Vöcking's analysis is also based on the observation that the winner (which in their analysis is a Pareto-optimal solution and not a flow) can be reconstructed when all except for one random coefficients are revealed. While this reconstruction is simple in the setting of bicriteria optimization problems, the reconstruction of the flow f_{i-1} in our setting is significantly more challenging and a main difficulty in our analysis.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows – theory, algorithms and applications.* Prentice Hall, 1993.

[2] René Beier and Berthold Vöcking. Random knapsack in expected polynomial time. *Journal of Computer and System Sciences*, 69(3):306–329, 2004.

[3] René Beier and Berthold Vöcking. Typical properties of winners and losers in discrete optimization. *SIAM Journal on Computing*, 35(4):855–881, 2006.

[4] Robert G. Busacker and Paul J. Gowen. A procedure for determining a family of miminum-cost network flow patterns. Technical Paper 15, Operations Research Office, Johns Hopkins University, 1960.

[5] Masao Iri. A new method for solving transportation-network problems. *Journal of the Operations Research Society of Japan*, 3(1,2):27–87, 1960.

[6] William S. Jewell. Optimal flow through networks. *Oper. Res.*, 10(4):476–499, 1962.

[7] Zoltán Király and Péter Kovács. Efficient implementations of minimum-cost flow algorithms. *Acta Universitatis Sapientiae, Informatica*, 4(1):67–118, 2012.

[8] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms.* Springer, 4th edition, 2007.

[9] Bodo Manthey and Heiko Röglin. Smoothed analysis: analysis of algorithms beyond worst case. *it – Information Technology*, 53(6):280-286, 2011.

[10] James B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Oper. Res.*, 41(2):338–350, 1993.

[11] Tomasz Radzik and Andrew V. Goldberg. Tight bounds on the number of minimum-mean cycle cancellations and related results. *Algorithmica*, 11(3):226–242, 1994.

[12] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.

[13] Roman Vershynin. Beyond hirsch conjecture: Walks on random polytopes and smoothed complexity of the simplex method. *SIAM Journal on Computing*, 39(2):646–678, 2009.

[14] Norman Zadeh. A bad network problem for the simplex method and other minimum cost flow algorithms. *Mathematical Programming*, 5(1):255–266, 1973.

The Spanning Tree Problem with One Quadratic Term

Christoph Buchheim[1] and Laura Klein[1]

[1]Fakultät für Mathematik, TU Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany,
{christoph.buchheim,laura.klein}@tu-dortmund.de

1 Introduction

A common approach to quadratic optimization over binary variables is to linearize the quadratic terms and to develop an appropriate polyhedral description of the corresponding set of feasible solutions. A straightforward idea is to linearize each product in the objective function independently and simply combine the result with the given linear side constraints. This approach yields a correct integer programming model of the problem, but the resulting LP-relaxations lead to very weak bounds in general, so that branch-and-cut algorithms based on this simple linearization idea perform very poorly in general. For this reason, one usually searches for stronger or even facet defining inequalities to tighten the description; see, e.g., [2].

In this paper, we consider another approach that, to the best of our knowledge, has not been investigated yet: we examine the problem version with only one product term in the objective function, but with all linear side constraints taken into account. Any valid cutting plane for this problem will remain valid for the original problem as well and potentially improves over the straightforward model, since the chosen product is considered together with all side constraints. The advantage of our approach lies in the fact that there exists a polynomial time separation algorithm for the one-product problem whenever the underlying linear problem is tractable: in this case, the corresponding optimization problem is polynomially solvable by solving the underlying linear problem four times, with different fixings of the two variables appearing in the chosen product.

From a practical point of view, this indirect separation approach does not pay off within a branch-and-cut algorithm; the effort for computing a single cutting plane is too high. Therefore, we apply this idea to an important specific problem in the following, namely the quadratic minimum spanning tree (QMST) problem. The linear spanning tree problem is well studied and solvable in polynomial time, while additional costs for pairs of edges render the problem NP-hard [1]. We introduce two new classes of facet defining inequalities for QMST with only one quadratic term in the objective function. These inequalities arise from certain subtour elimination constraints by adding the new product variable to the left hand side. We actually conjecture that the addition of these classes of inequalities suffices to obtain a complete polyhedral description of the linearized polytope in the one-product case. The new inequalities can be separated in polynomial time by adapting the separation routines for subtour elimination constraints. Our experimental results show a significant improvement of both bounds and running times when adding the new inequalities to a branch-and-cut algorithm for solving the original problem with all quadratic terms, with respect to the straightforward linearization.

2 Preliminaries

We assume that $G = (V, E)$ is a complete undirected graph. The *quadratic minimum spanning tree problem* can be formulated as an integer program with linear constraints and a quadratic objective function:

$$
\begin{aligned}
(\mathrm{QIP}_{\mathrm{QMST}}) \qquad \min \quad & \sum_{e \in E} c_e x_e + \sum_{e,f \in E,\ e \neq f} c_{ef} x_e x_f && (1)\\
\text{s.t.} \quad & \sum_{e \in E} x_e = |V| - 1 && (2)\\
& \sum_{e \in E(G[S])} x_e \leq |S| - 1 \qquad \forall\ \emptyset \neq S \subsetneq V && (3)\\
& x_e \in \{0,1\} \qquad \forall\ e \in E.
\end{aligned}
$$

Here $G[S]$ denotes the subgraph of G induced by the vertices in S and $E(G[S])$ denotes its edge set. The subtour elimination constraints (3) ensure that no subgraph induced by S contains a cycle; combined with Equation (2), this also guarantees connectivity.

To get rid of the quadratic terms in the objective function, we linearize all products $x_e x_f$ by introducing artificial binary variables y_{ef} and link them to the original variables using the following additional linear inequalities:

$$
\begin{aligned}
& y_{ef} \leq x_e, x_f \qquad \forall\ e, f \in E && (4)\\
& y_{ef} \geq x_e + x_f - 1 \qquad \forall\ e, f \in E. && (5)
\end{aligned}
$$

The x-entries of all feasible solutions of the linearized problem ($\mathrm{QIP}_{\mathrm{QMST}}$) model exactly the incidence vectors of all spanning trees, and due to the binary constraints, the value of every y_{ef} is exactly the product of x_e and x_f by (4) and (5). The latter does not remain true, however, after relaxing integrality.

When considering a quadratic objective function with a single product term, we have to distinguish between two cases. In the first case, the product term consists of variables of two adjacent edges. We denote these edges by $e_1 := \{u, v\}$ and $e_2 := \{v, w\}$ and the product of their variables is called a *connected monomial*. The corresponding problem is denoted by QMST^c in the following. In the second case, the edges of the product variables are non-adjacent in the graph, therefore, the edges are $e_1 := \{u, v\}$ and $e_2 := \{w, z\}$ with pairwise distinct vertices $u, v, w, z \in V$. We refer to a *disconnected monomial* and denote the problem by QMST^d. As the context leads to the correct association, we shortly denote the linearization variables $y_{\{u,v\}\{v,w\}}$ and $y_{\{u,v\}\{w,z\}}$ by y.

Our aim is thus to investigate the polytope corresponding to QMST^c, defined as

$$
P^c := \mathrm{conv}\left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \{0,1\}^{|E|+1} \;\middle|\; x \text{ satisfies (2) to (3) and } y = x_{\{u,v\}} x_{\{v,w\}} \right\}
$$

and the polytope corresponding to QMST^d, defined as

$$
P^d := \mathrm{conv}\left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \{0,1\}^{|E|+1} \;\middle|\; x \text{ satisfies (2) to (3) and } y = x_{\{u,v\}} x_{\{w,z\}} \right\}.
$$

3 Polyhedral Results

In the following we assume $|V| \geq 4$. The dimension of the (linear) spanning tree polytope P^{lin} is $|E| - 1$. Clearly, the additional linearization variable y increases the dimension by at most one. In fact, we have

Theorem 3.1.

$$\dim(P^c) = \dim(P^d) = \dim(P^{\text{lin}}) + 1 = |E|.$$

The following results introduce two classes of facet defining inequalities for the polytopes P^c and P^d, respectively. They strengthen the subtour elimination constraints (3); we call them *quadratic subtour elimination constraints* in the following.

Theorem 3.2. *Let $S \subset V$ be a set of vertices with $u, w \in S$ and $v \notin S$. Then the inequality*

$$\sum_{e \in E(G[S])} x_e + y \leq |S| - 1 \tag{6}$$

induces a facet of P^c.

Theorem 3.3. *Let $S_1, S_2 \subset V$ be disjoint subsets of vertices such that both edges $\{u, v\}$ and $\{w, z\}$ have exactly one end node in S_1 and one end node in S_2. Then the inequality*

$$\sum_{e \in E(G[S_1])} x_e + \sum_{e \in E(G[S_2])} x_e + y \leq |S_1| + |S_2| - 2 \tag{7}$$

induces a facet of P^d.

In fact, we conjecture that the classes of facets described in Theorems 3.2 and 3.3 yield a complete polyhedral description of P^c and P^d, respectively.

Conjecture 3.4.

$$P^c = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in [0,1]^{|E|+1} \;\middle|\; \begin{pmatrix} x \\ y \end{pmatrix} \text{ satisfies (2), (3), (4), (5), and (6)} \right\}$$

$$P^d = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in [0,1]^{|E|+1} \;\middle|\; \begin{pmatrix} x \\ y \end{pmatrix} \text{ satisfies (2), (3), (4), (5), and (7)} \right\}.$$

All three classes of subtour elimination constraints (3), (6) and (7) are of exponential size, so that these inequalities cannot be separated by enumeration. Therefore, a polynomial time separation routine is required. For (3), a well-known separation algorithm is based on a minimal cut algorithm. Using an appropriate adaption of this algorithm, we can show the following result:

Theorem 3.5. *The separation problems for the classes of inequalities* (6) *and* (7) *are polynomial time solvable.*

The problems can be reduced to solving one maximum s-t-flow problem for (6) and eight such problems for (7), corresponding to different fixings.

4 Experimental Results

Our aim is to determine the impact of the new inequalities for instances with more or all quadratic terms in the objective function. We implemented the separation algorithm of Theorem 3.5 and embedded it into the branch-and-cut software SCIL [4]. We considered the basic problem formulation using Constraints (2) to (5) where the subtour elimination constraints were separated dynamically and no further reformulation was applied (*stdlin*). For comparison we additionally separated the quadratic subtour elimination constraints (6) or (7) for each of the appearing products (*+qsec*). We tested graphs with edge densities of 33 %, 67 % and 100 %. The absolute weights are chosen randomly between 1 and 10, similar to the instances used in [3].

We discovered that, in the case of positive product weights, the separation of quadratic subtour elimination constraints does not improve the bounds at all, so we consider the case of negative product weights in the following. The separation for both connected and disconnected monomials leads to slightly longer running times but has a positive influence on the bounds and numbers of subproblems. In terms of running times the best approach turns out to be the separation for only connected monomials even if other monomials exist.

Actually, in most applications, the objective function contains only connected monomials. In this case using quadratic subtour elimination constraints (6) leads to significant improvements over *stdlin*, see the table below. Each line corresponds to the average of four tested instances.

edges	nodes	sep	# subs	# LPs	cputime	septime	rootgap
10	33 %	stdlin	5.00	8.50	0.01	0.00	1.63 %
		+qsec	1.00	3.25	0.00	0.00	0.00 %
	67 %	stdlin	38.50	45.50	0.14	0.00	23.65 %
		+qsec	27.50	32.75	0.21	0.05	21.91 %
	100 %	stdlin	3869.00	3635.25	36.39	0.42	72.31 %
		+qsec	1694.50	1788.25	24.91	3.42	71.46 %
15	33 %	stdlin	1434.00	1260.00	3.04	0.24	42.58 %
		+qsec	393.00	452.75	2.13	0.49	33.67 %
	67 %	stdlin	790.00	689.50	17.70	0.23	26.11 %
		+qsec	329.50	332.00	12.02	2.39	24.20 %

One can see that the separation time (*septime*) increases slightly but the bounds are improved significantly, leading to better root gaps (*rootgap*) and numbers of subproblems (*# subs*). Altogether, the running times (*cputime*) decrease considerably, showing the potential of our approach of considering only single product terms.

References

[1] A. Assad and W. Xu. The quadratic minimum spanning tree problem. *Naval Research Logistics (NRL)*, 39(3):399–417, 1992.

[2] C. Buchheim, F. Liers, and M. Oswald. Speeding up IP-based algorithms for constrained quadratic 0–1 optimization. *Mathematical Programming B*, 124(1–2):513–535, 2010.

[3] R. Cordone and G. Passeri. Solving the quadratic minimum spanning tree problem. *Applied Mathematics and Computation*, 218(23):11597–11612, 2012.

[4] SCIL – Symbolic Constraints in Integer Linear programming. `scil-opt.net`.

Robust Optimization Under Multiband Uncertainty

Christina Büsing[1], Fabio D'Andreagiovanni[2,3], and Annie Raymond[2]

[1]Chair of Operations Research, RWTH Aachen University, Kackertstrasse 7, 52072 Aachen, Germany, buesing@or.rwth-aachen.de
[2]Department of Optimization, Zuse-Institut Berlin (ZIB), Takustrasse 7, 14195 Berlin, Germany, {d.andreagiovanni, raymond}@zib.de
[3]Department of Computer, Control and Management Engineering, Sapienza Università di Roma, via Ariosto 25, 00185 Roma, Italy

We provide an overview of the main results that we obtained studying uncertain mixed integer linear programs when the uncertainty is represented through the new multiband model [4]. Such model extends and refines the classical one proposed by Bertsimas and Sim [2] and is particularly suitable in the common case of arbitrary non-symmetric distributions of the uncertainty. Our investigations were inspired by the practical needs of our industrial partners in the German research project ROBUKOM [8].

1 Introduction

A central assumption in classical optimization is that all coefficients of the considered problem are known exactly. However, many real-world problems involve uncertain data and neglecting these uncertainties may have dramatic effects: optimal solutions may reveal to be of very bad quality and solutions supposed to be feasible may turn out to be infeasible.

Over the past few years, Robust Optimization (RO) has increasingly gained attention as a valid methodology to tackle uncertainties affecting optimization problems. The key feature of RO is to take into account uncertainty as hard constraints which restrict the feasible set, thus maintaining only *robust* solutions, i.e. solutions protected from data deviations. For an exhaustive introduction to the theory and applications of RO, we refer the reader to the recent survey by Bertsimas et al. [1].

In this work, we focus on multiband uncertainty, a new approach to model uncertainty that refines and generalizes the widely known Γ-scenario set (BS) of Bertsimas and Sim [2]. The uncertainty model BS assumes that, for each coefficient a of the problem, we know the nominal value $\overline{a}$ as well as the maximum deviation d and that the actual value of a lies in the symmetric interval $[\overline{a} - d, \overline{a} + d]$. Moreover, a parameter Γ is introduced to represent the maximum number of coefficients that deviate from their nominal value. This parameter also controls the conservativeness of the robust model. A central result of BS is that the robust counterpart of an LP can be formulated as a compact linear problem. However, the use of a single deviation band may greatly limit the power of modeling uncertainty, as noted even by

Sim and his colleagues in [7]. This is particularly evident in real-world problems, where it is common to have *asymmetric probability distributions* for the deviations, that are additionally defined over *asymmetric* intervals. In such cases, neglecting the inner-band behavior and just considering the extreme values like in BS leads to a rough estimate of the deviations and produce over-conservative robust solutions. Having a higher modeling resolution is therefore highly desirable. This can be accomplished by a simple operation: breaking the single band of BS into multiple narrower bands, each with its own Γ value, as we do in multiband uncertainty.

The idea of using multiple bands was originally proposed for portfolio optimization [3], but this applied-oriented study was (surprisingly) not followed by a theoretical study. Our main objective has thus been to close the knowledge gap about the use of multiband uncertainty in RO.

For a comprehensive presentation of our results we refer the reader to [4, 5, 6]. In [4] we have presented two of the fundamental results of the study, namely: 1) the robust counterpart of a mixed integer linear program can be formulated as a *compact* mixed integer linear program; 2) the separation of robustness cuts can be done by solving a min-cost flow problem. A refinement of the results is presented in [5] and finally in [6] the study was extended by investigating special properties of uncertain binary programs as well as the probability bounds of constraint violation.

We note that our results are not obtained by simply extending the proofs of [2] for single-band uncertainty, but required alternative proof strategies.

2 Multiband uncertainty in Robust Optimization

We study the robust counterpart of the following Mixed-Integer Linear Program (MILP):

$$\max \sum_{j\in J} c_j\, x_j \qquad \text{s.t.} \quad \sum_{j\in J} a_{ij}\, x_j \le b_i\,,\;\; i \in I = \{1,\dots,m\}, \tag{1}$$

$$x_j \ge 0\,,\;\; j \in J = \{1,\dots,n\}, \quad x_j \in \mathbb{Z}^+,\;\; j \in J^{\mathbb{Z}} \subseteq J\,. \tag{2}$$

We assume that the value of each coefficient a_{ij} is uncertain and that the uncertainty is modeled through what we call a *multiband uncertainty set* $\mathcal{S}_M$. Specifically, we assume that, for each coefficient a_{ij}, we are given its nominal value $\overline{a}_{ij}$ and maximum negative and positive deviations $d_{ij}^{K^-}, d_{ij}^{K^+}$ from $\overline{a}_{ij}$, such that the actual value a_{ij} lies in the interval $[\bar{a}_{ij} + d_{ij}^{K^-}, \bar{a}_{ij} + d_{ij}^{K^+}]$. Moreover, we derive a generalization of the Bertsimas-Sim model by partitioning the single deviation band $[d_{ij}^{K^-}, d_{ij}^{K^+}]$ of each coefficient a_{ij} into K bands, defined on the basis of K deviation values: $-\infty < d_{ij}^{K^-} < \cdots < d_{ij}^{-2} < d_{ij}^{-1} < d_{ij}^{0} = 0 < d_{ij}^{1} < d_{ij}^{2} < \cdots < d_{ij}^{K^+} < +\infty$.

Through these deviation values, we define: 1) a set of positive deviation bands, such that each band $k \in \{1,\dots,K^+\}$ corresponds to the range $(d_{ij}^{k-1}, d_{ij}^{k}]$; 2) a set of negative deviation bands, such that each band $k \in \{K^- + 1,\dots,-1,0\}$ corresponds to the range $(d_{ij}^{k-1}, d_{ij}^{k}]$ and band $k = K^-$ corresponds to the single value $d_{ij}^{K^-}$ (the interval of each band but $k = K^-$ is thus open on the left). With a slight abuse of notation, we denote a generic deviation band by the index k, with $k \in K = \{K^-,\dots,-1,0,1,\dots,K^+\}$ and the corresponding range by $(d_{ij}^{k-1}, d_{ij}^{k}]$.

Additionally, for each band $k \in K$, we define a lower bound l_k and an upper bound u_k on the number of deviations that may fall in k, with $l_k, u_k \in \mathbb{Z}$ satisfying $0 \le l_k \le u_k \le n$. In the case of band k = 0, we assume that $u_0 = n$, i.e. we do not limit the number of coefficients that

take their nominal value. We also assume that $\sum_{k\in K} l_k \leq n$, so that there exists a feasible realization of the coefficient matrix.

Consider a feasible solution x and a constraint i of MILP and denote by $DEV_i(x, \mathcal{S}_M)$ the maximum overall deviation allowed by the multiband uncertainty set $\mathcal{S}_M$, then the robust counterpart of MILP can be defined by adding $DEV_i(x, \mathcal{S}_M)$ to each constraint $i \in I$, namely $\sum_{j\in J} a_{ij}\, x_j + DEV_i(x, \mathcal{S}_M) \leq b_i$. Since $DEV_i(x, \mathcal{S}_M)$ corresponds to a binary maximization program (see [4] for details), the resulting robust counterpart is actually a (non-linear) max-max problem. However, we prove that this problem can be reformulated as a compact and linear problem. For lack of space in the present extended abstract, we state only informally the main results of our investigations. We refer the reader to [4, 5, 6] for the formal complete statements and proofs of the presented theorems.

Theorem 2.1. *The robust counterpart of MILP under the multiband uncertainty set is equivalent to a compact mixed integer linear program, which includes $K \cdot m + n \cdot m$ additional variables and $K \cdot n \cdot m$ additional constraints.*

As an alternative to the direct solution of the compact and linear robust counterpart, we have also investigated the possibility of adopting a cutting-plane approach. Given a solution to MILP, we want to test if the solution is robust feasible. If not, we separate a cut that imposes robustness (*robustness cut*), we add it to the problem and we solve again the problem including the new cut. This basic step can be iterated as in a typical cutting-plane method until a robust feasible solution is found. In the case of the Bertsimas-Sim model, the problem of separating a robustness cut for a given constraint is very simple and essentially consists in sorting the deviations in increasing order and choose the worst $\Gamma > 0$. In the case of multiband uncertainty, this simple approach does not guarantee the robustness of a computed solution. However, we prove the following result:

Theorem 2.2. *The separation of a robustness cut for a constraint of MILP can be done in polynomial time by solving a* min-cost flow problem*.*

We refer again the reader to [4, 5, 6] for the formal statement and the detailed description of how we build the min-cost flow instance and structure the corresponding proof.

2.1 Binary Programs with cost uncertainty

In the case of pure binary programs where the uncertainty only affects the objective function, the results presented above can be refined. To this end, consider the following Binary Program:

$$\begin{aligned} \min \quad & \sum_{j\in J} c_j\, x_j \qquad\qquad (BP) \\ & x \in X \subseteq \{0,1\}^n\ , \end{aligned}$$

with non-negative cost vector, i.e. $c_j \geq 0$, for all $j \in J = \{1, \ldots, n\}$. Relevant problems such as the minimum spanning tree problem, the maximum weighted matching problem and the shortest path problem belong to this class of problems.

We have studied the robust version of the previous problem when only the cost coefficients are uncertain and uncertainty is modeled through a multiband set. More formally, for each element $j \in J$, we are given the nominal cost $\bar{c}_j$ and a sequence of $K^+ + 1$ deviation values d_j^k, with $k \in K = \{0, \ldots, K^+\}$, such that $0 = d_j^0 < d_j^1 < \ldots < d_j^{K+} < \infty$ (note that in contrast to

the previous section here w.l.o.g. we consider only positive deviations). Through these values, we define: 1) the zero-deviation band corresponding to the single value $d_j^0 = 0$; 2) a set K^+ of positive deviation bands, such that each band $k \in K \backslash \{0\}$ corresponds to the range $(d_j^{k-1}, d_j^k]$. Furthermore, integer values $l_k, u_k \in \mathbb{Z}$, with $0 \leq l_k \leq u_k \leq n$, represent the lower and upper bounds on the number of deviations falling in each band $k \in K$.

Since BP is a special case of MILP, we can solve it by referring to its compact robust counterpart or by adopting a cutting-plane algorithm based on the separation of robustness cuts, as shown above. However, as an alternative to these two approaches, we have proved the following result:

Theorem 2.3. *The robust optimal solution of BP with cost uncertainty modeled through a multiband set can be computed by solving a polynomial number of nominal problems BP with modified objective function, if the number of bands is constant. Tractability and approximability of BP are maintained.*

We refer the reader to [6] for the formal statement of the result. Our study has been completed by computational experiments on realistic network instances, defined in collaboration with our industrial partners in past and ongoing research projects. In particular, the experiments have highlighted a reduction in the price of robustness, thanks to the refined representation of the uncertainty obtained through the multiband model.

References

[1] Bertsimas, D., Brown, D., Caramanis, C.: Theory and Applications of Robust Optimization. SIAM Review 53 (3), 464–501 (2011)

[2] Bertsimas, D., Sim, M.: The Price of Robustness. Oper. Res., 52 (1), 35–53 (2004)

[3] Bienstock, D.: Histogram models for robust portfolio optimization. J. Computational Finance, 11, 1–64 (2007)

[4] Büsing, C., D'Andreagiovanni, F.: New Results about Multiband Uncertainty in Robust Optimization. In: Klasing, R. (ed.) Experimental Algorithms - SEA 2012, LNCS, vol. 7276, pp. 63-74. Springer, Heidelberg (2012)

[5] Büsing, C., D'Andreagiovanni, F.: New Results about Multiband Uncertainty in Robust Optimization. CoRR abs/1208.6322, http://arxiv.org/abs/1208.6322 (2012)

[6] Büsing, C., D'Andreagiovanni, F.: Robust Optimization under Multiband Uncertainty - Part I: Theory. Submitted for publication (2012) (preprint: Optimization Online 13-01-3748)

[7] Chen, X., Sim, M., Peng, S.: A Robust Optimization Perspective on Stochastic Programming. Oper. Res. 55 (6), 1058–1071 (2007)

[8] Koster, A.M.C.A., Helmberg, C., Bley, A., Grötschel, M., Bauschert, T.: BMBF Project ROBUKOM: Robust Communication Networks. In: ITG Workshop Euro View 2012, pp. 1–2, VDE Verlag, Berlin (2012)

Connected Dominating Set in Graphs Without Long Paths And Cycles

Eglantine Camby[1] and Oliver Schaudt[2]

[1]Département de Mathématique, Université Libre de Bruxelles, Boulevard du Triomphe, 1050 Brussels, Belgium,
[2]Combinatoire et Optimisation, Université Pierre et Marie Curie, 4 place Jussieu, 75252 Paris, France,

The ratio of the connected domination number, γ_c, and the domination number, γ, is strictly bounded from above by 3. It was shown by Zverovich that for every connected (P_5, C_5)-free graph, $\gamma_c = \gamma$.

We investigate the interdependence of γ and γ_c in the class of (P_k, C_k)-free graphs, for $k \geq 6$. We prove that for every connected (P_6, C_6)-free graph, $\gamma_c \leq \gamma + 1$ holds, and there is a family of (P_6, C_6)-free graphs with arbitrarily large values of γ attaining this bound. Moreover, for every connected (P_8, C_8)-free graph, $\gamma_c/\gamma \leq 2$, and there is a family of (P_7, C_7)-free graphs with arbitrarily large values of γ attaining this bound. In the class of (P_9, C_9)-free graphs, the general bound $\gamma_c/\gamma < 3$ is asymptotically sharp.

1 Introduction

A full paper version of this extended abstract, including proofs and more details, is available online at http://www.zaik.uni-koeln.de/∼schaudt/DS-PoC.pdf.

A *dominating set* of a graph G is a vertex subset X such that every vertex not in X has a neighbor in X. The minimum size of a dominating set of G is called the *domination number* of G and is denoted by $\gamma(G)$. A dominating set of size $\gamma(G)$ is called a *minimum dominating set*.

Dominating sets have been intensively studied in the literature. The main interest in dominating sets is due to their relevance on both theoretical and practical side. Moreover, there are interesting variants of domination and many of them are well-studied. A good introduction into the topic is given by Haynes, Hedetniemi and Slater [7].

A *connected dominating set* of a graph G is a dominating set X whose induced subgraph, henceforth denoted $G[X]$, is connected. The minimum size of such a set of a connected graph G, the *connected domination number* of G, is denoted by $\gamma_c(G)$. A connected dominating set of size $\gamma_c(G)$ is called a *minimum connected dominating set*. A connected dominating set such that every proper subset is not a connected dominating set is called *minimal connected dominating set*. Among the applications of connected dominating sets is the routing of messages in mobile ad-hoc networks. Blum, Ding, Thaeler and Cheng [1] explain the usefulness of connected dominating sets in this context.

A first impression of the relation of γ_c and γ is given by Duchet and Meyniel [4].

Observation 1.1 (Duchet and Meyniel [4]). *For every connected graph it holds that* $\gamma_c \leq 3\gamma - 2$.

As an immediate consequence of Observation 1.1,

$$\gamma_c/\gamma < 3. \tag{1}$$

Loosely speaking, the price of connectivity for minimum dominating sets, γ_c/γ, is strictly bounded by 3.

Let P_k be the induced path on k vertices and let C_k be the induced cycle on k vertices. It is easy to see that

$$\lim_{k\to\infty} \gamma_c(P_k)/\gamma(P_k) = 3 = \lim_{k\to\infty} \gamma_c(C_k)/\gamma(C_k). \tag{2}$$

Hence, the upper bound (1) is asymptotically sharp in the class of paths and in the class of cycles.

The price of connectivity has been introduced by Cardinal and Levy [3, 9] for the vertex cover problem. They showed that it was bounded by $2/(1+\varepsilon)$ in graphs with average degree εn, where n is the number of vertices. In a companion paper to the present one, the price of connectivity for vertex cover is studied by Camby, Cardinal, Fiorini and Schaudt [2]. In a similar spirit, Schaudt [11] studied the ratio between the connected domination number and the total domination number. Fulman [5] and Zverovich [13] investigated the ratio between the independence number and the upper domination number. Many results in this area concern graph classes defined by forbidden induced subgraphs. This line of research stems from the classical theory of perfect graphs, for which the clique number and the chromatic number are equal in every induced subgraph [6].

Motivated by (2), we study the interdependence of γ_c and γ in graph classes defined by forbidden induced paths and cycles. For this we use the following standard notation. If G and H are two graphs, we say that G is *H-free* if H does not appear as an induced subgraph of G. Furthermore, if G is H_1-free and H_2-free for some graphs H_1 and H_2, we say that G is *(H_1, H_2)-free*. Our starting point is the following result by Zverovich [12].

Theorem 1.2 (Zverovich [12]). *The following assertions are equivalent for every graph G.*

1. *For every connected induced subgraph of G it holds that $\gamma_c = \gamma$.*
2. *G is (P_5, C_5)-free.*

We aim for similar bounds in the class of (P_k, C_k)-free connected graphs for $k \geq 6$. The properties of connected dominating sets in P_k-free graphs have been studied before, e.g. by Liu, Peng and Zhao [10] and later van 't Hof and Paulusma [8].

Apart from the previous work, this research has an algorithmic motivation. The proofs of our results are constructive in the sense that it is possible to draw polynomial time algorithms from them. These algorithms can be used to build, given a dominating set of size k, a connected dominating set of size at most $f(k)$, for the suitable function f provided by the respective theorem. We do not explicitely give the algorithms, but leave it as a possible future application of our results.

2 Our Results

Our first result establishes the upper bound $\gamma_c \leq \gamma + 1$ in the class of connected (P_6, C_6)-free graphs.

Theorem 2.1. *For every connected (P_6, C_6)-free graph it holds that $\gamma_c \leq \gamma + 1$.*

To see that the bound given by Theorem 2.1 is best possible, consider the following family of connected (P_6, C_6)-free graphs. For each $k \in \mathbb{N}$, let F_k be the graph obtained from a k disjoint copies of C_4, by picking one vertex from every copy and identifying these picked vertices to a single vertex x, and afterwards attaching a path of length 2 to x (see Fig. 1). It is easy to see that, for all k, $\gamma_c(F_k) = \gamma(F_k) + 1$. Moreover, the graph F_k does not have a minimum connected dominating set that contains a minimum dominating set as subset.

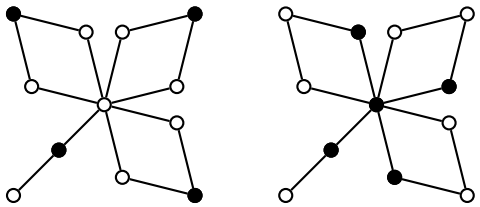


Figure 1: The black vertices indicate a minimum dominating set (resp. a minimum connected dominating set) of F_3.

Theorem 2.2. *For every connected (P_8, C_8)-free graph it holds that $\gamma_c/\gamma \leq 2$.*

The bound provided by Theorem 2.2 is attained by an infinite number of connected (P_7, C_7)-free graphs, given by the following construction. For every $k \in \mathbb{N}$, let H_k be the graph defined as follows (cf. Fig. 2). Start with k paths P^1, P^2, ..., P^k on three vertices each. For every $1 \leq i \leq k$, choose an end-vertex v_i of P^i. Let H_k be the graph obtained from the disjoint union of all P^i, $1 \leq i \leq k$, by adding all possible edges between the vertices v_i, $1 \leq i \leq k$. So, $H_k[\{v_i : 1 \leq i \leq k\}]$ is a complete graph. It is easily seen that, for all $k \in \mathbb{N}$, $\gamma_c(H_k)/\gamma(H_k) = 2$.

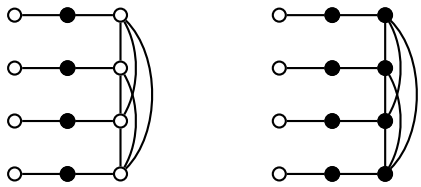


Figure 2: The black vertices indicate a minimum dominating set (resp. a minimum connected dominating set) of H_4.

A similar construction shows that (1) is asymptotically sharp in the class of connected (P_9, C_9)-free graphs, in the sense that there is a family $\{G_k : k \in \mathbb{N}\}$ of (P_9, C_9)-free graphs such that $\lim_{k\to\infty} \gamma_c(G_k)/\gamma(G_k) = 3$. For every $k \in \mathbb{N}$ let G_k be the graph obtained by attaching a pendant vertex to every pendant vertex of H_k. It is easy to check that for every $k \geq 2$, $\gamma(G_k) = k + 1$ and $\gamma_c(G_k) = 3k$. Furthermore, G_k is (P_9, C_9)-free.

3 A Conjecture

We close this abstract with a conjecture that came up during our research. As Theorem 2.2 shows, $\gamma_c \leq 2\gamma$ holds in every connected (P_8, C_8)-free graph. However, $\gamma_c(P_8)/\gamma(P_8) = 2 = \gamma_c(C_8)/\gamma(C_8)$, i.e., both P_8 and C_8 do not violate the bound given by Theorem 2.2.

Conjecture 3.1. *For every connected (P_9, C_9, H)-free graph, $\gamma_c \leq 2\gamma$ (see Fig. 3 for H).*

Note that P_9, C_9, and H violate $\gamma_c \leq 2\gamma$. Hence, if true, Conjecture 3.1 would give a characterization of the largest graph class that is closed under connected induced subgraphs where $\gamma_c \leq 2\gamma$ holds.

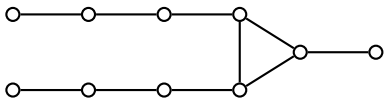


Figure 3: The graph H from Conjecture 3.1.

References

[1] J. Blum, M. Ding, A. Thaeler, X. Cheng, Connected Dominating Set in Sensor Networks and MANETs, pp. 329–369, In: D.-Z. Du, P.M. Pardalos (Eds.), *Handbook of Combinatorial Optimization*, Springer US, Boston, 2005.

[2] E. Camby, J. Cardinal, S. Fiorini, O. Schaudt, The Price of Connectivity for vertex cover: Perfect, Near-Perfect and Critical Graphs. in preparation.

[3] J. Cardinal, E. Levy, Connected vertex covers in dense graphs, Theor. Comput. Sci. 411 (2010), pp. 2581–2590.

[4] P. Duchet, H. Meyniel, On Hadwiger's number and the stability number, Ann. Discrete Math. 13 (1982), pp. 71–74.

[5] J. Fulman, A note on the characterization of domination perfect graphs, J. Graph Theory 17 (1993) pp. 47–51.

[6] M.C. Golumbic, Algorithmic graph theory and perfect graphs, North Holland 57 (2004).

[7] T.W. Haynes, S.T. Hedetniemi, P.J. Slater, Fundamentals of domination in graphs, CRC 208 (1998).

[8] P. van 't Hof, D. Paulusma, A new characterization of P_6-free graphs. Discrete Appl. Math. 158 (2010), pp. 731–740.

[9] E. Levy. (2009). Approximation Algorithms for Covering Problems in Dense Graphs. Ph.D. thesis. Université libre de Bruxelles, Brussels.

[10] J. Liu, Y. Peng, C. Zhao, Characterization of P_6-free graphs. Discrete Appl. Math. 155 (2007), pp. 1038–1043.

[11] O. Schaudt, On graphs for which the connected domination number is at most the total domination number. Discrete Appl. Math. 160 (2012), pp. 1281–1284.

[12] I.E. Zverovich, Perfect connected-dominant graphs. Discuss. Math. Graph Theory 23 (2003), pp. 159–162.

[13] I.E. Zverovich, V.E. Zverovich, A semi-induced subgraph characterization of upper domination perfect graphs, J. Graph Theory 31 (1999), pp. 29–49.

Total $L(2,1)$-coloring of graphs

Márcia R. Cerioli[1,2] and Daniel F. D. Posner[2]

[1]Instituto de Matemática
[2]COPPE/Sistemas e Computação, Universidade Federal do Rio de Janeiro, Caixa Postal 68511, 21945-970, Rio de Janeiro, Brazil

We rise the problem of total $L(2,1)$-coloring a graph and solve it for paths, cycles, stars, wheels, and complete graphs. Moreover, we present tight lower bounds and upper bounds for λ_T, the maximum integer used in an optimum total $L(2,1)$-coloring, for complete q-partite graphs and for trees. Furthermore, we show that the total $L(2,1)$-coloring problem is $\mathcal{NP}$-complete even for bipartite graphs.

1 Introduction

Let $G = (V, E)$ be a simple and undirected graph, with $n = |V|$, and $m = |E|$. The *distance* between two vertices u and v of G is the number of edges in a shortest path between them in G. For a vertex u, denote by $N(u)$ the set of vertices adjacent to u in G and by $N_e(u)$ the set of edges incident to it. Let $d(u) = |N(u)|$ be the *degree* of $u \in V$. A vertex is a *universal* vertex if its degree is $n-1$. Let $\Delta = \max_{u \in V}\{d(u)\}$.

An $L(2,1)$-coloring of a graph $G = (V, E)$ is a function $f : V \to \mathbb{N}$ such that if $uv \in E$, then $|f(u) - f(v)| \geq 2$, moreover, if vertices u and v have distance two, then $f(u) \neq f(v)$. The maximum integer used in an $L(2,1)$-coloring f of a graph is the *span* of f and the minimum span among all $L(2,1)$-colorings of a graph G is denoted by $\lambda(G)$ [4].

A *total* $L(2,1)$*-coloring* f of a graph $G = (V, E)$ is a function $f : (V \cup E) \to \mathbb{N}$ such that if $uv \in E$, then: $|f(u) - f(v)| \geq 2$; $|f(uv) - f(v)| \geq 2$; and $|f(u) - f(uv)| \geq 2$; moreover, if uv and vw are in E, then: $f(uv) \neq f(vw)$, $f(u) \neq f(vw)$, $f(uv) \neq f(w)$, $f(u) \neq f(w)$. The minimum span among all total $L(2,1)$-colorings of a graph is denoted by $\lambda_T(G)$, or λ_T when no confusion arises. An *optimum total* $L(2,1)$*-coloring* is a total $L(2,1)$-coloring whose span is equal to $\lambda_T(G)$.

The *square* of G, denoted by G^2 has the same vertices and edges of G plus the edges between pairs of vertices having distance two in G. The *incident graph* H of G is obtained by subdividing every edge of G. The *total graph* of G is the square of the incident graph of G whereas the graph obtained by adding the edges of G to its incident graph, is denoted by HG. Havet and Yu [5] studied $(2,1)$-total colorings of a graph G which is equivalent to an $L(2,1)$-coloring of H. On the other hand, Duan et al. [2] considered $L(2,1)$-colorings of the total graph of G which is equivalent to an $L(2,1)$-coloring of H^2. A total $L(2,1)$-coloring of a graph G is equivalent to an $L(2,1)$-coloring of HG. It is straightforward to notice that H is a subgraph of HG, and HG is a subgraph of H^2, i.e., a total $L(2,1)$-coloring is settled between these two other assignments, as can be seen in Figure 1.

The TOTAL $L(2,1)$-COLORING PROBLEM is the problem of giving a graph G and an integer k, decide if there is a total $L(2,1)$-coloring of G with span at most k.

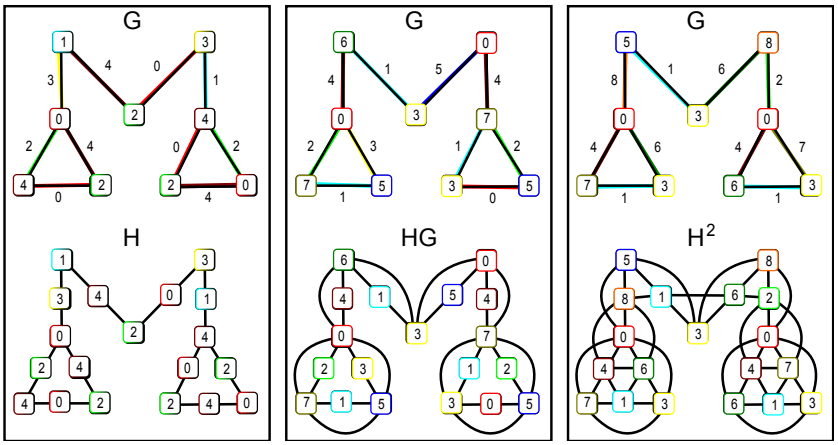


Figure 1: $(2,1)$-total, total $L(2,1)$, and $L(2,1)$-coloring of the total graph of G

Three vertices u, v, w form a *D2T* set if $d(u) = d(v) = d(w) = \Delta$ and the distance between each two of them is at most 2. The following straightforward lemma states a lower bound for λ_T.

Lemma 1.1. *For every graph, $\lambda_T \geq 2\Delta{+}1$. Moreover, if the graph has a D2T set, $\lambda_T \geq 2\Delta{+}2$.*

2 Optimum total $L(2,1)$-coloring

We give λ_T for some families of graphs. Due to lack of space we skip proofs of some small graphs.

A graph is a: (i) *path graph* (P_n) if $E(P_n) = \{v_i v_{i+1} \mid 1 \leq i \leq n-1\}$; (ii) *cycle graph* (C_n) if $E(C_n) = \{v_i v_{i+1} \mid 1 \leq i \leq n-1\} \cup \{v_1 v_n\}$; (iii) *star graph* (S_n) if it has a universal vertex and all others have degree 1; (iv) *wheel graph* (W_n) if it is obtained by adding a universal vertex to a graph C_{n-1}; (v) *complete graph* (K_n) if it has all possible edges between its vertices.

Theorem 2.1. *For a Path graph P_n, with $n \geq 4$, $\lambda_T(P_n) = 6$.*

Proof. In [1] it was shown that for a path of triangles $\lambda \leq 6$. It is possible to associate this result with a total $L(2,1)$-coloring of path graphs, using the top of the triangles labels on the edges of P_n. Hence $\lambda_T(P_n) \leq 6$. Moreover, by Lemma 1.1, $\lambda_T(P_n) \geq 2\Delta + 2 = 6$, $n \geq 5$. □

Theorem 2.2. *For a Cycle graph C_n, with $n \neq 5$, $\lambda_T(C_n) = 6$.*

Proof. By Lemma 1.1, $\lambda_T(C_n) \geq 2\Delta{+}2 = 6$. Let C_n be described as $(v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n, e_n)$. If $n \equiv 0 \mod 3$, when $n \geq 6$, label in a clockwise order: (i) $(6,1,3,5,0,2,6,1,3,5,0,2)$; (ii) add recursively $(6,1,3,5,0,2)$ for other vertices and edges. If $n \equiv 1 \mod 3$, when $n \geq 7$, label in a clockwise order: (i) $(3,5,0,2,6,4,1,3,5,2,0,4,6,1)$; (ii) add recursively $(3,5,0,2,6,1)$ for other vertices and edges. If $n \equiv 2 \mod 3$, when $n \geq 14$, label in a clockwise order: (i) $(3,5,0,2,6,4,1,3,5,2,0,4,6,1,3,5,0,2,6,4,1,3,5,2,0,4,6,1)$; (ii) add recursively $(3,5,0,2,6,1)$ for other vertices and edges. It is interesting to note that $\lambda_T(C_5) = 7$. □

Theorem 2.3. *For a Wheel graph ($n \geq 5$) and a Star graph ($n \geq 3$), $\lambda_T = 2\Delta + 1$.*

Proof. By Lemma 1.1, a wheel graph has $\lambda_T \geq 2\Delta{+}1$. Unless W_4 (a complete graph), this lower bound is tight. For W_n with $n \geq 7$, label $\{0, \ldots \Delta{+}1\}$ to its vertices cf. [4] (0 to the universal

vertex v and, in a clockwise order in the outer cycle, assign even labels in $\{2, \ldots, \Delta + 1\}$, and later the odd labels). Now, we can use labels in $\{\Delta+2, \ldots, 2\Delta+1\}$ to edges in $N_e(v)$. Finally, for an edge in the outer cycle, there are at most 10 forbidden labels and, there exists at least one available label in $\{0, \ldots, 2\Delta+1\}$. As S_n is subgraph of W_n, when $n \geq 5$, $\lambda_T(S_n) = 2\Delta+1$. □

Theorem 2.4. *For complete graphs, $\lambda_T(K_n) = 2n$ for even n, K_3 and K_5. Otherwise, $\lambda_T(K_n) = 2n + 1$ (unless $\lambda_T(K_1) = 0$).*

Proof. By Lemma 1.1, $\lambda_T(K_n) \geq 2\Delta + 2 = 2n$, when $n \geq 3$. This lower bound is tight for K_3, K_5, and K_n when n is even. Otherwise, we show that $\lambda_T(K_n) = 2n+1$. Assume $\lambda_T(K_n) = 2n$ when $n \geq 7$ is odd. There are $2n + 1$ labels in $\{0, 1, 2, \ldots, 2n\}$. Every vertex of K_n must receive a label that differ by at least two apart of each other. Thus, after an assignment of vertices of K_n, there exist at most 4 labels without its consecutive values assigned to vertices of the K_n. These 4 labels can cover at most $\frac{n-1}{2}$ edges each, and each of the others $n - 3$ labels can cover at most $\frac{n-3}{2}$ edges. As we are dealing with a complete graph, we need that $4\frac{(n-1)}{2} + (n-3)\frac{(n-3)}{2} \geq n\frac{(n-1)}{2}$. However, it requires that $n \leq 5$, a contradiction. It is interesting that, different from others odd complete graphs, $\lambda_T(K_n) = 2n$ for K_3 and K_5.

One can obtain an optimum total $L(2,1)$-coloring for odd complete graphs as follows: (i) use even labels in $\{0, \ldots, 2n-4\}$ to the vertices in a clockwise order, and label $2n$ to the last vertex; (ii) for each odd i in $\{1, \ldots, 2n-5\}$, label i to the $\frac{n-3}{2}$ parallel edges to the vertices that received labels $i-1$ and $i+1$; (iii) label $2n+1$ to the $\frac{n-1}{2}$ edges of the outer cycle in such way that they are not incident to the vertex with label $2n$; (iv) similar, label $2n-3$ to the others $\frac{n-1}{2}$ edges of the outer cycle; (v) label $2n-2$ to the remaining edge of the outer cycle and to all parallel edges of the vertices with label $2n-4$ and $2n$; (vi) label $2n-1$ to the remaining edges (the ones parallel to the vertices with labels 0 and $2n$). Thus, $\lambda_T(K_n) \leq 2n+1$ for odd values of n. Furthermore, label $2n+1$ is only used by edges of the complete graph, and every vertex have at least one more available label different from the other vertices. As a consequence, one can extend this assignment obtaining $\lambda_T(K_n) = 2n$ for even $n \geq 8$. □

A set of vertices of V is a *stable set* if there is no edges between them. A *complete q-partite graph* is a graph whose vertex set V can be partitioned into q stable sets $V_1, \ldots, V_q$ and has all the edges between vertices in different sets.

Let V_i be one of minimum size stable sets of a complete q-partite graph. Therefore, $\forall v \in V_i$, $d(v) = \Delta$). Every pair of vertices have distance two, and so receive different labels. There is a vertex v in V_i with label $f(v)$ such that $f(v) - 1$ or $f(v) + 1$ is not used in a vertex of V_i. As every vertex in $V(G)$ and every edge in $N_e(v)$ get different labels, there exist, at least, $2+n-1+\Delta = n+\Delta+1$ different labels, which implies a span at least $n+\Delta$. Then, for complete q-partite graphs, $\lambda_T \geq n + \Delta$. Besides, if there is no universal vertex, then $\lambda_T = n + \Delta$. The upper bound of λ_T is obtained using an optimum total coloring of complete q-partite graphs.

Theorem 2.5. *For complete q-partite graphs $n + \Delta \leq \lambda_T \leq n + \Delta + 2$. Moreover, If there is no universal vertex, then $\lambda_T = n + \Delta$.*

A *tree* is a connected and acyclic graph. One can obtain a rooted version of this tree. Then, label the root and its neighbourhood as a star graph. For the other vertices v, respecting its depths in the tree, when v receives its label, it has one vertex and one edge already labeled in its neighbourhood (the ones that precede it in the rooted tree). However, there are $2\Delta+1$ sufficient labels in $\{0, \ldots, 2\Delta + 2\}$ to v and its at most $2\Delta - 2$ unlabeled elements in $N(v)$ and $N_e(v)$.

Theorem 2.6. *For a tree, $2\Delta + 1 \leq \lambda_T \leq 2\Delta + 2$.*

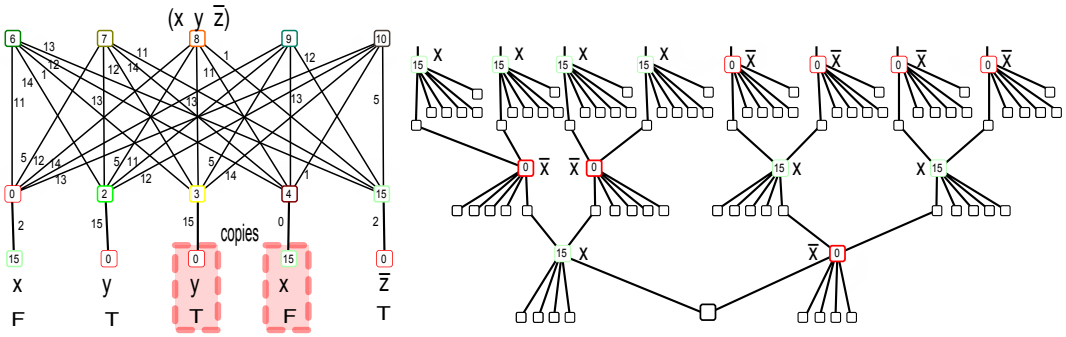


Figure 2: The graph G corresponding to a NAE 3-SAT instance

3 $\mathcal{NP}$-completeness

We give a sketch of the proof from the $\mathcal{NP}$-complete problem NAE 3-SAT [3]. Given an instance of NAE 3-SAT, construct a bipartite graph G with $\Delta = 7$ as follows: (i) for each literal $\mathbf{x}$ construct a tree that replicates the values of $\mathbf{x}$ and $\overline{\mathbf{x}}$ (e.g., if $\mathbf{x}$ is true, $f(\mathbf{x}) = 0$ and $f(\overline{\mathbf{x}}) = 15$) in vertices with degree Δ (unless the ones near the leaves which have $\Delta - 1$), two times the number of appearance of the literal; (ii) for each clause, construct a bipartite complete graph $K_{5,5}$; (iii) add edges between vertices of one part of each $K_{5,5}$'s to the vertices with degree $\Delta - 1$ that represent their literals (complete the two others with any of these literals).

Assume $\lambda_T(G) = 15$, by Lemma 1.1, vertices with degree Δ need labels 0 or 15. By Theorem 2.5, $\lambda_T(K_{5,5}) = n + \Delta = 15$. Additionally, there are vertices or edges of $K_{5,5}$'s receiving label 0 or 15 (otherwise $\lambda_T(K_{5,5}) = 14$). If all vertices near the leaves connected to a $K_{5,5}$ have the same label (0 or 15) (i.e., every literal in a clause have the same value) that label is not used in vertices or edges of the $K_{5,5}$, a contradiction. Conversely, if we assume the formula is true, we give a total $L(2,1)$-coloring of G with span 15 as follows: (i) each $K_{5,5}$ can be labeled as in Figure 2; (ii) vertices and edges of the tree can be labeled as in Theorem 2.6 (for a vertex v, there are two labeled vertices and two labeled edges in its neighbor, however, there exists an assignment of the remaining 11 labels to v and the unlabeled elements in $N(v)$ and $N_e(v)$).

Theorem 3.1. *The* TOTAL $L(2,1)$-COLORING PROBLEM *is $\mathcal{NP}$-complete even bipartite graphs.*

References

[1] Bonomo, F. and Cerioli, M. R., On $L(2,1)$-labeling of block graphs, *Int. J. Comput. Math.* **88** (2010), 468–475.

[2] Duan, Z., Lv, P., Miao, L., Miao, Z. and Wang, C., The Δ^2-conjecture for $L(2,1)$-labelings is true for total graphs, *Appl. Math. Letters*, **24** (2011), 1491–1494.

[3] Garey, M. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., (1979).

[4] Griggs, J. R. and Yeh, R. K., Labelling graphs with a condition at distance 2, *SIAM J. Disc. Math.*, **5** (1992), 586–595.

[5] Havet, F. and Yu, M.-L., $(p,1)$-total labelling of graphs, *Disc. Math.* **308** (2008), 496–513.

Testing Uniformity of Stationary Distribution

Sourav Chakraborty[1], Akshay Kamath[1], and Rameshwar Pratap[1]

[1]Chennai Mathematical Institute, Chennai, India

1 Introduction

Markov chains are one of the most important and most studied structures in Theoretical Computer Science. The most important characteristics of a Markov chain are its stationary distribution and its mixing time. In particular, one often wants to know if a given distribution is a stationary distribution of a given Markov chain. In this paper, we focus on the Markov chain obtained by a random walk on a directed graph. Stationary distribution of a Markov chain is a global property of the graph, hence whether a particular distribution is a stationary distribution of a Markov chain depends on the global structure of that Markov chain. We prove that contrary to normal perception, if the graph is regular then whether the uniform distribution on the vertices of the graph is a stationary distribution depends on a local property of the graph. The following theorem, which is the main result of this paper, is a statement about that local property. (See [3] for full version of this paper.)

Theorem 1.1. *If $\overrightarrow{G} = (V, \overrightarrow{E})$ is a digraph such that the total degree (that is $Indegree(v) + Outdegree(v)$) for every vertex $v \in V$ is the same, then the uniform distribution on the vertices of $\overrightarrow{G}$ is a stationary distribution (for the Markov chain generated by a random walk on $\overrightarrow{G}$) if and only if the graph have the following properties:*

1. *For all $v \in V$, $Indegree(v) \neq 0$ and $Outdegree(v) \neq 0$,*
2. *For every edge $(u, v) \in \overrightarrow{E}$, $Outdegree(u) = Indegree(v)$.*

2 Preliminaries

2.1 Graph Notations

Throughout this paper, we will be dealing with directed graphs (possibly with multiple edges between any two vertices) in which each edge is directed only in one direction. We will call them **oriented graphs**. We will denote the oriented graph by $\overrightarrow{G} = (V, \overrightarrow{E})$ and the underlying undirected graph (that is when the direction on the edges are removed) by $G = (V, E)$. For a vertex $v \in V$, the in-degree and the out-degree of v in $\overrightarrow{G}$ are denoted by $d^-(v)$ and $d^+(v)$ respectively. An oriented graph $\overrightarrow{G} = (V, \overrightarrow{E})$ is called a degree-Δ oriented graph if for all $v \in V$, $d^-(v) + d^+(v) = \Delta$. In this paper, we will be focusing on degree-Δ oriented graphs.

2.2 Markov Chains

A Markov chain is a stochastic process on a set of states given by a transition matrix. Let S be the set of states with $|S| = n$. Then, the transition matrix T is a $n \times n$ matrix with entries from positive real; the rows and columns are indexed by the states; the u, v-th entry $T_{u,v}$ of the matrix denotes the probability of transition from state u to state v. Since T is stochastic, $\sum_v T_{u,v}$ must be 1. A distribution $\mu : S \to \mathbb{R}^+$ on the vertices is said to be stationary if for all vertices v,

$$\sum_v \mu(u) T_{u,v} = \mu(v).$$

If $\overrightarrow{G}$ is an oriented graph then a random walk on $\overrightarrow{G}$ defines a Markov chain, where, the states are the vertices of the graph; the probability to traverse an edge is given by the quantity $p_{u,v} = \frac{1}{d^+(u)}$; and hence, the transition probability $T_{u,v}$ from vertex u to vertex v is $p_{u,v}$ times the number of edges between u and v. The uniform distribution on the vertices of $\overrightarrow{G}$ is a stationary distribution for this Markov chain if and only if for all $v \in V$,

$$\sum_{u:(u,v)\in\overrightarrow{E}} p_{u,v} = 1 = \sum_{w:(v,w)\in\overrightarrow{E}} p_{v,w}.$$

3 Structure of Graphs with Uniform Stationary Distribution

The following Theorem is a rephrasing of Theorem 1.1.

Theorem 3.1. *Let $\overrightarrow{G} = (V, \overrightarrow{E})$ be a degree-Δ oriented graph, then the uniform distribution on the vertices of $\overrightarrow{G}$ is a stationary distribution (for the Markov chain generated by a random walk on $\overrightarrow{G}$) if and only if for all $v \in V$, both $d^-(v), d^+(v) \neq 0$ and for all $(u, v) \in \overrightarrow{E}$,*

$$d^+(u) = d^-(v)$$

Proof. First of all, note that the uniform distribution is a stationary distribution for $\overrightarrow{G}$, iff for all $v \in V$

$$\sum_{u:(u,v)\in\overrightarrow{E}} p_{u,v} = 1 = \sum_{w:(v,w)\in\overrightarrow{E}} p_{v,w},$$

where $p_{u,v}$ is the transition probability (defined in subsection 2.2) from vertex u to vertex v.

Thus, if the graph $\overrightarrow{G}$ has the property that for all $(u, v) \in \overrightarrow{E}$, $d^+(u) = d^-(v)$, then note that

$$\sum_{u:(u,v)\in\overrightarrow{E}} p_{u,v} = \sum_{u:(u,v)\in\overrightarrow{E}} \frac{1}{d^+(u)} = \sum_{u:(u,v)\in\overrightarrow{E}} \frac{1}{d^-(v)} = 1,$$

the last equality holds because the summation is over all the edges entering v (which is non-empty) and thus have $d^-(v)$ number of items in the summation.

Similarly, we can also prove that $\sum_{w:(v,w)\in\overrightarrow{E}} p_{v,w} = 1$. Thus, we have proved this direction.

Now let us prove the other direction, that is, let us assume that the uniform distribution is a stationary distribution for the Markov chain. Note that, if the uniform distribution is a stationary distribution then there is a path from u to v if and only if u and v are in the same

strongly connected component of $\overrightarrow{G}$. This is because the uniform distribution is a stationary distribution if and only if for every cut $C = V_1 \cup V_2$ where $V_2 = (V \backslash V_1)$, we have

$$\sum_{(u,v)\in\overrightarrow{E}, \text{ and } u\in V_1, v\in V_2} p_{u,v} = \sum_{(u,v)\in\overrightarrow{E}, \text{ and } u\in V_2, v\in V_1} p_{u,v}.$$

In other words, if a stationary distribution is uniform then it implies that every connected component in the undirected graph is strongly connected in the directed graph.

Let $v_0, v_1, v_2, \ldots, v_t$ be a sequence of vertices such that the following conditions are satisfied. We call such a sequence as "degree-alternating" sequence of vertices.

- For all $i \geq 0$, $(v_{i+1}, v_i) \in \overrightarrow{E}$
- For all $i \geq 0$, $d^+(v_{2i+1}) = \min\left\{d^+(w) \ : \ (w, v_{2i}) \in \overrightarrow{E}\right\}$ and
- For all $i > 0$, $d^+(v_{2i}) = \max\left\{d^+(w) \ : \ (w, v_{2i-1}) \in \overrightarrow{E}\right\}$

Claim 3.2. *Let $\{v_i\}$ be a "degree-alternating" sequence of vertices. If we define a new sequence $\{S\}$ of positive integers as: for all $k \geq 0$, $s_{2k} = d^-(v_{2k})$ and $s_{2k+1} = d^+(v_{2k+1})$, then this sequence of positive integers is a non-increasing sequence. Moreover, if v_i and v_{i+1} are two consecutive vertices in the sequence such that $d^-(v_{i+1}) \neq d^+(v_i)$ then $s_{i+1} < s_i$.*

Using this claim, we would finish the proof of Theorem 3.1. Let there be one vertex $w \in V$ such that $d^+(u) \neq d^-(w)$ for some edge $(u, w) \in \overrightarrow{E}$. Let w' be the vertex such that $(w', w) \in \overrightarrow{E}$ and

$$d^+(w') = \min\{d^+(u) : (u, w) \in \overrightarrow{E}\}.$$

Since we have already argued that in the graph every connected component has to be strongly connected, we can create an infinite sequence of vertices such that w and w' appears consecutively and infinitely often. Now by Claim 3.2, it means that the sequence $\{S\}$ is a non-increasing sequence that decreases infinitely many times. But this cannot happen as all the numbers in the sequence $\{S\}$ represent in-degree or out-degree of vertices and hence, are always finite integers and can never be negative. Thus, if one vertex $w \in V$ such that $d^+(u) \neq d^-(w)$ for some edge $(u, w) \in \overrightarrow{E}$ then we hit a contradiction. Thus, for all edges $(u, v) \in \overrightarrow{E}$, $d^+(u) = d^-(v)$. □

Proof of Claim 3.2. This proof uses the fact that since the uniform distribution is a stationary distribution for the Markov chain, for all vertices v

$$\sum_{u:(u,v)\in\overrightarrow{E}} \frac{1}{d^+(u)} = 1 = \sum_{w:(v,w)\in\overrightarrow{E}} \frac{1}{d^+(v)}.$$

Let us first prove that in the sequence $\{S\}$, $s_{2i} \geq s_{2i+1}$.
Since $d^+(v_{2i+1}) = \min\left\{d^+(w) \ : \ (w, v_{2i}) \in \overrightarrow{E}\right\}$,

$$1 = \sum_{(u,v_{2i})\in\overrightarrow{E}} \frac{1}{d^+(u)} \leq \frac{d^-(v_{2i})}{d^+(v_{2i+1})}, \tag{1}$$

and hence, we have $d^-(v_{2i}) \geq d^+(v_{2i+1})$ which by definition gives $s_{2i} \geq s_{2i+1}$.

Now let us also prove that in the sequence $\{S\}$, $s_{2i-1} \geq s_{2i}$. By definition, this is same as proving $d^+(v_{2i-1}) \geq d^-(v_{2i})$. Since we have assumed that the graph is a degree-Δ graph, proving $s_{2i-1} \geq s_{2i}$ is same as proving $d^-(v_{2i-1}) \leq d^+(v_{2i})$. Now, using arguments smililar to previous case we can also prove that $s_{i+1} < s_i$. □

From Theorem 3.1 we can also obtain the following corollary. Both Theorem and Corollary has an application to property testing. We briefly present this application in next section.

Corollary 3.3. *Let $\overrightarrow{G} = (V, \overrightarrow{E})$ be a connected degree-Δ oriented graph. Then the uniform distribution of vertices is a stationary distribution for the random walk markov chain on $\overrightarrow{G}$, if and only if the following conditions apply:*

1. *If $G = (V, E)$ is non-bipartite, then the graph $\overrightarrow{G}$ is Eulerian.*
2. *If G is bipartite with bipartition $V_1 \cup V_2 = V$ then $|V_1| = |V_2|$ and in-degree of all vertices in one partition will be same and it will be equal to out-degree of all vertices in other partition.*

4 Application to Property Testing

In property testing, the goal is to look at a very small fraction of the input and distinguish whether the input has a certain property or it is "far" from satisfying the property. Here "far" means that one has to change at least ϵ fraction of the input to make the input satisfy the property. Theorem 1.1 also has an application to the problem of testing whether a given distribution is uniform or "far" from being uniform. More precisely, if the distribution is the stationary distribution of the random walk on a directed graph and the graph is given as an input, then how many bits of the input graph do one need to query in order to decide whether the distribution is uniform or "far" from it? We consider this problem in the **orientation model** [1] (see [2]). We reduced this problem to testing Eulerianity in the orientation model. And using result from [1] on query complexity of testing Eulerianity, we obtain bounds on the query complexity for testing whether the stationary distribution is uniform.

5 Conclusion

The result holds only for graphs where the in-degree plus out-degree of all the vertices are the same. It would be interesting to see if one can make a similar statement for general graphs.

References

[1] Eldar Fischer, Oded Lachish, Ilan Newman, Arie Matsliah, and Orly Yahalom. On the query complexity of testing orientations for being eulerian. In *APPROX-RANDOM*, pp. 402-415, 2008.

[2] Shirley Halevy, Oded Lachish, Ilan Newman, and Dekel Tsur. Testing properties of constraint-graphs. In *IEEE Conference on Computational Complexity*, pp. 264-277, 2007.

[3] Sourav Chakraborty, Akshay Kamath, and Rameshwar Pratap. Testing Uniformity of Stationary Distribution. *CoRR*, abs/1302.5366, 2013.

[1] In orientation model, underlying undirected graph is known in advance and one has to query the orientation of edges in order to test the property. The graph is said to be "ϵ-far" from satisfying the property if one has to reorient at least ϵ fraction of the edges to make the graph have the property.

Balanced Abelian group valued functions on directed graphs: Extended abstract*

Yonah Cherniavsky[1], Avraham Goldstein[2], and Vadim E. Levit[3]

[1]Department of Computer Science and Mathematics, Ariel University, Israel; yonahch@ariel.ac.il
[2]City University, New-York, USA; avraham.goldstein.nyc@gmail.com
[3]Department of Computer Science and Mathematics, Ariel University, Israel; levitv@ariel.ac.il

1 Introduction

Let A be an Abelian group with the group operation denoted by $+$ and the identity element denoted by 0. Let G be a graph. Roughly speaking, an A-valued function f on vertices and/or edges of G is called *balanced* if the sum of its values along any cycle of G is 0. Our cycles are not permitted to have repeating edges.

The study of balanced functions can be conducted in three cases:

1. The graph G is directed with the set of vertices V and the set of directed edges E. When traveling between the vertices, we are allowed to travel with or against the direction of the edges. The value of a function f on $\bar{e}$, which represents traveling the edge e against its direction, is equal to $-f(e)$. In this context, when the function is defined on edges only, the pair (G, f) is called a network or a directed network. In this paper we shall call this *the flexible case*, meaning that the direction of an edge does not forbid us to walk against it. The notion of balanced functions on edges for the flexible case, for functions taking values only on the edges, is introduced in the literature under different names. Thus, for example, in [1] the set of such functions is exactly $Im(d)$ and in [6], in somewhat different language, that set is referred-to as the set of consistent graphs. In a rather common terminology introduced by Zaslavsky, [8], a pair of graph and such a function on edges of a graph is a *gain graph*.

2. The graph G is directed with the set of vertices V and the set of directed edges E, but we are only allowed to travel with the direction of the edges. In this paper we shall call this *the rigid case*. When f takes values only on the edges then in some literature, following Sierre, the flexible case is described as a particular instance of the rigid case by introducing the set $\mathbb{E}$ as the new set of directed edges of G (the cardinality of $\mathbb{E}$ is twice that of E), denoting by $\bar{e} \in \mathbb{E}$ the inverse of the directed edge $\bar{e} \in \mathbb{E}$ and requiring $f(\bar{e}) = -f(e)$, [1], [7].

3. The graph G is undirected. The value of a function f on an edge e does not depend on the direction of the travel on e. This case was studied in [2], [5] and [3].

*For the full-text paper see [4].

2 Results

In what follows we say that a directed graph is weakly connected if its underlying undirected graph is connected.

2.1 The flexible case.

Let $G = (V, E)$ be a weakly connected directed graph, possibly with loops and multiple edges. Let $v, w \in V$ be two vertices connected by an edge e; v is the origin of e and w is the endpoint of e. For $e \in E$ denote by $\bar{e}$ the same edge as e but taken in the opposite direction. Thus $\bar{e}$ goes from w to v. Let $\mathbb{E} = \{e, \bar{e} \,|\, e \in E\}$. A path p from a vertex x to a vertex y is an alternating sequence $v_1, e_1, v_2, e_2, ..., v_n, e_n$ of vertices from V and edges from $\mathbb{E}$ such that $v_1 = x$ and each e_j, for $j = 1, ..., n-1$, goes from v_j to v_{j+1} and e_n goes from v_n to y. A cycle is a path from a vertex to itself. The length of a cycle is the number of its edges.

Definition 2.1. A function $f : \mathbb{E} \to A$ such that $f(\bar{e}) = -f(e)$ is called *balanced* if the sum $f(e_1) + ... + f(e_n)$ of the values of f over all the edges of any cycle of G is equal to 0.

Definition 2.2. The set of all the balanced functions $f : \mathbb{E} \to A$ is denoted by $\mathcal{HF}(\mathbb{E}, A)$. $\mathcal{HF}(\mathbb{E}, A)$ is a subgroup of the Abelian group $A^{\mathbb{E}}$ of all the functions from $\mathbb{E}$ to A.

Definition 2.3. A function $g : V \to A$ is called *balanceable* if exists some $f : \mathbb{E} \to A$ such that $f(\bar{e}) = -f(e)$ and the sum of all the values $g(v_1) + f(e_1) + g(v_2) + f(e_2) + ... + g(v_n) + f(e_n)$ along any cycle of G is zero. We say that this function $f : \mathbb{E} \to A$ balances the function $g : V \to A$.

Definition 2.4. The set of all the balanceable functions $g : V \to A$ is denoted by $\mathcal{BF}(V, A)$. The group $\mathcal{BF}(V, A)$ is a subgroup of the free Abelian group A^V of all the functions from V to A.

Definition 2.5. A function $h : V \bigcup \mathbb{E} \to A$, which takes both vertices and edges of G to some elements of A, is called balanced if $h(\bar{e}) = -h(e)$ and the sum of its values $h(v_1) + h(e_1) + h(v_2) + h(e_2) + ... + h(v_n) + h(e_n)$ along any cycle of G is zero.

Definition 2.6. The set of all the balanced functions $h : V \bigcup \mathbb{E} \to A$ is denoted by $\mathcal{WF}(G, A)$. The group $\mathcal{WF}(G, A)$ is a subgroup of the Abelian group $A^{V \bigcup \mathbb{E}}$ of all the functions from $V \bigcup \mathbb{E}$ to A.

Clearly, any balanced function $f \in \mathcal{HF}(\mathbb{E}, A)$ can be viewed as a balanced function from $V \bigcup \mathbb{E}$ to A which takes zero value on every vertex of G. Thus, we will regard $\mathcal{HF}(\mathbb{E}, A)$ as a subgroup of $\mathcal{WF}(V \bigcup \mathbb{E}, A)$.

Proposition 2.7. *The quotient $\mathcal{WF}(V \bigcup \mathbb{E}, A)/\mathcal{HF}(\mathbb{E}, A)$ is naturally isomorphic to $\mathcal{BF}(V, A)$.*

Definition 2.8. The subgroup of all elements of A of order 2 is denoted by A_2.

The following fact about the group $\mathcal{HF}(\mathbb{E}, A)$ is well known.

Proposition 2.9. *The group $\mathcal{HF}(\mathbb{E}, A)$ is isomorphic to $A^{|V|-1}$.*

Theorem 2.10. *Let $G = (V, E)$ be a weakly connected directed graph and G' be its underlying undirected graph. Then:*

1. *If G' is bipartite, then the group $\mathcal{WF}(V \bigcup \mathbb{E}, A)$ is isomorphic to $A^{|V|}$.*

2. *If G' is not bipartite, then $\mathcal{WF}(V \bigcup \mathbb{E}, A)$ is isomorphic to $A_2 \times A^{|V|-1}$.*

Notice that if the graph G' is bipartite, then the group of balanceable functions $\mathcal{BF}(V, A)$ is isomorphic to A and if G' is not bipartite, then the group of balanceable functions $\mathcal{BF}(V, A)$ is isomorphic to A_2 - the group of involutions of A.

2.2 The rigid case.

Let $G = (V, E)$ be a weakly connected directed graph. Recall that in this case we are allowed to walk only in the direction of an edge but not against it. It naturally changes the notion of a path and of a cycle in comparison with the flexible case. Similarly to the flexible case denote by $\mathcal{BR}(V, A)$, $\mathcal{HR}(E, A)$ and $\mathcal{WR}(V \bigcup E, A)$ the groups of balanceable functions on vertices, balanced functions on edges and balanced labelings of the graph G respectively.

Proposition 2.11. *Any function on V is balanceable, i.e. $\mathcal{BR}(V, A) = A^V$.*

Definition 2.12. Two vertices x and y of G are called strongly connected if exists a path P_1 from x to y and a path P_2 from y to x.

Strong connectivity defines an equivalence relation on the vertices of G. The equivalence classes of strongly connected vertices, together with all the edges between the vertices in each class, are called the strongly connected components of G. We denote the number of strongly connected components of G by $\bar{k}(G)$.

Lemma 2.13. *If $\bar{k}(G) = 1$, then the group $\mathcal{HR}(E, A)$ is isomorphic to $A^{|V|-1}$ just like in the flexible case.*

Theorem 2.14. *The group $\mathcal{HR}(E, A)$ is isomorphic to $A^{|V|-\bar{k}(G)+r(G)}$, where $r(G)$ is the number of all the edges in G which go from a vertex in one strongly connected component of G to a vertex in another strongly connected component of G.*

Theorem 2.15. *$\mathcal{WR}(V \bigcup E, A)$ is isomorphic to $A^{|V|} \times A^{|V|-\bar{k}(G)+r(G)}$*

We finish with following simple claim, which connects this work to [3].

Proposition 2.16. *Let G be an undirected connected graph and let G_{dir} be a directed graph obtained from G by any assigning of directions to the edges of G. Denote by $H(E, A)$ the group of A-valued balanced functions on edges of G. Choose any order on edges of G and embed $H(E, A)$ and $\mathcal{HR}(E(G_{dir}), A)$ into $A^{|E|}$. For an undirected graph G the group of balanced functions on edges of G is equal to the intersection of all the groups $\mathcal{HR}(E(G_{dir}), A)$, where G_{dir} runs over all directed graphs for all $2^{|E|}$ possible direction assignments to the edges of G. The same is true for the groups of balanced functions on the entire graph (both vertices and edges). Namely, $W(V \bigcup E, A) = \bigcap \mathcal{WR}(V \bigcup E(G_{dir}), A)$.*

References

[1] Roland Bacher, Pierre de la Harpe, Tatiana Nagnibeda. The lattice of integral flows and the lattice of integral cuts on a finite graph. Bulletin del la S.M.F., tome 125, number 2 (1997).

[2] R. Balakrishnan and N. Sudharsanam. Cycle vanishing edge valuations of a graph. Indian J. Pure Appl. Math. 13 (3) (1982), 313 – 316.

[3] Y. Cherniavsky, A. Goldstein and V. E. Levit. On the structure of the group of balanced labelings on graphs. Preprint available at http://arxiv.org/abs/1301.4206.

[4] Y. Cherniavsky, A. Goldstein and V. E. Levit. Balanced Abelian group valued functions on directed graphs. Preprint available at http://arxiv.org/abs/1303.5456.

[5] Manas Joglekar, Nisarg Shah, Ajit A. Diwan. Balanced group-labeled graphs. Discrete Mathematics 312 (2012), 1542 – 1549.

[6] Martin Kreissig, Bin Yang. Efficient Synthesis of Consistent Graphs. EURASIP, 2010, ISSN 2076 – 1465, 1364 – 1368.

[7] Sierre, Jean-Pierre. Arbres, amalgames, SL_2 (1977; Trees).

[8] T. Zaslavsky. A mathematical bibliography of signed and gain graphs and allied areas. Electron. J. Combin. 5, (1998). Dynamic Surveys in Combinatorics, No. DS8 (electronic).

An oriented 8-coloring for acyclic oriented graphs with maximum degree 3

Hebert Coelho[1], Luerbio Faria[2], Sylvain Gravier[3], and Sulamita Klein[4]

[1]INF/UFG, Goiás, Brazil and COPPE/Sistemas - UFRJ, Rio de Janeiro, Brazil
[2]DCC/UERJ, Rio de Janeiro, Brazil.
[3]Institut Fourier, Maths à Modeler team, CNRS - UJF, St Martin d'Hères, France.
[4]Instituto de Matemática and COPPE/Sistemas - UFRJ, Rio de Janeiro, Brazil

1 Introduction

The oriented coloring was introduced independently by Courcelle [2] and Raspaud and Sopena [5]. Let $\vec{G}$ be an oriented graph, $xy, zt \in A(\vec{G})$ and $C = \{1, 2, \ldots, k\}$ be a set of colors. An *oriented k-coloring* of $\vec{G}$ is a function $\phi : V(\vec{G}) \to C$, such that $\phi(x) \neq \phi(y)$, and if $\phi(x) = \phi(t)$, then $\phi(y) \neq \phi(z)$. The *oriented chromatic number* $\chi_o(\vec{G})$ is the smallest k such that $\vec{G}$ admits an oriented k-coloring. Let $\vec{G}_1$ and $\vec{G}_2$ be two oriented graphs, a *homomorphism* of $\vec{G}_1$ to $\vec{G}_2$ is a mapping $f : V(\vec{G}_1) \to V(\vec{G}_2)$ such that $f(u)f(v) \in A(\vec{G}_2)$ whenever $uv \in A(\vec{G}_1)$. Clearly, $\vec{G}$ has an oriented k-coloring if and only if there is a tournament $\vec{T}$ on k vertices, such that $\vec{G}$ has a homomorphism to $\vec{T}$. If $\vec{G}$ has a homomorphism to $\vec{T}$, then $\vec{G}$ is *$\vec{T}$-colorable*.

We denote the *minimum* and *maximum indegrees*, and the *minimum* and *maximum outdegrees* of $\vec{G}$ respectively by $\delta^-(\vec{G})$, $\Delta^-(\vec{G})$, $\delta^+(\vec{G})$ and $\Delta^+(\vec{G})$. We consider also $\delta(\vec{G}) = \delta(G) = \delta$ and $\Delta(\vec{G}) = \Delta(G) = \Delta$, where G is the underlying graph of $\vec{G}$.

Oriented coloring has been studied by many authors. A survey on oriented coloring can be seen in [7]. Given an oriented graph $\vec{G}$ and a positive integer k, the ORIENTED COLORING PROBLEM (OCN_k) consists of determining whether there exists an oriented k-coloring of $\vec{G}$. By its strong appeal, OCN_k complexity has been exhaustively studied. In 2006, Culus and Demange [3] presented two NP-complete results from 3-SAT: that OCN_4 is NP-complete on acyclic oriented graphs with $\Delta = \max(p + 3; 6)$, and that OCN_4 is NP-complete on bipartite oriented graphs with $\Delta = \max(p+3; 7)$, where p denotes the maximum number of occurrences of a literal. Most recently, in 2010, Ganian and Hliněný [4] got an improvement in the Culus and Demange acyclic result proving that OCN_4 is NP-complete for connected acyclic oriented graphs with $\Delta = \max(p + 2; 4)$. We proved that OCN_4 is NP-complete [1] even when $\vec{G}$ is connected, planar, bipartite, acyclic oriented graph with $\Delta \leq 3$.

In this paper we are concerned with upper bounds and algorithms for determining oriented coloring of oriented graphs with $\Delta \leq 3$. Sopena and Vignal [8], exhibited a proof that if $\vec{G}$ is an oriented graph with $\Delta \leq 3$, then $\chi_o(\vec{G}) \leq 11$. In 1997 Sopena [6] posed a conjecture that: if $\vec{G}$ is an oriented graph such that $\Delta \leq 3$ and G is connected, then $\chi_o(\vec{G}) \leq 7$. In this work, we prove that there is a tournament $\vec{R}$ with 8 vertices such that if $\vec{G}$ is an acyclic oriented graph with $\Delta \leq 3$, then $\vec{G}$ is $\vec{R}$-colorable. Additionally, we provide a polynomial time algorithm to compute an R-coloring for $\vec{G}$.

2 The Color Digraph $\vec{R}$

Let $\vec{G}$ be an oriented graph with $\Delta \leq 3$. In this section, we show an oriented graph $\vec{R}$ on 8 vertices such that $\vec{G}$ admits an $\vec{R}-$coloring. Let p be a prime power such that $p \equiv 3(\text{mod } 4)$, the *Paley tournament* $Q\vec{R}_p$ is an oriented graph with $V(Q\vec{R}_p) = \{0, 1, \ldots, p-1\}$ and such that $xy \in A(Q\vec{R}_p)$ if and only if $y - x$ is a non-zero quadratic residue of p. In this work we are concerned only with $Q\vec{R}_7$, see Figure 1(a), that is the tournament with vertex set $V(Q\vec{R}_7) = \{0, 1, \ldots, 6\}$ and such that $xy \in A(Q\vec{R}_7)$ if an only if $y - x \equiv 1, 2$ or 4 (mod 7).

Observe that the transitive tournament $\vec{T}_4$, see Figure 1(c), is an oriented acyclic graph with $\Delta(\vec{T}_4) = 3$, which is not $Q\vec{R}_7$-colorable. Note that each vertex $v \in V(Q\vec{R}_7)$ has 3 successors, however the three successors of v always form a directed cycle, the same holds with the predecessors of v. Hence, there is no homomorphism from $\vec{T}_4$ to $Q\vec{R}_7$, since the 3 successors of the vertex $a \in V(\vec{T}_4)$ defines an acyclic graph. We build the graph $\vec{R}$ from $Q\vec{R}_7$ by the addition of vertex s, and the set of arcs $\vec{S} = \{sv : v \in V(Q\vec{R}_7)\}$. We consider the colors $\{0, 1, 2, 3, 4, 5, 6\}$ assigned to the vertices of $Q\vec{R}_7$ and the color 7 assigned to vertex s.

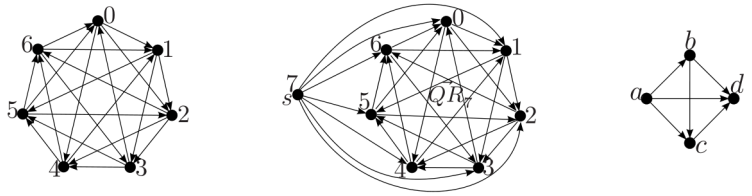


Figure 1: (a) Paley tournament $Q\vec{R}_7$, (b) Oriented graph $\vec{R}$, (c) transitive tournament $\vec{T}_4$.

3 $\vec{R}$-coloring of an acyclic oriented graph $\vec{G}$ with $\Delta \leq 3$

Lemma 3.1. *If $Q\vec{R}_7$ is the Paley tournament on 7 vertices, then for each arc $xy \in A(Q\vec{R}_7)$ there are $a, b, c, d \in \{0, 1, 2, 3, 4, 5, 6\}$ such that ax, ay, xb, by, cx, yc, $xd, yd \in A(Q\vec{R}_7)$.*

It is proved in [6] that $Q\vec{R}_7$ is the smallest graph satisfying the property of Lemma 3.1.

Let $\vec{G}$ be an oriented graph, $u, v, w \in V(\vec{G})$. Let ϕ be an $\vec{R}$-coloring for $\vec{G} \setminus \{w\}$, where u and v take colors $\phi(u), \phi(v)$ in $\{0, 1, 2, 3, 4, 5, 6\}$. The *triple* $(w, \phi(u), \phi(v)) = k$ is the smallest color k in $\{0, 1, 2, 3, 4, 5, 6\}$, such that k has the same adjacency relation with $\phi(u)$ and $\phi(v)$ in $Q\vec{R}_7$, as w has with u and v in $\vec{G}$. Please follow Figure 1(a). For example, if $wu, vw \in A(\vec{G})$, $\phi(u) = 0$ and $\phi(v) = 1$, then *triple* $(w, \phi(u), \phi(v)) = 3$. Note that the colors inciding in 0 are $3, 5, 6$, while the colors incided by 1 are $2, 3$ and 5.

Lemma 3.2. *Let ϕ be an $\vec{R}$-coloring of an acyclic oriented graph $\vec{G}$ such that $\Delta \leq 3$. If there is a vertex $x \in V(\vec{G})$, such that $deg^+_{\vec{G}}(x) = 3$ and $\phi(x) \in \{0, 1, 2, 3, 4, 5, 6\}$, then a new $\vec{R}$-coloring ϕ' can be defined from ϕ by replacing color $\phi(x)$ of x by the color $\phi'(x) = 7$.*

Proof. See that vertex s is a predecessor of all vertices in $(V(\vec{R}) \setminus s) \subseteq V(Q\vec{R}_7)$. As $deg^+_{\vec{G}}(x) = 3$, there is no conflict if we replace $\phi(x)$ by the color $\phi'(x) = 7$. □

Lemma 3.3. *Let ϕ be an $\vec{R}$-coloring of an acyclic oriented graph $\vec{G}$ with $\Delta \leq 3$. If there is a vertex $x \in V(\vec{G})$, such that $deg_G(x) \leq 2$ and $\phi(x) = 7$, then a new $\vec{R}$-coloring ϕ' can be defined from ϕ by replacing color $\phi(x)$ of x by one color $\phi'(x)$ in $\{0, 1, 2, 3, 4, 5, 6\}$.*

Proof. If $\phi(x) = 7$ and $deg_G(x) \leq 2$, then x is a source vertex. If $deg_G(x) = 0$, then any color in $\{0, 1, 2, 3, 4, 5, 6\}$ can be assigned to x. Suppose $1 \leq deg_G(x) \leq 2$. Let $N_G(x) = \{u_1, u_{deg_G(x)}\}$. By Lemma 3.1, color: *triple* $\big(x, \phi(u_1), \phi(u_{deg_G(x)})\big)$ in $\{0, 1, 2, 3, 4, 5, 6\}$ can be assigned to x. □

Theorem 3.4. *If $\vec{G}$ is an acyclic oriented graph, $\Delta \leq 3$, then there is an $\vec{R}$-coloring for $\vec{G}$.*

The proof is done by induction on $n = |V|$ in several cases. In order to illustrate the technique we consider the case in Figure 2 where there is a source s of $\vec{G}$ adjacent to a vertex h, $N_G(h) = \{s, c, d\}$ and $dh, hc \in A(\vec{G})$.

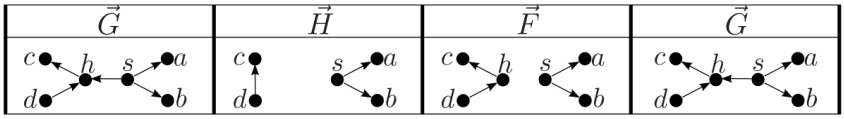


Figure 2: One case for $deg_{\vec{G}}(s) = 3$ and $deg_{\vec{G}}(h) = 3$.

In this case, we construct $\vec{H}$ from $\vec{G}$ by the removal of vertex h from $\vec{G}$ and addition of the arc dc to $A(\vec{H})$. Note that the arc dc does not create a cycle in $\vec{H}$. By induction hypothesis we can assign an $\vec{R}$-coloring $\phi_{\vec{H}}$ for $\vec{H}$. As $dc \in A(\vec{H})$, then $\phi_{\vec{H}}(c) \neq \phi_{\vec{H}}(d)$. Now we use $\phi_{\vec{H}}$, and Lemmas 3.1, 3.2 and 3.3 in order to define an $\vec{R}$-coloring $\phi_{\vec{G}}$ to $\vec{G}$ into 2 steps, see the oriented graph $\vec{F}$ in Figure 2. □

The structure of the proof of Theorem 3.4 yields a polynomial time method which allows to device a polynomial time algorithm to color an acyclic oriented graph with $\Delta \leq 3$.

Next, we describe the procedure $color(\vec{G_i})$ yielding an oriented 8-coloring from an acyclic oriented graph $\vec{G} = (V, A)$, $\Delta \leq 3$ and $n = |V|$ vertices. The execution starts with the call: $color(\vec{G_n})$, where $\vec{G_n} = \vec{G}$.

```
Procedure color(G_i);
  begin
      if (|V(G_i)| > 1) then
         Find source s_i in G_i;
         if (deg_{G_i}(s_i) = 3) then
            {Let N_{G_i}(s_i) = {h_i, a_i, b_i}};
            if (deg_{G_i}(h_i) = 3) then
               {Let N_{G_i}(h_i) = {s_i, c_i, d_i}};
               G_{i-1} ← G_i \ h_i;
               if (c_{i-1}d_{i-1} ∉ A_{i-1} and d_{i-1}c_{i-1} ∉ A_{i-1}) then
                  if (c_{i-1}h_{i-1} ∈ A_{i-1} and h_{i-1}d_{i-1} ∈ A_{i-1}) then
                     E_{i-1} ← E_{i-1} ∪ {c_{i-1}d_{i-1}};
                  if (d_{i-1}h_{i-1} ∈ A_{i-1} and h_{i-1}c_{i-1} ∈ A_{i-1}) then
                     E_{i-1} ← E_{i-1} ∪ {d_{i-1}c_{i-1}};
            else G_{i-1} ← G_i \ s_i;
         if (deg_{G_i}(s_i) = 2) then
            N_{G_i}(s_i) = {a_i, b_i};
            G_{i-1} ← G_i \ s_i;
         if (deg_{G_i}(s_i) = 1) then
            N_{G_i}(s_i) = {a_i};
            G_{i-1} ← G_i \ s_i;
         color(G_{i-1});
         if (deg_{G_i}(s_i) = 3) then
```

```
            if (deg_{G_i}(h_i) = 3) then
                φ(h_i) := triple(h(i), φ(c_i), φ(d_i));
                φ(s_i) := 7;
            else φ(s_i) := 7;
        if (deg_{G_i}(s_i) = 2) then
            φ(s_i) := triple(s_i, φ(a_i), φ(b_i));
        if (deg_{G_i}(s_i) = 1) then
            φ(s_i) := triple(s_i, φ(a_i), φ(a_i));
    else φ(v) := 0;
end;
```

4 Conclusion

Up to the moment, we have verified, using a brute force algorithm, that if $\vec{G}$ is a cubic graph with $|V(\vec{G})| \leq 12$, then $\chi_o(\vec{G}) \leq 7$. Our computational result is a positive evidence to support the conjecture of Sopena [6].

In this work, we prove that there is a color graph $\vec{R}$ on 8 vertices such that every acyclic oriented graph $\vec{G}$ with $\Delta(\vec{G}) \leq 3$ is $\vec{R}$-colorable. Additionally, we provide a polynomial time algorithm to compute an oriented 8-coloring for $\vec{G}$.

References

[1] H. Coelho, L. Faria, S. Gravier, and S. Klein. Oriented coloring in planar, bipartite, bounded degree 3 acyclic oriented graphs. *ENDM. To appear on proceedings of the VII Latin-American Algorithms, Graphs, and Optimization Symposium LAGOS, 2013.*

[2] B. Courcelle. The monadic second order logic of graphs vi: On several representations of graphs by relational structures. *Discrete Applied Mathematics*, 54:117–149, 1994.

[3] J. Culus and M. Demange. Oriented coloring: Complexity and approximation. *32rd Conference on Current Trends in Theory and Practice of Computer Science - LNCS 3831*, pages 226–236, 2006.

[4] R. Ganian and P. Hliněný. New results on the complexity of oriented colouring on restricted digraph classes. *SOFSEM'10, LNCS*, 5901:428–439, 2010.

[5] A. Raspaud and E. Sopena. Good and semi-strong colorings of oriented planar graphs. *Information Processing Letters*, 51:171–174, 1994.

[6] E. Sopena. The chromatic number of oriented graphs. *Journal of Graph Theory*, 25:191–205, 1997.

[7] E. Sopena. Oriented graph coloring. *Discrete Mathematics*, 229:359–369, 2001.

[8] E. Sopena and L. Vignal. A note on the oriented chromatic number of graphs with maximum degree three. Technical report, Université Bordeaux I, 1996.

Bound-optimal cutting planes

Stefano Coniglio*[1]

[1]Lehrstuhl II für Mathematik, RWTH Aachen University, Aachen

We propose a new paradigm for cutting plane generation for Mixed Integer Programming which allows for the simultaneous generation of k cuts which, when added to the current linear programming relaxation, yield the largest bound improvement. By Linear Programming duality arguments and simple linearizations we show that, for a large family of cutting planes, our cut generation problem is a Mixed 0-1 Integer Program. We present preliminary computational experiments on the generation of bound-optimal stable set inequalities for the max clique problem, which suggest the potential of the idea.

1 Introduction

Consider the following Mixed Integer Program (MIP) over n variables, p of which integer constrained, $P := \max\{cx : Ax \leq b, x \in \mathbb{Z}^p \times \mathbb{Q}^{n-p}\}$. Let $P_{LP} := \max\{cx : Ax \leq b, x \in \mathbb{Q}^n\}$ be its Linear Programming (LP) relaxation and let x^* be (one of) its optimal solution(s). Let us focus on a single family of valid inequalities $\alpha x \leq \alpha_0$ with coefficients (α, α_0) belonging to some set $\mathcal{C}$. It is common practice in cutting plane methods for mixed integer programming to generate, at each iteration, a violated inequality (a cut) which maximizes the cut violation $\alpha x^* - \alpha_0$ at x^*. For any detail, see [1] and the references therein.

In contrast to the fact that the generation of a cut with a *strictly positive* violation is a necessary condition for the convergence of the method, the generation of a *maximally violated* cut is not mandatory, although commonly considered the standard procedure.

In our view, a simple analogy with the simplex method suffices to suggest that alternatives to the cut violation are worth being explored. Assume that we are solving P_{LP} tightened with all the valid inequalities in $\mathcal{C}$, namely, $P^*_{LP} := \max\{cx : Ax \leq b, \alpha x \leq \alpha_0 \; \forall(\alpha, \alpha_0) \in \mathcal{C}, x \in \mathbb{Q}^n\}$ with the dual simplex method. Also assume that, at the current iteration, all the inequalities $\alpha x \leq \alpha_0$ are still violated (that is, the corresponding slack variables, when bringing the problem to standard form, are strictly negative). Adding a maximally violated cutting plane among those in $\mathcal{C}$ and reoptimizing thus amounts to pivoting on a row with the most negative slack or, looking at the dual, pivoting on the column with most negative reduced cost. Computational experience suggests that such pivoting rule, which sometimes goes by the name of *Dantzig's rule*, is in practice a poor choice, performing as poorly as pivoting on a random column with a negative reduced cost [2].

*Work partially funded by the Google Focused Grant Program on Mathematical Optimization and Combinatorial Optimization in Europe – "Robust Mixed Integer Programming".

Inspiration for alternatives to the cut violation can be drawn from the best pivoting rules usually adopted in the simplex method. Many of such rules, with steepest edge pricing [3] as a prominent example, aim at providing a better approximation of the *actual* variation in objective function value that is obtained after pivoting on a certain row or column, which is in general poorly approximated by the reduced cost[1]. Differently from the case of the simplex method, where the whole formulation is readily available and, in principle, the "best" pivot can be found (even though inefficiently) by tentatively performing all the feasible pivots and then backtracking and choosing the best one, in a cutting plane setting the cuts are not available *a priori* and hence such a naive approach is not applicable.

2 Bound-optimal cutting planes

Differently from the standard separation problem, in this paper we address the following:

Definition 2.1. *[Bound-optimal cutting plane generation] Given* $P := \max\{cx : Ax \leq b, x \in \mathbb{Z}^p \times Q^{n-p}\}$*, where* $Ax \leq b$ *contains lower and upper bounds for all the components of* x*, and its continuous relaxation* $P_{LP} = \max\{cx : Ax \leq b, x \in \mathbb{Q}^n\}$ *with optimal value* z_{LP}*, find a cutting plane* $\alpha x \leq \alpha_0$ *with* $(\alpha, \alpha_0) \in \mathcal{C}$ *which maximizes the bound variation* $z_{LP} - z'_{LP}$*, where* z'_{LP} *is the optimal value of* $P'_{LP} := \max\{cx : Ax \leq b, \alpha x \leq \alpha_0, x \in \mathbb{Q}^n\}$*.*

We show the following:

Theorem 2.2. *Assuming that* $\mathcal{C}$ *can be represented as a Mixed Integer polyhedron, a bound-optimal cutting plane as in Definition 2.1 is found by solving the following Quadratically Constrained Mixed Integer Program:*

$$\begin{aligned}
\min \quad & \textstyle\sum_{j=1}^{n} c_j x_j && && (1)\\
s.t. \quad & \textstyle\sum_{j=1}^{n} a_{ij} x_j \leq b_i && \forall i = 1, \ldots, m && (2)\\
& \textstyle\sum_{j=1}^{n} \alpha_j x_j \leq \alpha_0 && && (3)\\
& x_j \geq 0 && \forall j = 1, \ldots, n && (4)\\
& \textstyle\sum_{i=1}^{m} a_{ij} y_i + \alpha_j y_{m+1} \geq c_j && \forall j = 1, \ldots, n && (5)\\
& y_i \geq 0 && \forall i = 1, \ldots, m+1 && (6)\\
& \textstyle\sum_{j=1}^{n} c_j x_j = \sum_{i=1}^{m} b_i y_i + \alpha_0 y_{m+1} && && (7)\\
& (\alpha, \alpha_0) \in \mathcal{C}. && && (8)
\end{aligned}$$

Proof. By (2)–(4), P'_{LP} is rewritten as a parametric LP of variable $x \in \mathbb{R}^n$ depending on the variable coefficients of the new cut $\alpha x \leq \alpha_0$, the validity of which is imposed in (8). The optimality of x for P'_{LP} is enforced by introducing the parametric dual of P'_{LP} of variable $y \in \mathbb{R}^{m+1}$ in (5)–(6), where y_{m+1} is the dual variable of the new inequality in (3), and by imposing strong LP duality in (7). Since, for any feasible $(\alpha, \alpha_0) \in \mathcal{C}$, every x which is feasible for (2)–(7) is optimal for P'_{LP}, a bound-optimal cut is found by minimizing the objective function of P'_{LP} (rather than by maximizing it). □

Problem (1)–(8), which is of polynomial size w.r.t. m and n, is nonlinear because of the bilinear products $\alpha_j x_j$ in (3), $\alpha_j y_{m+1}$ in (5), and $\alpha_0 y_{m+1}$ in (7). Two results directly follow:

[1] Even though, clearly, finding the best neighboring solution within a pivoting operation has no relationship with the actual number of pivots needed to converge to an optimal solution.

Corollary 2.3. *If $(\alpha, \alpha_0) \in \mathbb{Z}^{n+1}$ and α_j, α_0 are upper bounded by some $M > 0$ for all $j = 1, \ldots, n$, then problem (1)-(8) can be cast as a Mixed 0-1 Integer Program of pseudopolynomial size w.r.t. m, n, M.*

Proof. It suffices to rewrite each variable α_j, for all $j = 1, \ldots, n$, and α_0 as the sum of at most M 0-1 variables, thus transforming all the bilinear products between an integer and a continuous variable in (3), (5), (7) into the sum of M bilinear products between a binary and a continuous variable, each of which can then be rewritten in a linear fashion by employing the McCormick envelope, which is tight if one of the variables in the product is binary. □

Corollary 2.4. *If $\alpha \in \{0,1\}^n$ for all $j = 1, \ldots, n$ and α_0 is an affine function of α, then problem (1)–(8) can be cast as a Mixed 0-1 Integer Program of polynomial size in m, n.*

Proof. If $\alpha_j \in \{0,1\}$ for every $j = 1, \ldots, n$, we can directly apply the McCormick envelope without transforming α_j into the sum of 0-1 variables. If $\alpha_0 = \sum_{j=1}^{n} w_j \alpha_j + w_0$ for some $w_0, \ldots, w_n \in \mathbb{R}$, the product $\alpha_0 y_{m+1}$ in (7) becomes $\sum_{j=1}^{n} w_j \alpha_j y_{m+1} + w_0 y_{m+1}$ and we can simply apply the envelope to each product $w_j \alpha_j y_{m+1}$. □

This second result is of practical interest, since it encompasses many families of combinatorial cutting planes, such as stable set inequalities (where $\alpha_0 = 1$), cut-set inequalities (where $\alpha_0 = 1$ for connectivity or $\alpha_0 = 2$ for biconnectivity), and cover inequalities (where $\alpha_0 = \sum_{j=1}^{n} \alpha_j - 1$).

Our bound-optimal cutting plane paradigm has two major differences w.r.t. the standard one: 1) it does not require to reoptimize the LP relaxation, which is automatically reoptimized each time Problem (1)–(8) is solved and 2) it does not directly involve the separation of an infeasible solution x^*. Note that, without imposing a strictly positive cut violation w.r.t. x^*, the method *may not* converge. This is the case of problems where the LP relaxation admits an optimal face but none of the cuts in $\mathcal{C}$ can, by itself, cut the entire face away.

Although we do not show it here for space reasons, Problem (1)–(8) can be easily extended so as to generate *any* number k of cutting planes which *simultaneously* yield the maximum bound variation. The impact of adopting $k > 1$ is shown in the following.

3 Bound-optimal stable set inequalities

Let us consider the max clique problem for an undirected graph $G = (V, E)$. Let $\mathcal{S} := S_1, \ldots, S_m$ be a collection of m stable sets containing at least all those of cardinality two. The LP relaxation of the problem reads $\max_{x \geq 0}\{\sum_{j \in V} x_j : \sum_{j \in S_i} x_j \leq 1 \ \forall S_i \in \mathcal{S}\}$. The "standard" separation problem is $\min_{\alpha \in \{0,1\}^n}\{\sum_{j \in V} x_j^* \alpha_j : \alpha_i + \alpha_j \leq 1 \ \forall (i,j) \in E\}$. Applying Theorem 2.2 and Corollary 2.4, a bound-optimal stable set inequality is obtained as a solution to the following Mixed 0-1 IP:

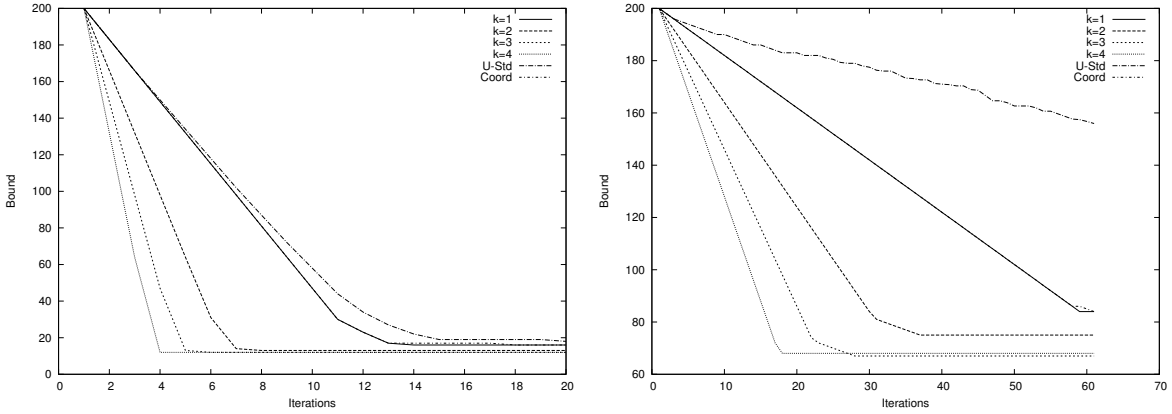
$$\begin{aligned}
\min \quad & \textstyle\sum_{j \in V} x_j & & (9)\\
s.t. \quad & \textstyle\sum_{j \in S_i} x_j \leq 1 & \forall i = 1, \ldots, m & (10)\\
& \textstyle\sum_{j \in V} \alpha_j x_j \leq 1 & & (11)\\
& \textstyle\sum_{i=1:S_i \ni j}^{m} y_i + \alpha_j y_{m+1} \geq 1 & \forall j \in V & (12)\\
& \textstyle\sum_{j \in V} x_j = \sum_{i=1}^{m} y_i + y_{m+1} & & (13)\\
& \alpha_i + \alpha_j \leq 1 & \forall (i,j) \in E & (14)\\
& \alpha_j \in \{0,1\}^n, x_j \geq 0 & \forall j = 1, \ldots, n & (15)\\
& y_i \geq 0 & \forall i = 1, \ldots, m+1, & (16)
\end{aligned}$$

which is linearized, for each $j = 1, \ldots, n$, by substituting two new variables $z_j, h_j \geq 0$ for, respectively, $\alpha_j x_j$ and $\alpha_j y_{m+1}$, and then introducing the McCormick envelope constraints: $z_j \leq x_j - \alpha_j + 1$, $z_j \geq x_j + \alpha_j - 1$, $z_j \leq \alpha_j$, $h_j \leq y_{m+1} - \alpha_j + 1$, $h_j \geq y_{m+1} + \alpha_j - 1$, $h_j \leq \alpha_j$.

4 Preliminary computational experiments

We illustrate preliminary results obtained on two of the max clique instances used in [1]: `c-fat200-1` and `c-fat200-5`. We generate bound-optimal stable set inequalities by solving (1)–(8) and its extension (which we only mentioned) which allows to simultaneously generate any number k of cuts, adopting $k = 1, 2, 3, 4$. We compare to the separation of maximally violated cuts, modified so as to guarantee that the cuts are undominated (U-Std) and to the generation of coordinated cutting planes (Coord), which is proposed in [1]. CPLEX 12.2 is used.

The promise of bound-optimal cuts is shown in the following charts, illustrating for `c-fat200-1` (resp. `c-fat200-5`) the evolution of the bound over the first 20 (resp. 60) cutting plane iterations. Indeed, with bound optimal cuts and $k = 1$, we always match the bounds obtained with Coord and, for a larger k, the bounds are much tighter.



This comes at a price, since the time required to generate a bound-optimal cut can be very large. Nevertheless, a more efficient cut generation might be obtained by adopting better formulations for the set $\mathcal{C}$, such as, for the max clique case, by adding some nontrivial clique inequalities.

Note that, due to the way it is defined, the bound-optimal cutting plane method tends to stall with a bound slightly larger than the best one. Nevertheless, since our method provides much tight bounds within the very first iterations, a possible way to overcome the convergence issue is that of switching to a standard cutting plane separation when the bound stops improving.

References

[1] E. Amaldi, S. Coniglio, and S. Gualandi. Coordinated cutting plane generation via multi-objective separation. *Mathematical Programming*, pages 1–24, 2012.

[2] R. Bixby. Advanced Mixed Integer Programming: Solving MIPs in practice. In *Combinatorial Optimization at Work 2, Berlin*, 2009.

[3] J.J. Forrest and D. Goldfarb. Steepest-edge simplex algorithms for linear programming. *Mathematical programming*, 57(1):341–374, 1992.

On specifying boundary conditions for the graph sandwich problem*

Fernanda Couto[1], Luerbio Faria[2], Sulamita Klein[1,3], Loana T. Nogueira[4], and Fábio Protti[4]

[1]COPPE/PESC, Universidade Federal do Rio de Janeiro, Brazil
[2]Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Brazil
[3]Instituto de Matemática, Universidade Federal do Rio de Janeiro, Brazil
[4]Instituto de Computação, Universidade Federal Fluminense, Niterói, Brazil

Abstract. The original graph sandwich problem for a property Π, as defined by Golumbic, Kaplan, and Shamir, can be stated as follows: given two graphs $G^1 = (V, E^1)$ and $G^2 = (V, E^2)$, is there a graph $G = (V, E)$ such that $E^1 \subseteq E \subseteq E^2$ and G satisfies Π (or belongs to a class Π)? The graph G is called a *sandwich graph*. The importance of the graph sandwich problem lies in the fact that it naturally generalizes the recognition problem of graphs with property Π, by setting $G^1 = G^2$. Our proposal is to introduce a generalization of the original graph sandwich problem, by specifying special properties Π_1, Π_2 for graphs G^1, G^2, respectively. In this new approach, each type of graph sandwich problem can be represented by a triple (Π^1, Π, Π^2)-SP, whose meaning is precisely "seek for a sandwich graph G satisfying Π, when it is known in advance that G^i satisfies Π^i, $i = 1, 2$". Such a generalization is called *graph sandwich problem with boundary conditions*. When G^i is not required to satisfy any special property, let us denote Π^i by $*$. One of the motivations for introducing boundary conditions is to develop a more refined complexity analysis of a NP-complete $(*, \Pi, *)$-SP problem, since its complexity status can change to polynomially solvable by suitably selecting Π^1, Π^2. For instance, let (k, ℓ) denote the class of (k, ℓ)-graphs (graphs whose vertex set can be partitioned into k stable sets and ℓ cliques). It is known that $(*, (k, \ell), *)$-SP is NP-complete for $k + \ell \geq 3$ and polynomial otherwise. In this work we prove that this problem can be solved in polynomial time when choosing convenient boundary conditions. We prove that (CHORDAL, (k, ℓ), CHORDAL)-SP, $(*, (2, 1),$ TRIANGLE-FREE)-SP, and $(*, (2, 1),$ BOUNDED DEGREE)-SP can be solved in polynomial time.
Keywords: boundary conditions, (k, ℓ)-graphs, sandwich problem

1 Introduction

Golumbic, Kaplan and Shamir introduced in [12] the GRAPH SANDWICH PROBLEM FOR PROPERTY Π in its original form, as follows:

*This work is partially supported by CNPQ and FAPERJ, Brazilian research agencies. List of emails: nandavdc@gmail.com, luerbio@cos.ufrj.br, sula@cos.ufrj.br, loana@ic.uff.br, fabio@ic.uff.br

Instance: $G^1 = (V, E^1)$ and $G^2 = (V, E^2)$, such that $E^1 \subseteq E^2$.
Question: Is there a graph $G = (V, E)$ such that $E^1 \subseteq E \subseteq E^2$ and G satisfies Π?

We call E^1 the set of *mandatory* edges, $E^2 \setminus E^1$ the set of *optional* edges, and $E(\overline{G^2})$ the set of *forbidden* edges. Hence, any sandwich graph $G = (V, E)$ for the pair G^1, G^2 must contain all mandatory edges and no forbidden edges. The *recognition problem* for a class of graphs $\mathcal{C}$ is equivalent to a particular graph sandwich problem where $E^1 = E^2$. Graph sandwich problems have drawn much attention because they naturally generalize graph recognition problems and have many applications [6, 7, 11, 15]. After studying many of them, we started questioning why not choosing special properties for graphs G^1 and G^2, since we can refine the complexity analysis of a sandwich problem by introducing such properties. Thus, we propose a generalized version of sandwich problems, that we call *sandwich problems with boundary conditions*. In this new approach, each type of problem is denoted by a triple (Π^1, Π ,Π^2)-SP, where input graphs G^1 and G^2 satisfy properties Π^1 and Π^2, respectively. We formalize it as follows:

GRAPH SANDWICH PROBLEM FOR PROPERTY Π WITH BOUNDARY CONDITIONS-(Π_1, Π, Π_2)-SP
Instance: $G^i = (V, E^i)$ satisfying Π^i, $i = 1, 2$, such that $E^1 \subseteq E^2$.
Question: Is there a graph $G = (V, E)$ such that $E^1 \subseteq E \subseteq E^2$ and G satisfies Π?

If G^i is not required to satisfy any special property, we denote Π^i by $*$. Golumbic et al. [12] remark that $(*, \Pi, *)$-SP is interesting when property Π is polynomially recognizable. In contrast, we remark that (Π^1, Π, Π^2)-SP has meaning when $(*, \Pi, *)$-SP is NP-complete, since (Π^1, Π, Π^2)-SP is not more difficult than $(*, \Pi, *)$-SP.

We focus on a particular property Π related to the (k, ℓ)-partition problem [8, 9]. A graph G is (k, ℓ) if $V(G)$ can be partitioned into k stable sets and ℓ cliques (some of them may be empty). In [1], the problem of recognizing (k, ℓ)-graphs was shown to be NP-complete if $k \geq 3$ or $\ell \geq 3$, and polynomially solvable otherwise. In [1–3, 9] polynomial-time algorithms are described for deciding if a graph admits a $(2, 1)$- or $(2, 2)$-partition. The problem $(*, (k, \ell), *)$-SP, for all $k + \ell \geq 3$, was shown to be NP-complete in [5] and polynomial if $k + l \leq 2$ [12]. In Section 2 we prove that this problem can be solved in polynomial time when choosing convenient boundary conditions. We study three problems: (CHORDAL, (k, ℓ), CHORDAL)-SP, $(*, (2, 1),$ TRIANGLE-FREE)-SP, and $(*, (2, 1),$ BOUNDED DEGREE)-SP.

2 Three Sandwich Problems with Boundary Conditions

Theorem 2.1. *Let $k, \ell \geq 0$ be fixed integers.* (CHORDAL, (k, ℓ), CHORDAL)-SP *is in P.*

The proof of Theorem 2.1 is based on Algorithm 1, which runs in $O(mn^\ell)$ time.

Algorithm 1: Algorithm for solving (CHORDAL, (k, ℓ), CHORDAL)-SP

```
Let C be the collection of maximal cliques of G_2 ;
for each subcollection {C_1, C_2, ..., C_ℓ} of C do
    let C' = V(C_1) ∪ V(C_2) ∪ ... ∪ V(C_ℓ) ;
    if G^1\C' is k-colorable then
        return G = (V, E_1 ∪ E(C_1) ∪ ... ∪ E(C_ℓ))
return there is no (k, ℓ)-graph G such that E^1 ⊆ E(G) ⊆ E^2
```

Theorem 2.2. (∗, (2,1), TRIANGLE-FREE)-SP *is in P.*

Algorithm 2 solves (∗, (2,1), TRIANGLE-FREE)-SP and runs in $O((n+m)m)$ time.

Algorithm 2: Algorithm for solving $(*,(2,1),$ TRIANGLE-FREE$)$-SP.

```
begin
    for each (u, v) ∈ E²\E¹ do
        V' := V\{u, v} ;
        G' = G¹[V'] ;
        if G' is bipartite then
            return G = (V, E¹ ∪ {(u, v)})
    return there is no (2, 1)-graph G such that E¹ ⊆ E(G) ⊆ E²
```

Finally, we state the following result:

Theorem 2.3. (∗ , (2,1), BOUNDED DEGREE)-SP *is in P.*

Algorithm 3 solves (∗ , (2,1), BOUNDED DEGREE)-SP by listing all maximal cliques of G^2 and testing, for each maximal clique, if the deletion of its vertices in G^1 yields a bipartite graph. It runs in $O(mn^{k+1})$ time, where $k = \Delta(G^2)$.

Algorithm 3: Algorithm for solving $(*,(2,1),$ BOUNDED DEGREE$)$-SP

```
begin
    let C = {C_1, ··· , C_l} be the collection of maximal cliques of G²;
    for each C_i ∈ C do
        if G¹\V(C_i) is bipartite then
            return G = (V, E¹ ∪ E(C_i))
    return there is no (2, 1)-sandwich graph G = (V, E) such that E¹ ⊆ E ⊆ E²
```

3 Conclusions

We observe that $(k,\ell)-$CHORDAL is a polynomially recognizable property [10,13,14]. Recently the versions (∗, CHORDAL-$(2,1),*)$-SP and (∗, STRONGLY CHORDAL-$(2,1),*)$-SP have been proved to be NP-complete [4]. As future works we are trying to choose properties Π^1 and Π^2 to analyze the complexity of (Π^1, CHORDAL-$(k,\ell),\Pi^2)$-SP and (Π^1, STRONGLY CHORDAL-$(k,\ell),\Pi^2)$-SP. Table 1 summarizes results of this paper for $k=2,\ell=1$, together with other results and open problems.

References

[1] A. Brandstadt. Partitions of graphs into one or two independent sets and cliques. *Discrete Mathematics*, **152**, (1996) pp. 47–54.

[2] A. Brandstadt and V. B. Le and T. Szymczak. The complexity of some problems related to Graph 3-colorability. *Discrete Applied Mathematics*, **89**, (1998) pp. 59–73.

[3] A. Brandstadt. Corrigendum. *Discrete Mathematics*, **186**, (2005) p. 295.

$(\Pi_1, (2,1), \Pi_2)$-SC				
$\Pi_1 \backslash \Pi_2$	CHORDAL	TRIANGLE-FREE	BOUNDED DEGREE k	$*$
CHORDAL	$O(mn)$	$O(m^2)$	$O(mn^{k+1})$	?
TRIANGLE-FREE	$O(mn)$	$O(m^2)$	$O(mn^{k+1})$	?
BOUNDED DEGREE k	$O(mn)$	$O(m^2)$	$O(mn^{k+1})$	?
$*$	$O(mn)$	$O(m^2)$	$O(mn^{k+1})$	NP-c [5]

Table 1: Complexity results and open problems for $(\Pi_1, (2,1), \Pi_2)$-SP, where properties Π_1, Π_2 are in {CHORDAL, TRIANGLE-FREE, BOUNDED DEGREE k , $*$}.

[4] F.Couto. Problemas Sanduíche para Grafos-(2,1) com Condições de Contorno (in Portuguese). M.Sc. thesis, (2012).

[5] S. Dantas and C.M. de Figueiredo and L. Faria. On decision and optimization (k,l)-graph sandwich problems. *Discrete Applied Mathematics*, **143**, (2004), pp. 155–165.

[6] S. Dantas and S. Klein and C. P. de Mello and A. Morgana. The Graph Sandwich Problem for P_4-sparse graphs. *Discrete Mathematics*. **309**, (2009) pp. 557–561 full paper Ars Comb. **88**, (2008) pp. 3664 – 3673.

[7] S. Dantas and R.B. Teixeira and C.M.H. Figueiredo. The polynomial dichotomy for three nonempty part sandwich problems. *Discrete Applied Mathematics* , (2010) pp. 1286–1304.

[8] T. Feder and P. Hell and S. Klein and R. Motwani. Complexity of graph partition problems. Proceedings of the thirty-first annual ACM symposium on Theory of computing, (1999), pp. 464–472.

[9] T. Feder and P. Hell and S. Klein and R. Motwani. List partitions. *SIAM Journal Discrete Mathematics*, **16**, (2003), pp. 449–478.

[10] T. Feder and P. Hell and S. Klein and L. T. Nogueira and F. Protti. List matrix partitions of chordal graphs. *Theoretical Computer Science*, **349**, (2005), pp. 52–66.

[11] C.M.F. Figueiredo and L. Faria and S. Klein and R. Sritharan. On the complexity of the sandwich problems for strongly chordal graphs and chordal bipartite graphs. *Theoretical Computer Science*, **381**, (2007) pp. 57–67.

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

[13] P. Hell and S. Klein and L. T. Nogueira and F. Protti. Partitioning chordal graphs into independent sets and cliques. *Discrete Applied Mathematics*, **141**, (2004), pp. 185–194.

[14] P. Hell and S. Klein and L. T. Nogueira and F. Protti. Packing r-cliques in weighted chordal graphs. Annals of OR, **138**, (2005), pp. 179–187.

[15] R. Sritharan. Chordal bipartite completion of colored graphs. *Discrete Mathematics*, **308**, (2008), pp. 2581–2588.

A tight bound on the number of minimal dominating sets in split graph

Jean-François Couturier[1] and Mathieu Liedloff[2]

[1]LITA, Université de Lorraine, 57045 Metz Cedex 01, France, couturier@univ-metz.fr
[2]LIFO, Université d'Orléans, 45067 Orléans Cedex 2, France, mathieu.liedloff@univ-orleans.fr

Given a graph $G = (V, E)$, a subset $D \subseteq V$ is a dominating set if each vertex of $V \setminus D$ has at least one neighbor in D. The set D is minimal if no proper subset of D is a dominating set. It has been proved that any n-vertex graph has at most 1.7159^n such minimal dominating sets whereas a lower bound of 1.5705^n is known. Several graph classes have been studied recently and for few of them the proved upper bound match the lower bound.

In this paper we prove an upper bound of $O(1.4423^n)$ for split graphs. This result is tight and settles a conjecture given in [1]. Since our result is algorithmically based, all minimal dominating sets of a split graphs can be enumerated within the same bound.

1 Introduction

One of the most famous results about independent sets is due to Moon and Moser [8]; it states that the maximum number of maximal independent sets in any n-vertices graph is $3^{n/3}$. Such a bound is the key-point of (the running-time analysis of) several algorithms, like the coloring algorithm by Lawler [7].

Others graphs properties has been also recently considered, like minimal feedback vertex sets, bicliques, potential maximal cliques, dominating sets [2, 6, 5, 3, 1] with algorithmic consequences for some well-known problems. Regarding the dominating set property, Fomin et al. [3] have shown that the maximum number of minimal dominating set at most 1.7159^n and at least 1.5705^n. Recently, Couturier et al. [1] have considered several graph classes (chordal, cobipartite, split, proper interval, cographs, trivially perfect, threshold and chain graphs) and established corresponding lower bounds and upper bounds on their maximum number of minimal dominating sets. For some of these graph classes, a tight bound was obtained (*i.e.* the lower bound matches the upper bound), whereas a gap exists for others. In particular for split graphs, a $3^{n/3} \approx 1.4422^n$ lower bound and 1.4656^n upper bound have been proved, and it is conjectured (see [1]) that the upper bound should be $3^{n/3}$. In this paper we settle the conjecture and show the $3^{n/3}$ upper bound (up to a polynomial factor).

2 Preliminaries

Given a graph $G = (V, E)$, a set $D \subseteq V$ is called a dominating set if each vertex of $V \setminus D$ has at least one neighbor in D. The set D is called minimal dominating set (mds) if there is no

$D' \subset D$ such that D' is a dominating set. A vertex $x \in V \setminus D$ is called a private neighbor of a vertex $v \in D$ if $N[x] \cap D = \{v\}$. If D is a mds then any $v \in D$ has a private neighbor.

A graph $G = (V, E)$ is a split graphs if its vertex set can be partitioned into a clique C and an independent set I. Given a vertex $v \in C$ we denote by $d_I(v)$ the number of neighbors of v in I. Given a set $X \subseteq V$, we let $N_X(v) = N(v) \cap X$.

3 An algorithm to enumerate mds in split graphs

Throughout this section we assume $G = (V, E)$ to be a split graph, given together with a partition (C, I) of its vertex set into a clique C and an independent set I.

To enumerate all mds of a given split graph $G = (V, E)$ we design a branching algorithm. Its execution can be viewed as a search tree, whose leaves correspond to the minimal dominating sets of G. Thus, by upper bounding the maximum number of leaves of this search tree, we obtain an upper bound on the maximum number of mds. It is standard to establish linear recurrences to analyze the running-time of branching algorithms, and we refer the reader to the book by Fomin and Kratsch [4] for further details. Typically, we define recurrences of type $T(n) = \sum_{i=1}^{k} T(n - r_i)$, where r_i is the number of vertex removed before the corresponding recursive call. We name $(r_1, r_2, \ldots, r_k)$ the branching vector of this recurrence. Finally, the running-time $O^*(\alpha^n)$ is obtained by taking α^n as the worse-case solution over all the recurrences. We are now ready to describe our algorithm. Note that it is a collection of (sub-)cases and we assume that a case is applied only if no other case appearing before can be applied. Suppose D to be the current minimum dominating set that is building and let v be a vertex of C with a maximum number of neighbor in I.

Description of the algorithm

Case 1. If $d_I(v) \geq 3$ then we branch into :

- "$v \in D$". At least 4 vertices (v and $N_I(v)$) are deleted.
- "$v \notin D$". Whenever a neighbor of v will be dominated, vertex v will be dominated too; so we can safely delete v.

This gives a $(4, 1)$ branching vector, whose corresponding solution is $O(1.3803^n)$.

Case 2. If $d_I(v) = 2$ then we let $N_I(v) = \{x, y\}$. Wlog, we assume $d(x) \leq d(y)$.

Case 2.1. If $d(x) = 1$ then v is its only neighbor and there are two possibilities to dominate x:

- "$x \in D$". By minimality of D, vertex v cannot be in the dominating set and x and v are removed.
- "$x \notin D$". Thus to dominate x we need to add v to D. Vertex y is removed as it is also dominated by v, and by minimality of D cannot belongs to D (recall that C is a clique and $v \in C$).

This gives a $(2, 3)$ branching vector and the corresponding solution is $O(1.3248^n)$.

Case 2.2. If $d(x) = 2$ then we call v' the other neighbour of x.

Case 2.2.1. If $d(v')_I = 1$ then x is the only neighbour of v' in I, and there is three possibilities to dominate x:

- "$v \in D$". Because x is the only neighbour of v' in I, there is no more minimal dominating set containing v'. So at least 4 vertices (v, $\{x, y\} \subseteq N_I(v)$ and v') are deleted.
- "$v' \in D$". If v is not in the dominating set, we can use v' to dominate x. We delete x, v and v'. Note that the case "$v \in D$" was previously considered so we know that v cannot be longer added to D (and v is dominated by v').

- "$x \in D$". If neither v or v' are in the dominating set, we have to add x to D. Again, we delete v, v' and x.

This gives a $(4, 3, 3)$ branching vector and its solution is $O(1.3954^n)$.

Case 2.2.2 If $d(v')_I \geq 2$ and $y \notin N(v')$ then $d_I(v') = 2$ as $d_I(v)$ is maximum over all vertices $v \in I$. Let us call z the other neighbor of v' in I.

- "$v \in D$ and x is its private neighbor". We can delete v, v' and $N_I(v)$, *i.e.* at least 4 vertices.
- "$v \in D$ and y is its private neighbor". As x is not the private neighbor of v, it follows that v' belongs to D. Thus, at least 6 vertices are deleted (v, x, y, v', z and $N(y)$).
- "$v \notin D, v' \in D$". In this case, x and z are dominated by v' and can be deleted, as well as v and v'.
- "$x \in D$". As neither v and v' are in D, x is added to D and $N(x)$ is removed.

This gives a recurrence whose branching vector is no worse that $(4, 6, 4, 3)$ branching vector, giving $O(1.4064^n)$.

Case 2.2.3. If $d_I(v') \geq 2$ and $y \in N(v')$ then $N(v') = N(v)$.

- "$v \in D$". By minimality, v' can never be added to D. We delete v, v', x and y.
- "$v' \in D$". This branch is symmetric to the previous one.
- "$x \in D$". As v and v' are not in D, x belongs to D and $N[x]$ is removed.

The corresponding branching vector is $(4, 4, 3)$. That is $O(1.3533^n)$.

Case 2.3. If $d(x) = 3$ then we call v, v'_1 and v'_2 its neighbors. Wlog, assume that $d(v'_1) \leq d(v'_2)$.

Case 2.3.1. If $y \notin N(v'_1) \cup N(v'_2)$ then we branch in the following ways :

- "$v \in D$ and x is its private neighbor". We delete 5 vertices: v, $N_I(v)$ and $N(x)$.
- "$v \in D$, y is its private neighbor and $v'_1 \in D$". If $d_I(v'_1) = 1$ this branch is skipped as it is not minimal to add both v'_1 and v to D. Thus we assume that $d_I(v'1) \geq 2$. In that case, we delete v, v'_1, x, y, $N(y)$ (as v is its private neighbor) and $N_I(v'_1)$; *i.e.* at least 7 vertices.
- $v \in D$, y is its private neighbor and $v'_2 \in D$". Due to the previous branch, $v'_1 \notin D$. Again, in this branch we can assume that $d_I(v'_2) \geq 2$ otherwise we skip it. So we delete v, v'_1, v'_2, x, y, $N(y)$, and $N_I(v'_2)$, *i.e.* at least 8 vertices.
- "$v \notin D$". In this case we only delete v.

This gives a recurrence whose branching vector is no worse than $(5, 7, 8, 1)$, which gives $O(1.4331^n)$.

Case 2.3.2. If $y \in N(v'_1) \cup N(v'_2)$ then we branch in the following ways. Note that at least one branch is skipped.

- "$v \in D$ and x is its private neighbor". We delete 5 vertices : v, $N_I(v)$ and $N(x)$.
- "$v \in D$, y is its private neighbor and $v'_i \in D$". If $y \in N(v'_1) \cap N(v'_2)$ this branch is skipped as y cannot be a private neighbor of v; otherwise let i such that $y \notin N(v'_i)$. If $d_I(v'_i) = 1$ this branch is skipped as it is not minimal to add both v'_1 and v to D; otherwise we delete v, v'_i, x, y, $N(y)$ (as v is its private neighbor) and $N_I(v'_i)$; *i.e.* at least 7 vertices.
- "$v \notin D$". In this case we only delete v.

This gives a recurrence whose branching vector is no worse than $(5, 7, 1)$, or $(5, 1)$ in the case that $y \in N(v'_1) \cap N(v'_2)$. This later case gives $O(1.3248^n)$.

Case 2.4. If $d(x) \geq 4$ then we do the following branching:

- "$v \in D$ and x is its private neighbor". So we delete v, x, y and $N(x)$. At least 6 vertices removed.

- "$v \in D$ and y is its private neighbor". So we delete v, x, y and $N(y)$. Again, at least 6 vertices are removed.
- "$v \notin D$". We only delete v.

This gives us a $(6, 6, 1)$ branching vector, corresponding to $O(1.3881^n)$. Observe that the mds where x and y are only dominated by v are generated by both the first two branchings, so some minimal dominating sets can be output more than once by our algorithm.

Case 3. As all the previous cases can be no longer applied, it results that for each vertex $v \in C$, $d(v) \leq 1$. Let's take a vertex $x \in I$ of maximum degree and denote by $v_1, \ldots, v_{d(x)}$ its neighbors. Observe that as soon as x is dominated, all its neighbors can be immediately removed. This suggests the following branching :

- "$x \in D$" and $N[x]$ is removed.
- "$x \notin D$ and $v_i \in D$" for $1 \leq i \leq d(x)$. We have $d(x)$ possibilities to pick a neighbour of x and to add it to D. In each case, we delete $N[x]$.

This gives the $(d(x)+1, d(x)+1, \ldots, d(x)+1)$ branching vector of length $(dx)+1$ and its worst case solution is obtained when $d(x) = 2$ and thus $O(3^{n/3}) = O(1.4423^n)$.

Case 4. Finally if the graph contains no more edges then all remaining vertices of I are added to D. If $D \cap C$ is not empty then the remaining vertices of C are removed, otherwise for each vertex v of C we add it to D and we return $D \cup \{v\}$.

End of the Algorithm

The previous algorithm and its running-time analysis show the Theorem :

Theorem 3.1. *The minimum dominating sets of a split graph can be enumerated in time* $O(poly(n) \cdot 3^{n/3}) = O(1.4423^n)$.

References

[1] J.F. Couturier, P. Heggernes, P. van 't Hof, and D. Kratsch. Minimal Dominating Sets in Graph Classes: Combinatorial Bounds and Enumeration. Proceedings of SOFSEM 2012, pp. 202-213 (2012).

[2] F. V. Fomin, S. Gaspers, A. V. Pyatkin, and I. Razgon. On the minimum feedback vertex set problem: Exact and enumeration algorithms. Algorithmica 52(2): 293-307 (2008).

[3] F. V. Fomin, F. Grandoni, A. V. Pyatkin, and A. A. Stepanov. Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications. ACM Trans. Algorithms 5(1): (2008).

[4] F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms.* Springer, Texts in Theoretical Computer Science (2010).

[5] F. V. Fomin and Y. Villanger. Finding induced subgraphs via minimal triangulations. Proceedings of STACS 2010, pp. 383-394 (2010).

[6] S. Gaspers, D. Kratsch, and M. Liedloff. On Independent Sets and Bicliques in Graphs. Algorithmica 62(3-4): 637-658 (2012).

[7] E. L. Lawler. A note on the complexity of the chromatic number problem. Inform. Proc. Lett. 5(3): 66-67 (1976).

[8] J. W. Moon and L. Moser. On cliques in graphs. Israel J. Math. 3: 23-28 (1965).

A Quantization Framework for Smoothed Analysis on Euclidean Optimization Problems

Radu Curticapean[1] and Marvin Künnemann[1,2]

[1]Universität des Saarlandes, Saarbrücken, Germany
[2]Max-Planck-Institut für Informatik, Saarbrücken, Germany

Smoothed analysis has been introduced by Spielman and Teng (2004) to analyze the performance of algorithms on real-world instances, providing a more realistic view than average-case or even worst-case analysis. Typical performance measures that can be analyzed within this framework include running time and approximation ratio.

Recent results (Bläser et al., 2012) give an optimistic view on the tractability of some Euclidean optimizations problems by employing the paradigm of smoothed analysis. In particular, this paradigm proves useful for designing and analyzing asymptotically optimal approximation algorithms. We extend this view by providing a general framework for designing fast algorithms whose approximation performance converges to optimality if a sufficiently large perturbation of the input is assumed. Applications of our framework include approximation algorithms for maximum Euclidean matching and TSP, bin packing and k-means clustering.

1 Smoothed Analysis

Smoothed Analysis has been introduced by Spielman and Teng [8] to give a theoretical foundation for analyzing the practical performance of algorithms. In particular, this analysis paradigm helps to provide an explanation why the simplex method is observed to run fast in practice despite its exponential worst-case running time.

Central to smoothed analysis is the concept of an adversarial decision merged with randomization. This follows the reasoning that even if hard instances exist, they are typically isolated, specifically constructed and hence unlikely to occur in practice. But in contrast to average-case analysis, for which the probability distribution of the input instances is assumed to be known, smoothed analysis lets an adversary choose worst-case distributions of bounded "power". These distributions are unknown to the analyzed algorithm.

We call a problem *smoothed tractable* if it admits a linear-time algorithm with an approximation ratio that can be bounded by $1 - o(1)$ with high probability over the perturbation of the input. Assuming perturbed input is especially plausible in a Euclidean setting in which the data may result from physical measurements, e.g., from locating a position on a map with an inherent inaccuracy.

A widely-used and very general perturbation model is the *one-step model*, which has been successfully applied for a number of problems, see, e.g., [1, 2, 3, 5]. In this model, an adversary chooses probability densities according to which the input instance is drawn. To prevent the adversary from modeling a worst-case instance too closely, we impose restrictions on the density functions, using a parameter ϕ. Roughly speaking, for large ϕ, we expect the algorithm to perform almost as bad as on worst-case instances. Likewise, choosing ϕ as small as possible requires the adversary to choose the uniform distribution on the input space, which corresponds to an average-case analysis. Thus, the adversarial power ϕ serves as an interpolation parameter between worst case and average case.

Formally, given a set of feasible distributions $\mathcal{F}$ depending on ϕ and a performance measure t, we define the smoothed performance of an algorithm under the perturbation model $\mathcal{F}$ as $\max_{f_1,\dots,f_n\in\mathcal{F}} \mathrm{E}_{(X_1,\dots,X_n)\sim(f_1,\dots,f_n)}[t(X_1,\dots,X_n)]$. In this work, we will be concerned with analyzing both the smoothed approximation ratio and bounds on the approximation ratio that hold with high probability over the perturbations.

This work complements the analysis of the smoothed performance of partitioning heuristics for Euclidean minimization problems previously conducted by Bläser et al. [2]. Their framework applies to so-called *smooth* and *near-additive* functionals and analyzes algorithms that patch local solutions to a solution on the whole instance. We establish an additional framework for smoothed analysis on a general class of Euclidean functionals that is disjoint to the smooth and near-additive functionals. This class contains optimization problems whose objective values are sufficiently large to compensate for rounding errors. In particular, we consider the maximization counterparts of two problems studied in [2], Euclidean matching and TSP.

Since smoothed analysis incorporates average-case analysis, we generalize average-case analysis results for Euclidean maximum matching and TSP [4] as well as for bin packing [7]. However, for bin packing, Karger and Onak [6] have already provided a linear-time algorithm that is asymptotically optimal on instances smoothed with any constant ϕ, which we extend slightly. To the best of our knowledge, this is the only problem that fits into our framework and has already been analyzed under perturbation.

2 Framework

Our framework builds on the notion of *quantizable* functionals. These are functionals that admit fast approximation schemes on perturbed instances using general rounding strategies. The idea is to round an instance of n points to a *quantized instance* of $\ell \ll n$ points, each equipped with a multiplicity. This quantized input has a smaller problem size, which allows to compute an approximation faster than on the original input. However, the objective function needs to be large enough to make up for the loss due to rounding.

Definition 2.1. *Let $d \geq 1$ and $\mathcal{F}$ be a family of probability distributions $[0,1]^d \to \mathbb{R}_{\geq 0}$. Let $t, R : \mathbb{N} \to \mathbb{R}$ and $Q \in \mathbb{R}$. We say that a Euclidean functional $F : ([0,1]^d)^* \to \mathbb{R}_{\geq 0}$ is* t-time (R,Q)-quantizable *with respect to $\mathcal{F}$, if there is a* quantization algorithm *A and an* approximation functional *$g : ([0,1]^d \times \mathbb{N})^* \to \mathbb{R}$ with the following properties. For any function ℓ satisfying $\ell \in \omega(1)$ and $\ell \in o(n)$,*

1. *The quantization algorithm A maps a collection of points $X = (X_1,\dots,X_n) \in [0,1]^{dn}$ to a multiset $A(X) = X' = ((X'_1, n_1),\dots,(X'_\ell, n_\ell))$, the* quantized input*, with $X'_i \in [0,1]^d$ in time $O(n)$.*

2. *The approximation functional g is computable in time $t(\ell)$ and, for any $f \in \mathcal{F}^n$, fulfills* $\Pr_{X\sim f}[|F(X) - g(A(X))| \leq nR(\ell)] \in 1 - o(1)$.

3. *For any $f \in \mathcal{F}^n$, we have* $\Pr_{x\sim f}[F(X) \geq nQ] \in 1 - o(1)$.

Quantizable functionals induce natural approximation algorithms on smoothed instances. We can thus restrict our attention to finding criteria that make a functional quantizable.

Theorem 2.2. *Let $\mathcal{F}$ be a family of probability distributions and F be $t(\ell)$-time $(R(\ell), Q)$-quantizable with respect to $\mathcal{F}$. Then for every ℓ with $\ell \in \omega(1)$ and $\ell \in o(n)$, there is an approximation algorithm* ALG *with the following property. For every $f \in \mathcal{F}^n$, the approximation* $\mathrm{ALG}(X)$ *on the instance X drawn from f is $(1-\frac{R(\ell)}{Q})$-close to $F(X)$ with high probability. The approximation can be computed in time $O(n + t(\ell))$.*

For all problems considered here, we also design algorithms whose *expected* approximation ratio converges to optimality. A sufficient condition for F to allow for such an algorithm is that a linear-time algorithm approximating F within a constant factor $0 < c < 1$ exists.

We propose two methods to verify quantizability. The first, *grid quantization*, partitions the unit-cube into ℓ equal-volume cells using a grid. By rounding each point to the centroid of its corresponding cell, a high-multiplicity version of the problem is obtained that typically has an objective value close to the original value. Using algorithms that run fast in this compact representation, quantizability can typically be established immediately. Applications of this quantization procedure include Euclidean maximum matching and TSP as well as bin packing.

If no fast algorithm for the high-multiplicity version is known, *balanced quantization* might be applied. This method determines a number $\ell' \approx \ell$ of cells $C_1, \ldots, C_{\ell'}$ with three properties: (1) each cell contains exactly the same number of points, (2) the diameter of each cell converges to zero for n tending to infinity and (3) all but a sublinear number of points are covered by the cells. By compressing the original instance to the centroids of the cells $C_1, \ldots, C_{\ell'}$, a much smaller instance is obtained whose objective value (or optimal solution of the optimization problem at hand) can, for certain functionals, be transformed to a close approximation of the optimal solution of the original instance. This approach enables an approximation of k-means clustering and can additionally be used to improve the quantization of Euclidean maximum matching and TSP for higher dimensions ($d \geq 3$).

3 Results

Our findings are summarized in Table 1. We obtain fast and simple approximation algorithms on sufficiently smoothed inputs for the following problems: The maximum Euclidean matching problem $\mathrm{MaxM}(X)$, the maximum Euclidean Traveling Salesman problem $\mathrm{MaxTSP}(X)$, the k-means clustering problem $\mathrm{KMeans}(X; k)$ where k denotes the number of desired clusters, and the d-dimensional bin packing problem $\mathrm{BP}_d(X)$. The approximation ratio converges to one with high probability over the random inputs. Additionally, all of these algorithms can be adapted to additionally yield asymptotically optimal expected approximation ratios.

Our results extend the rather optimistic view of smoothed tractability. Due to its generality and the examples provided here, the framework suggests that combining these two relaxations of tractability (asymptotically optimal approximation and perturbation of the input) may enable an analysis of general rounding techniques for hard optimization problems. The perturbation assumption is especially plausible in the problem settings we consider here.

problem	running time	restriction on adversary power
MaxM(X)	$O(n)$	$\phi \in o(\sqrt[4]{n})$ or $\phi \in o(n^{\frac{1}{2}\frac{d}{d+2}-\varepsilon})$
MaxTSP(X)	$O(n)$	$\phi \in o(\sqrt[4]{n})$ or $\phi \in o(n^{\frac{1}{2}\frac{d}{d+2}-\varepsilon})$
KMeans($X;k$)	$O(n)$	$k\phi \in o(n^{\frac{1}{2}\frac{1}{kd+1}\frac{d}{d+1}})$
$\mathrm{BP}_1(X)$	$O(n \log n)$	$\phi \in o(n^{1-\varepsilon})$
$\mathrm{BP}_d(X)$	$O(n)$	$\phi \in o\left(\sqrt[d(d+1)]{\log\log n / \log^{(3)} n}\right)$

Table 1: All (near) linear-time algorithms derived in our framework

Our framework complements a previous framework by Bläser et al. [2] for smooth and near-additive Euclidean functions that is orthogonal to our approach. A smooth Euclidean functional F on n points has, by definition, a value that is too small to compensate for the rounding errors that our quantization methods induce. This implies that for any Euclidean functional, at most one of both frameworks is applicable.

References

[1] R. Beier and B. Vöcking. Typical properties of winners and losers in discrete optimization. *SIAM Journal on Computing*, 35(4):855–881, 2006.

[2] M. Bläser, B. Manthey, and B. V. R. Rao. Smoothed analysis of partitioning algorithms for euclidean functionals. *Algorithmica*, pages 1–22, 2012.

[3] E. Boros, K. Elbassioni, M. Fouz, V. Gurvich, K. Makino, and B. Manthey. Stochastic mean payoff games: Smoothed analysis and approximation schemes. In *38th Int. Coll. on Automata, Languages and Programming*, ICALP'11, pages 147–158. Springer, 2011.

[4] M. E. Dyer, A. M. Frieze, and C. J. H. McDiarmid. Partitioning heuristics for two geometric maximization problems. *Operations Research Letters*, 3(5):267–270, 1984.

[5] M. Englert, H. Röglin, and B. Vöcking. Worst case and probabilistic analysis of the 2-opt algorithm for the TSP: Extended abstract. In *18th Ann. ACM-SIAM Symp. on Discrete Algorithms*, SODA '07, pages 1295–1304, Philadelphia, PA, USA, 2007. SIAM.

[6] D. Karger and K. Onak. Polynomial approximation schemes for smoothed and random instances of multidimensional packing problems. In *18th Ann. ACM-SIAM Symp. on Discrete Algorithms*, SODA '07, pages 1207–1216, Philadelphia, PA, USA, 2007. SIAM.

[7] R. M. Karp, M. Luby, and A. Marchetti-Spaccamela. A probabilistic analysis of multidimensional bin packing problems. In *16th Annual ACM Symp. on Theory of Computing*, STOC '84, pages 289–298, New York, NY, USA, 1984. ACM.

[8] D. A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

Linear Time and Almost Linear Time Cases for Minimal Elimination Orderings

Elias Dahlhaus[1]

[1]Department of Computer Science, Darmstadt University of Technology

1 Introduction

One of the major problems in computational linear algebra is that of sparse Gauss elimination. The problem is to find a pivoting, such that the number of zero entries of the original matrix that become nonzero entries during the elimination process is minimized. In case of symmetric positive definite matrices, it translates to the following graph theory problem [10].

Minimum Elimination Ordering: Given a graph $G = (V, E)$, find a superset E' of E of minimum size and ordering $<$ on V, such that the greater E'-neighbors of any vertex is complete in $G' = (V, E')$.

Note that this problem is NP-complete [13].

For this reason, one considers also the following relaxation of the problem.

Minimal Elimination Ordering: Given a graph $G = (V, E)$, find an inclusion minimal superset E' of E and ordering $<$ on V, such that the greater E'-neighbors of any vertex is complete in $G' = (V, E')$. This problem can be solved in $O(nm)$ time [11].

The Minimal Elimination Ordering problem can be solved in $O(nm)$ time in general [11] and for planar graphs and for graphs of bounded degree in linear time [4, 5].

In this paper, first a simpler linear time algorithm for the minimal elimination ordering problem for planar graphs is presented.

It is to mention that the fill-in of a minimal elimination ordering can be much larger than the fill-in of a minimum elimination ordering. But there are heuristics, like nested dissection, that are quite appropriate to get an ordering that is not too far from a minimum elimination ordering. On the other hand, such an ordering needs not to be minimal. In so far, it is of interest, to transform any nested dissection or even any vertex ordering to a minimal elimination ordering. A general apporach is in [1]. Here we are interested in linear time special cases. We consider any edge nested dissection, i.e. a recursive partition of the vertices, such that the number of edges joining two vertices of different sets of the partition is small. The algorithm of 2 can be used to find a minimal elimination ordering that is faithful to the given edge nested dissection.

Finally, we consider a vertex nested dissection, i.e. we consider a minimal vertex separator that divides a given planar graph into two components that are of almost equal size. Vertices in the components are considered as smaller than the vertices of the separator. The separation

procedure is applied recursively to the components. Bornstein, Maggs, and Miller [2] developed an algorithm that transformed the nested dissection ordering into a minimal elimination ordering. We will show that structural properties of planar graphs allow us to implement the algorithm of [2] in $O(n \log n)$, provided the given graph is planar.

2 Minimal Elimination Ordering

We start with a result that is also given in [5]. We call an ordering $<$ *compatible* with an ordered partition $(V_1, \ldots, V_k)$ if with $x \in V_i$, $y \in V_j$, and $i < j$, we have $x < y$. We call an ordered partition $V_1, \ldots, V_k$ an *approximation* of a minimal elimination ordering if some minimal elimination ordering is compatible with it.

We determine an approximation of a minimal elimination ordering as follows.

1. We determine a spanning tree T of G and a postorder enumeration $v_1, \ldots, v_n$ of the vertices of T (and therefore of G).
2. The approximation $(V_1, \ldots, V_n, V_{n+1}$ is defined as follows.
   - $V_{n+1} = \{v_n\}$
   - For $i = 1, \ldots, n$, V_i consists of those vertices u of G, such that u is in the neighborhood of v_i but not in the neighborhood of a v_j with $j > i$, i.e. Let $j(u)$ be the maximum i, such that $v_i u$ is an edge of G. Then $V_i = \{u | j(u) = i\}$.

Theorem 2.1. *$(V_1, \ldots, V_{n+1})$ is an approximation of a minimal elimination ordering.*

Lemma 2.2. *[9] $(V_1, \ldots, V_{n+1})$ can be determined in $O(\log n)$ time with a linear workload by a CREW-PRAM if a spanning tree T for G is known.*

The overall complexity to get the ordered partition $(V_1, \ldots, V_{n+1}$ is therefore $O(n + m)$ sequential time and in parallel $O(n+m)$ processors and $O(\log n)$ time ona CRCW-PRAM (bound to compute a spanning tree [12]).

The following result is known (see for example [11].

Lemma 2.3. *Let $<$ be an ordering on the vertices of G and let C be a connected component of $G[\{w | w < v\}]$. Then in the chordal completion of G and $<$, all neighbors u of C with $u \geq v$ are pairwise adjacent.*

As a consequence, we get the following.

Corollary 2.4. *Let C be a connected component of $V_1 \cup \ldots \cup V_{i-1}$. Then the vertices that are neighbors of C in G' and that do not belong to C are pairwise adjacent in any fill-in of an ordering $<$ that is compatible with $(V_1, \ldots, V_n, V_{n+1})$.*

Now we assume that the given graph is planar. We consider any level V_i and any connected component C of $\bigcup_{j<i} V_j$. The neighborhood of C is, due to planarity, cyclicylly ordered, and neighbors with respect to the cyclic ordering can be joined by edges, and the graph remains still planar. ;oreover, these edges are fill-in edges or edges of the original graph. Due to the fact that all vertices of V_i are neighbors of v_i, the given graph restricted to V_i is outerplanar. Some faces are filled with vertices of smaller level and have therefore to be declared to be complete. Other faces do not contain vertices of smaller level and have to be triangulated. So we get a chordal extension of V_i that is representable in linear space and computable in linear time. To get the final refinement of V_i, we make use of the cyclic ordering of the neighborhood of any connected component of V_i in levels greater than V_i.

3 Transforming an Edge Nested Dissection into a Minimal Elimination Ordering

An edge nested dissection is done as follows.

1. We partition the vertices of the given graph into two almost equal sized components, such that the number of edges between the two components is small.

2. We recursively apply this procedure, for each of these components.

Vertices that are incident to edges between the components are considered as larger than vertices that are only adjacent to one of the components.

For our considerations, it is only relevant that we have a system of connected subsets of the vertex set that are ordered tree-like, We call such a structure a *clustered graph* [6]. Sets of the system are also called *clusters*. In [6] it has been shown the following.

Lemma 3.1. *Given a clustered graph with connected clusters. We can determine a spanning tree in linear time, such that the restriction to each cluster forms a subtree.*

The key result is the following.

Theorem 3.2. *Given a clustered graph with connected clusters and a spanning tree as determined in previous lemma. One can determine an ordering on the vertices, such that*

1. *Each final segment forms a subtree.*

2. *The vertices of any cluster appear consecutively in the ordering.*

Assuming that the clustered graph is planar or degree bounded, one applies the algorithm in previous section.

4 Less Parallel Vertex Nested Dissection for Planar Graphs

A vertex nested dissection is determined as follows.

1. Determine a minimal separator that partitions the graph into two almost equal sized components.

2. Apply this recursively to the components.

Vertices in the separator are considered as larger that vertices not in the separator. Bornstein, Maggs, and Miller [2] used a similar idea as in [7] to transform a nested dissection into a minimal elimination ordering.

1. Determine a minimal extension to a chordal graph of each component when we remove the vertices of the main minimal separator.

2. Determine, for each minimal separator of the minimal chordal extension of each component, the the connected component with all its neighbors in the separator with most neighbors in the main separator and consider all vertices in other connected components as smaller than the vertices in the separator.

To get everything efficient for planar graphs, we make use of the fact that minimal separators in planar graphs are cycles or paths in some triangulation and of the cyclic ordering of the neighborhood of the neighborhood of any connected component when we remove the main separator.

Since the recursion depth is $O(\log n)$, we get the following result.

Theorem 4.1. *A vertex nested dissection of a planar graph can be transformed into a minimal elimination ordering with less fill-in edges in $O(n \log n)$ time.*

References

[1] A. Berry, E. Dahlhaus, P. Heggernes, G. Simonet, *Sequential and parallel triangulating algorithms for Elimination Game and new insights on Minimum Degree*, Theoretical Computer Science, Vol.409,3(2008), pp. 601-616

[2] C. Bornstein, B. Maggs, G. Miller, *Tradeoffs Between Parallelism and Fill in Nested Dissection*, Proceedings of the Thirty-Eighth Annual Symposium on Foundations of Computer Science (1997).

[3] P. Bunemann, *A Characterization of Rigid Circuit Graphs*, Discrete Mathematics 9 (1974), pp. 205-212.

[4] E. Dahlhaus, *Minimal Elimination of Planar Graphs*, Algorithm Theory - SWAT'98, LLNCS 1432 (1998), pp. 210-221.

[5] E. Dahlhaus, *Minimal Elimination Ordering for Graphs of Bounded Degree*, submitted.

[6] E. Dahlhaus, *A linear time algorithm to recognize clustered planar graphs and its parallelization*, LATIN'98: Theoretical Informatics Lecture Notes in Computer Science Volume 1380, 1998, pp 239-248

[7] Elias Dahlhaus, Marek Karpinski, *An Efficient Parallel Algorithm for the Minimal Elimination Ordering (MEO) of an Arbitrary Graph*, Theoretical Computer Science 134 (1994), pp. 493-528.

[8] J. Gilbert, R. Tarjan, *The Analysis of a Nested Dissection Algorithm*, Numerische Mathematik 50 (1987), pp. 427-449.

[9] P. Klein, *Efficient parallel algorithms for chordal graphs*, SIAM J. Comput. 25 (1996), pp. 797-827 (1996).

[10] D. Rose, *Triangulated Graphs and the Elimination Process*, Journal of Mathematical Analysis and Applications 32 (1970), pp. 597-609.

[11] D. Rose, R. Tarjan, G. Lueker, *Algorithmic Aspects on Vertex Elimination on Graphs*, SIAM Journal on Computing 5 (1976), pp. 266-283.

[12] Y. Shiloach, U. Vishkin, *An O(log n) Parallel Connectivity Algorithm*, Journal of Algorithms 3 (1982), S. 57-67.

[13] M. Yannakakis, Computing the Minimum Fill-in is NP-complete, *SIAM Journal on Algebraic and Discrete Methods* 2 (1981), pp. 77-79.

On total coloring and equitable total coloring of cubic graphs with large girth*

S. Dantas[1], C. M. H. de Figueiredo[2], G. Mazzuoccolo[3], M. Preissmann[3], V. F. dos Santos[2], and D. Sasaki[2]

[1]IME, Universidade Federal Fluminense, Brazil
[2]COPPE, Universidade Federal do Rio de Janeiro, Brazil
[3]CNRS/Grenoble-INP/UJF-Grenoble 1, G-SCOP UMR5272 Grenoble, F-38031, Grenoble, France

We determine the total chromatic number and the equitable total chromatic number of some infinite families of snarks and of generalized Petersen graphs. Moreover, a sufficient condition for a cubic graph not to have total chromatic number 4 is presented.

1 Introduction

Let G be a simple graph. A *k-total-coloring* of G is an assignment of k colors to the edges and vertices of G, so that adjacent or incident elements have different colors. The *total chromatic number* of G, denoted by $\chi_T(G)$, is the least k for which G has a k-total-coloring. Clearly, $\chi_T \geq \Delta + 1$, where Δ is the maximum degree of G, and the Total Coloring Conjecture [12] states that $\chi_T \leq \Delta + 2$. This has been proved for cubic graphs [10], so the total chromatic number of a cubic graph is either 4 or 5. Graphs with $\chi_T = \Delta + 1$ are said to be *Type 1*, and graphs with $\chi_T = \Delta + 2$ are said to be *Type 2*. The problem of deciding whether a graph is Type 1 has been shown NP-complete even for cubic bipartite graphs [9].

A graph G *contains a square* if it has a chordless cycle on four vertices as an induced subgraph and a cubic graph is *cyclically* 4*-edge-connected* if every edge-cutset of cardinality less than 4 consists of three edges incident to one vertex. So a cyclically 4-edge-connected cubic graph may contain squares, but it does not contain triangles unless it is the complete graph on four vertices. The *girth* of a graph is the length of a shortest cycle contained in the graph.

Snarks are cyclically 4-edge-connected cubic graphs with chromatic index 4. Their study was motivated by the Four Color Problem. The importance of these graphs arises mainly from the fact that several conjectures would have snarks as minimal counterexamples, such as Tutte's 5-Flow Conjecture, the 1-Factor Double Cover Conjecture, and the Cycle Double Cover Conjecture.

In 2003, Cavicchioli et al. [5] reported that their extensive computer study of snarks showed that all snarks with girth greater than 4 and with less than 30 vertices are Type 1, and asked for the smallest order of a Type 2 snark with such girth. Brinkmann et al. [1] have recently

*Partially supported by CNPq, CAPES, and FAPERJ.

extended the computer study up to order 36. In 2011, it was proved that all members of the two infinite families of Flower and Goldberg snarks are Type 1 [4]; recently, all members of the two additional infinite families of Blanuša and Loupekhine snarks have been proved to be Type 1 [11]; all graphs of these four families are square-free.

Type 2 snarks have very recently been found, but so far every known Type 2 snark contains a square [2]. On the other hand, since 1988 and even considering bipartite graphs, Type 2 cubic graphs were known [6], and all of them contain a square or a triangle. In fact, all Type 2 cubic graphs that we know (whatever their chromatic index or cyclic-edge-connectivity) have triangles or squares. So it could be that there exists no Type 2 cubic graph of girth greater than 4. Then, we consider Question 1, which is a relaxation of the question proposed in [5].

Question 1. *[2] Does there exist a Type 2 cubic graph G of girth greater than 4?*

A k-total coloring is *equitable* if the cardinalities of any two color classes differ by at most one. The least k for which G has an equitable k-total coloring is the *equitable total chromatic number* of G, denoted by $\chi_e(G)$. In [13] it was conjectured that $\chi_e(G) \leq \Delta + 2$ for any graph G, and this conjecture was proved for cubic graphs in the same work. Until now, every known Type 1 graph such that the total chromatic number is strictly less than the equitable total chromatic number was not cubic [7]. We present in Figure 1 the first known Type 1 cubic graphs such that $\chi_e(G) = 5$; notice that the colors of the vertices are not indicated as they are easily deduced from the colors of incident edges. Note that these graphs contain squares or triangles. The equitable total chromatic number of any Flower snark is known to be 4 [4]. We start to study this more restrictive type of total coloring, proposing Question 2.

Question 2. *Does there exist a Type 1 cubic graph G of girth greater than 4 such that $\chi_e(G) = 5$?*

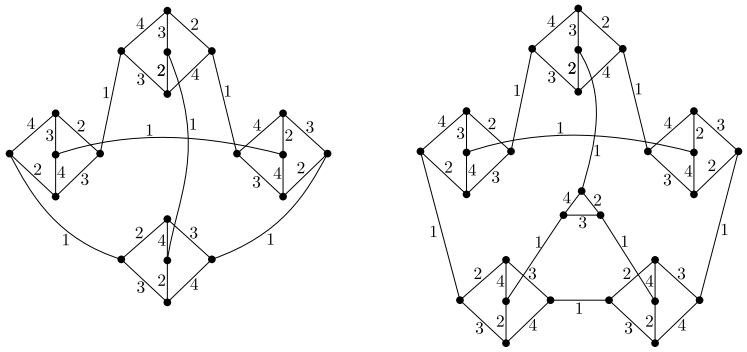


Figure 1: Type 1 cubic graphs of small girth such that $\chi_e(G) = 5$.

A sufficient condition for a cubic graph to be Type 2 is presented in this work, contributing in the search for an answer to Question 1. In addition, we contribute to Questions 1 and 2 by exhibiting infinite families that indicate that possibly both questions would have a **No** answer, or at least provide further evidences that a **Yes** answer would require, for both questions, a very large graph.

2 Generalized Petersen graphs

We consider the well-known generalized Petersen graphs $G(n,k)$. Graph $G(n,k)$ has vertex set $V = \{u_0, u_1, ..., u_{n-1}, v_0, v_1, ..., v_{n-1}\}$ and edge set consistingof all pairs of the form (v_i, v_{i+1}), (v_i, u_i), and (u_i, u_{i+k}), for $i = 0, ..., n-1$, where subscripts are read modulo n and $k < \frac{n}{2}$. Graph $G(5,2)$ is the Petersen graph that is Type 1 and the unique of chromatic index 4 among these graphs [3]. Graphs $G(n,1)$ are the n-Prisms that are Type 1 except for the graph $G(5,1)$ [6].

We have verified by a computer search[1] that the generalized Petersen graphs of order up to 70 are Type 1 with only two exceptions: $G(5,1)$ which has squares and $G(9,3)$ which has triangles; and except for these two graphs, all generalized Petersen graphs of order up to 40 have equitable total chromatic number 4. Furthermore, we prove the following results:

Theorem 1. *For every $n \geq 5$, the generalized Petersen graph $G(n,2)$ is Type 1. Furthermore, for every $n \equiv 0 \bmod 4$, the generalized Petersen graph $G(n,2)$ has equitable total chromatic number 4.* □

Theorem 2. *For every $n \geq 7$, the generalized Petersen graph $G(n,3)$ is Type 1, except for $G(9,3)$. Furthermore, for every $n \equiv 0 \bmod 4$, the generalized Petersen graph $G(n,3)$ has equitable total chromatic number 4.* □

Theorem 3. *If $G(n,k)$ is Type 1, then $G(n',k')$ is Type 1, for all $n' \equiv 0 \bmod n$ and all $k' \equiv k \bmod n$.* □

3 Snarks

First, we consider Blowup and SemiBlowup infinite families of square-free snarks recently defined by Hägglund [8]. Following the notation of [8], we define for each $n \geq 5$, the n-Blowup and the n-SemiBlowup as the Blowup$(G(n,1); C_n)$ and the SemiBlowup$(G(n,1); C_n)$, respectively, where $G(n,1)$ is the generalized Petersen graph also known as n-Prism and C_n is the outer cycle of $G(n,1)$. Let P be the Petersen graph, and let $Z_1, Z_2, ..., Z_n$ be n copies of the graph Z obtained by removing two adjacent vertices of P. According to Figure 2 where we depict the cases for $n = 5$, the n-Blowup and the n-SemiBlowup are the two cubic graphs constructed by putting $Z_1, Z_2, ..., Z_n$ along a "cycle". The smallest cases covered by Theorem 4 are depicted in Figure 2.

We remark that we are able to extend both families (by replacing the central cycle by other subgraphs) and show additional Blowup and SemiBlowup families whose graphs are Type 1.

Theorem 4. *For each $n \geq 5$, the n-Blowup and the n-SemiBlowup are Type 1.* □

Second, we consider the two Blanuša families, constructed using the dot product (a well-known operation used for constructing infinitely many snarks) of Petersen graphs starting from the two Blanuša snarks of order 18 [14]. The next result uses the 4-total-colorings determined in [11].

Theorem 5. *All snarks of Blanuša families have equitable total chromatic number 4.* □

[1] Backtracking algorithm implemented in C programming language and run in Mac OS X system over a 2.4GHz dual core processor.

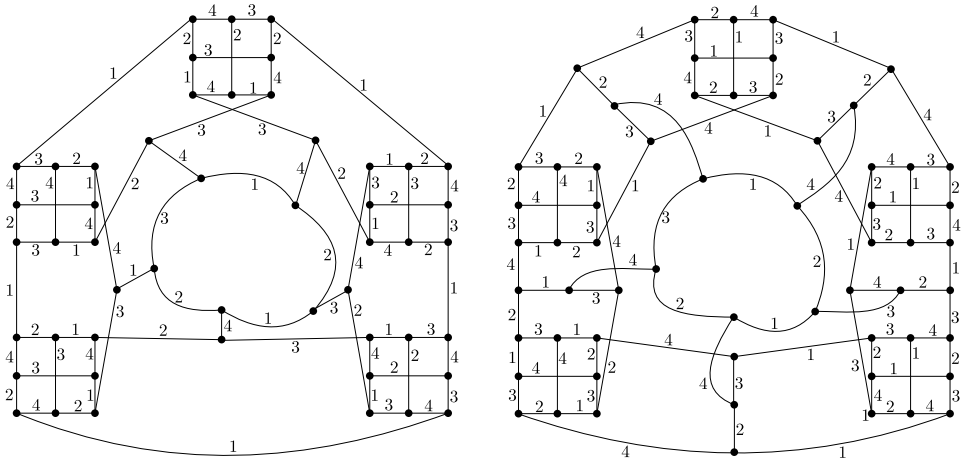


Figure 2: Theorem 4 for 5-SemiBlowup and 5-Blowup.

4 Type 2 cubic graphs

We present a sufficient condition for a cubic graph to be Type 2. We use the fact that in a 4-total-coloring of a Type 1 cubic graph G, each color class is the union of a matching M and a set of independent vertices U. The independence condition on U implies that M is a maximal matching of G. So, we have that a cubic graph is Type 1 iff its set of edges may be partitioned into 4 maximal matchings. Type 2 cubic graphs $G(5,1)$ and $G(9,3)$ satisfy the assumption of the following proposition.

Proposition 6. *If a cubic graph G has no maximal matching of size less or equal to $\frac{|E|}{4}$, then G is Type 2.*

References

[1] G. Brinkmann, J. Goedgebeur, J. Hägglund, K. Markström. Generation and properties of snarks. To appear in J. Comb. Theory B, (2011).

[2] G. Brinkmann, S. Dantas, C. M. H. de Figueiredo, M. Preissmann, D. Sasaki. Snarks with total chromatic number 5. Proc. 11^{th} CTW 2012, (2012) pp. 40–43 (full version submitted for publication).

[3] F. Castagna, G. Prins. Every Generalized Petersen Graph has a Tait Coloring. *Pacific J. Math.*, **40**, (1972) pp. 53–58.

[4] C. N. Campos, S. Dantas, C. P. Mello. The total-chromatic number of some families of snarks. *Discrete Math.*, **311**, (2011) pp. 984–988.

[5] A. Cavicchioli, T. E. Murgolo, B. Ruini, F. Spaggiari. Special classes of snarks. *Acta Appl. Math.*, **76**, (2003) pp. 57–88.

[6] A. G. Chetwynd, A. J. W. Hilton. Some refinements of the total chromatic number conjecture. *Congr. Numer.*, **66**, (1988) pp. 195–216.

[7] H. L. Fu. Some results on equalized total coloring. *Congr. Numer.*, **102**, (1994) pp 111–119.

[8] J. Hägglund. On snarks that are far from being 3-edge-colorable. *arXiv:1203.2015*, (2012).

[9] C. J. H. McDiarmid, A. Sánchez-Arroyo. Total colouring regular bipartite graphs is NP-hard. *Discrete Math.*, **124**, (1994) pp. 155–162.

[10] M. Rosenfeld. On the total coloring of certain graphs. *Israel J. Math.*, **9**, (1971) pp. 396–402.

[11] D. Sasaki, S. Dantas, C. M. H. de Figueiredo. The hunting of a snark with total chromatic number 5. Proc. LAGOS 2011, Electron. Notes Discrete Math., **37**, (2011) pp. 45–50 (full version submitted for publication).

[12] V. G. Vizing. On an estimate of the chromatic class of a p-graph. *Diskret. Analiz No.*, **3**, (1964) pp. 25–30.

[13] W. F. Wang. Equitable total coloring of graphs with maximum degree 3. *Graphs Combin.*, **18**, (2002) pp. 677–685.

[14] J. J. Watkins. On the construction of snarks. *Ars Combin.*, **16-B**, (1983) pp. 111–123.

Routing ATM Loading Vehicles

Ekrem Duman*[1] and Ahmet Altun[2]

[1]Özyeğin University, Faculty of Engineering, Industrial Engineering Department, Istanbul, Turkey, ekrem.duman@ozyegin.edu.tr

[2]Intertech, Decision Support Systems Department, Istanbul, Turkey, ahmet.altun@intertech.com.tr

1 Introduction

In this study we take up the problem of finding the routes of an ATM loading machine in multiple days. Here the owner bank of a set of ATMs wants to determine when and how much to load each ATM and what should be the route of the money vehicle so that the total cost of money holding and money loading be minimized and that a minimum service level (probability that customers will find money in the ATM) is attained. Forecasting the amount of money to be withdrawn in the future days is an important first step in solving this complicated problem. Forecasting is out of the scope of our study but it will be the main input to our optimization and routing layer. We start with a general overview of routing problems.

The vehicle routing problem (VRP) is a well-known combinatorial optimization problem where a set of customers with known demands are to be served by an homogeneous fleet of vehicles located at a central depot (Hong and Park, 1999; Toth and Vigo, 2002; Jin et al., 2007). Each vehicle starts at the depot, loaded by the commodity to be distributed to the customers, visits a number of pre-determined subset of customers and returns back to the depot after making its tour. The amount of load on a vehicle cannot exceed its capacity. The objective is to minimize the total distance (or time) traveled by all vehicles. The number of vehicles is usually unknown and determining it is a part of the solution. VRP can be regarded as a combination of two well-known NP-Hard problems: the Bin Packing Problem (BPP) and the Traveling Salesman Problem (TSP). The grouping of the customers so that the sum of the customer demands in a group does not exceed the vehicle capacity is the BPP part and then determining the visiting sequence of customers (for each vehicle) so that the distance traveled by the vehicle is minimized constitutes the TSP part.

VRP has many interesting variants. VRP with heterogeneous fleet of vehicles (VRPHF) where the vehicles are not identical in terms of capacity and/or speed (Golden et al. 1984; Taillard, 1999), VRP with time windows (VRPTW) where customers may have restrictions on when they can be visited (Tan et al., 2001; Haghani and Jung, 2005), time dependent VRP (TDVRP) where the cost (distance) of traveling between two points depends on the time of the day due to traffic rush hours etc (ref), or, VRP with pickup and delivery (VRPPD) where besides delivery there are also items to be picked up from the customers (Nagi and Salhi, 2005), are to name a few. All of these variants are NP-Hard similar to the classical VRP (Gary and Johnson, 1979; Laporte, 1992).

*Corresponding author.

Coming back to our case, we developed a four layer solution to the full ATM cash management and optimization problem as follows:

- Forecasting
  - Forecasting withdrawal amounts for the planning period
  - Inputs: past amounts withdrawn and some explanatory variables
- Optimization
  - Using the holding cost (interest that can be earned) and the money loading cost (cassette preparation and transfer costs) determine the money loading days which will minimize the total costs
  - Inputs: monthly forecast values and all cost figures
- Routing
  - First form the initial routes using the loading days obtained in the previous step
  - Improve the routes by considering changes
    - Within route
    - Between routes
    - Between days
- Daily Route Modifications
  - The actual money withdrawals might be different than the forecasts. It may be a necessity to go to an ATM before it was planned
    - When to go that ATM
    - How much to load on that ATM
    - Should other nearby ATMs be visited also
    - How the routes should be modified

We are concerned with layer 3 here. In the second layer, the optimum loading days are found by assuming fixed (route independent) loading costs. However, since loading costs are closely related to the routes of the vehicles, the solution found in layer 2 can be regarded as a rough approximate to the overall problem. Once the lists of ATMs to be loaded each day are obtained from layer 2, we can start forming the vehicle routes. However, the VRPS tackled with are different than classical VRP instances in that we do not have to go to an ATM at the stated day. It is possible to visit an ATM in an alternative day and if it brings reduction in cost it should be implemented. For example if we visit an ATM one day before the planned day the money holding cost will be more (the money loaded will be more to satisfy the demand of the extra day) but if a saving in routing can be obtained at the same amount or more than making a change in the loading day will be beneficial. Similar problems are defined in inventory routing problems literature but as for the VRP literature this is a interesting variant.

For the solution of VRP, we start with the Savings algorithm to find initial routes for each day of the planning period. Then each these routes are improved first by the Or-Opt algorithm (Or, 1976). Later, the routes of the same day are improved in total by switching nodes from one route to other or by exchanging nodes between two routes. Then, the routes of the entire

planning period are improved by switching or exchanging the nodes in the routes of different (but consecutive) days. For the example problem of Istanbul municipality the improvements obtained by each of these operators are tabulated in Table 1 and Table 2.

Table 1. Contributions of Different Operators to Solution Quality in the Same Day

Date	Phase	Phase	Route Count	Node Count	Total Length (KM)	Total Cost (TL)	Cost Saving (%)
04.07.2012	1	INIT_SOL	4	61	496,94	648,47	0
04.07.2012	2	OR_OPT	4	61	433,5	616,75	4,89
04.07.2012	3	NODE_INSERTION	4	61	402,16	601,08	7,31
04.07.2012	4	NODE_EXCHANGE	4	61	402,16	601,08	7,31
04.07.2012	5	OR_OPT_2	4	61	391,66	595,83	8,12
09.07.2012	1	INIT_SOL	1	13	154,63	177,32	0
09.07.2012	2	OR_OPT	1	13	149,01	174,51	1,58
09.07.2012	3	NODE_INSERTION	1	13	149,01	174,51	1,58
09.07.2012	4	NODE_EXCHANGE	1	13	149,01	174,51	1,58
09.07.2012	5	OR_OPT_2	1	13	149,01	174,51	1,58
10.07.2012	1	INIT_SOL	4	54	460,49	630,25	0
10.07.2012	2	OR_OPT	4	54	414,91	607,45	3,62
10.07.2012	3	NODE_INSERTION	4	54	372,49	586,24	6,98
10.07.2012	4	NODE_EXCHANGE	4	54	371,03	585,52	7,1
10.07.2012	5	OR_OPT_2	4	54	367,23	583,61	7,4

Table 2. Contributions of Different Operators to Solution Quality in the Consecutive Load Days

From Date	To Date	Switched ATM	Route Saving (TL)	Change Saving (TL)	All Saving (TL)
10.07.2012	09.07.2012	ATM_1	1,83	-0,28	1,55
10.07.2012	09.07.2012	ATM_2	0,89	-0,76	0,13
10.07.2012	09.07.2012	ATM_3	1,38	-0,66	0,72
07.07.2012	06.07.2012	ATM_4	-5,61	8,54	2,93
07.07.2012	06.07.2012	ATM_5	13,24	-4,16	9,08
07.07.2012	06.07.2012	ATM_6	101,43	6,37	107,8

The main contributions of this study are twofold. First, we provide a mathematical formulation for this interesting scheduling problem that turns out to be a new variant of VRP. Because of its difficulty the formulation was even not attempted in the earlier study of Duman et al. [3]. The second main contribution is that, although they are not targeting to solve the main scheduling problem (VRPSC), new and simple heuristic methods are proposed to solve a simplified version of it. However, the heuristics suggested can also be used to solve the VRPSC due to its special structure.

References

[1] Fotakis D. and Spirakis, P., 1998. A hamiltonian approach to the assignment of non-reusable frequencies. in Proceedings of the Eighteenth Conference on the Foundations of Software Technology and Theoretical Computer Science. India: Chennai 18-29.

[2] Duman, E., Ceranoglu, A.N., and Ozcelik, M.H., 2005. A TSP (1,2) application arising in cable assembly shops. Journal of the Operational Research Society 56, 642-648.

[3] Hong, S.C., Park, Y. B., 1999. A heuristic for bi-objective vehicle routing with time window constraints. International Journal of Production Economics 62, 249-258.

[4] Duman, E, Yildirim, M.B. and Alkaya, A.F., 2008. Scheduling continuous aluminum lines. International Journal of Production Research 46, 5701-5718.

[5] Garey M.R. and Johnson, D.S., 1979. Computers and intractability: A guide to the theory of NP-Completeness. Freeman. San Francisco.

[6] Golden B.L., Assad, A.A., Levy L. and Gheysens, F.G., 1984. The fleet size and mix vehicle routing problem. Computers and Operations Research 1, 49-66.

[7] Haghani A. and Jung, S., 2005. A dynamic vehicle routing problem with time-dependent travel times. Computers and Operations Research 32, 2959-2986.

[8] Hong, S.C., Park, Y. B., 1999. A heuristic for bi-objective vehicle routing with time window constraints. International Journal of Production Economics 62, 249-258.

[9] Jin, M, Liu, K and Bowden R.O., 2007. A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. International Journal of Production Economics 105, 228-242.

[10] Laporte, G, 1992. The vehicle routing problem: An overview of exact and approximate algorithms. European Journal of Operational Research 59, 345-358.

[11] Nagy G. and Salhi S. 2005. Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. European Journal of Operational Research 162, 126-141.

[12] Taillard E.D., 1999. A heuristic column generation method for the heterogeneous fleet VRP. RAIRO Operations Research 33, 1-14.

[13] Tan, K.C. Lee L.H. and Ou, K., 2001. Artificial intelligence heuristics in solving vehicle routing problems with time window constraints. Engineering Applications of Artificial Intelligence 14, 825-837.

[14] Toth P. and Vigo D., editors. 2002. The vehicle routing problem. In: SIAM Monographs on Discrete Mathematics and Applications. Philadelphia.

Performance Guarantees for Scheduling Algorithms under Perturbed Machine Speeds*

Michael Etscheid[1]

[1]Dept. of Computer Science, University of Bonn, Germany, etscheid@cs.uni-bonn.de

We study three simple scheduling algorithms. For unrestricted machines, Brunsch et al. [3] showed that the worst-case performance guarantees of these algorithms are not robust if the job sizes are subject to random noise. However, in the case of restricted related machines the worst-case bounds turned out to be robust even in the presence of random noise. We show that if the machine speeds rather than the job sizes are perturbed, also the performance guarantees for restricted machines decrease thus yielding a stronger result.

1 Introduction

For many simple scheduling algorithms, the worst-case performance guarantees are known up to a constant factor. However, the instances used to construct lower bounds seem to be artificial and not practically relevant if there is some noise on the input. Therefore, we use the framework of smoothed analysis to identify worst-case bounds which are too pessimistic with high probability if the input is perturbed. In this section, we define the scheduling problem and introduce the framework of smoothed analysis shortly. In Section 2, we compare briefly our results with the worst-case bounds and the bounds given in [3]. In Section 3, we derive a smoothed lower bound for the jump algorithm on restricted machines.

The Scheduling Problem. Let $J = \{1, \dots, n\}$ be the set of jobs and $M = \{1, \dots, m\}$ be the set of machines on which the jobs shall be processed. Each machine $i \in M$ has a speed s_i and each job $j \in J$ has a processing requirement p_j. The speeds of the fastest and the slowest machine are denoted by $s_{\max}$ and $s_{\min}$, respectively. We consider two different environments: In the case of unrestricted machines, each job is allowed to run on every machine. In the case of restricted machines, each job $j \in J$ has a set $M_j \subseteq M$ of allowed machines. These variables form an instance I of the scheduling problem.

A function $\sigma : J \to M$ is called a schedule. The time a machine i needs to process job j is p_j/s_i if job j is allowed to run on machine i, and ∞ otherwise. Given a schedule σ for an instance I, the load of a machine i is defined as $L_i(I, \sigma) = \sum_{j \in \sigma^{-1}(i)} p_j/s_i$. The makespan is defined as $C_{\max}(I, \sigma) = \max_{i \in M} L_i(I, \sigma)$. We write $C^*_{\max}(I)$ for an optimal makespan. The goal is to minimize the makespan.

*This research was supported by ERC Starting Grant 306465 (BeyondWorstCase).

algorithm	worst-case	perturbed job sizes	perturbed machine speeds
jump	$\Theta(\sqrt{m})$ [4, 7]	$\Theta(\phi)$ [3]	$\Theta(\phi)$
lex-jump	$\Theta\left(\min\left\{\frac{\log m}{\log\log m}, \log\frac{s_{\max}}{s_{\min}}\right\}\right)$ [5]	$\Theta(\log\phi)$ [3]	$\Theta(\log\phi)$
list	$\Theta(\log m)$ [1, 4]	$\Theta(\log\phi)$ [3]	$\Theta(\log\phi)$

Table 1: Performance guarantees for unrestricted machines.

Studied Scheduling Algorithms. We study a greedy and two local search algorithms.

The *list scheduling* algorithm starts with an empty schedule. Then it iteratively assigns an unscheduled job to the machine on which it will be completed first with respect to the current partial schedule. Each schedule which can be generated this way is called a *list schedule*.

The *jump* and *lex-jump* algorithms start with an arbitrary schedule and then perform local improvement steps. In each step, a job is reassigned to a different machine where it finishes earlier. We assume here that all jobs assigned to a machine finish at the same time, which is the load of the machine. In the jump algorithm, only jobs assigned to a *critical* machine, i.e., a machine with maximal load, are allowed to be reassigned whereas the lex-jump algorithm does not have this limitation. A schedule which cannot be improved by the (lex-)jump algorithm is called *(lex-)jump optimal*.

We write $\mathrm{Jump}(I)$ for the set of all jump optimal schedules for a scheduling instance I.

Smoothed Analysis. The framework of smoothed analysis was introduced by Spielman and Teng [8] to explain the good runtime of some algorithms in practice despite a bad worst-case runtime. We use the more general model suggested by Beier and Vöcking [2]. Let $\phi \geq 1$ be a parameter for the maximum density of the noise. A *ϕ-smooth* instance $\mathcal{I}$ consists of the job sizes $p_1, \ldots, p_n$, subsets $M_j \subseteq M, j \in J$, in the case of restricted machines, the number m of machines, and density functions $f_i \colon [0,1] \to [0,\phi]$ for all machines $i \in \{1,\ldots,m\}$. Each machine speed s_i is then chosen according to the density function f_i independently of the other machine speeds. Thus, every ϕ-smooth instance is a distribution over infinitely many scheduling instances. For $\phi = 1$, this model complies with an average case analysis, whereas for $\phi \to \infty$ the smoothed analysis tends to a worst-case analysis.

2 Related work and results

Table 1 shows an overview of the worst-case and smoothed performance guarantees in the environment of unrestricted machines. We were able to reproduce the same results as Brunsch et al. [3] with perturbed machine speeds instead of perturbed job sizes. Accordingly, we get the same conclusions that the lex-jump algorithm and the list jump algorithm should perform well in practice. An interesting deduction of theirs is that the smoothed price of anarchy for routing games on parallel links is $\Theta(\log\phi)$ as well, as pure Nash equilibria can be seen as local optima according to the lex-jump algorithm. This result carries over to our smoothed model with perturbed link speeds.

As Table 2 shows, the worst-case performance bounds in the environment of restricted machines are robust against random noise on the job sizes. We calculate the expected values of these worst-case bounds to obtain the smoothed bounds in our model. As the expected speed

algorithm	worst-case	perturbed job sizes	perturbed machine speeds
jump	$\Theta\left(\sqrt{m \cdot \frac{s_{\max}}{s_{\min}}}\right)$ [6]	$\Theta\left(\sqrt{m \cdot \frac{s_{\max}}{s_{\min}}}\right)$ [3]	$\Theta\left(m\sqrt{\phi}\right)$
lex-jump	$\Theta\left(\frac{\log S}{\log\log S}\right)$ [6]	$\Omega\left(\frac{\log m}{\log\log m}\right)$ [3]	$\Theta\left(\min\left\{m, \frac{\log(m\phi)}{\log\log(m\phi)}\right\}\right)$

Table 2: Performance guarantees for restricted machines. Here, $S = \sum_{i=1}^{m} \frac{s_i}{s_{\min}}$.

of the slowest machine is in $\Theta(1/(m\phi))$, the bounds change in an intuitive way. On the other hand, we construct classes of smoothed scheduling instances showing that the resulting upper bounds are tight up to a constant factor.

3 Proof of a lower bound for the jump algorithm

We use the remaining space to prove a lower bound for the jump algorithm in the environment of restricted machines.

Lemma 3.1. *Let $m \geq 2$, $s_1, \ldots, s_m$ be drawn from $[0, 1/\phi]$. Then $\mathbf{Pr}\left[s_{\min} \in \left[\frac{1}{m\phi}, \frac{2}{m\phi}\right]\right] \geq \frac{1}{9}$.*

Proof. For a given $\alpha \in [0, 1/\phi]$, the inequality $s_{\min} > \alpha$ holds iff $s_i > \alpha$ for all $i \in \{1, \ldots, m\}$. As $\mathbf{Pr}[s_i \leq \alpha] = \alpha\phi$ and the machine speeds are drawn independently, it follows directly that $\mathbf{Pr}[s_{\min} \leq \alpha] = 1 - (1 - \alpha\phi)^m$. Therefore,

$$\mathbf{Pr}\left[s_{\min} \in \left[\frac{1}{m\phi}, \frac{2}{m\phi}\right]\right] = \left(1 - \frac{\phi}{m\phi}\right)^m - \left(1 - \frac{2\phi}{m\phi}\right)^m \geq \left(1 - \frac{1}{2}\right)^2 - e^{-2} \geq \frac{1}{9},$$

where the first inequality follows from the well-known fact that $(1 + x/m)^m$ converges to e^x for $m \to \infty$ and that the term is monotonically increasing for $m \geq -x$. □

Theorem 3.2. *For every $\phi > 4$ and $m \geq 6$, there is a ϕ-smooth instance $\mathcal{I}$ with m restricted machines and uniform job sizes such that*

$$\mathop{\mathbf{E}}_{I \sim \mathcal{I}}\left[\max_{\sigma \in Jump(I)} \frac{C_{max}(I, \sigma)}{C^*_{max}(I)}\right] = \Omega\left(m\sqrt{\phi}\right).$$

Proof. Let $k = \left\lfloor \frac{m}{3} \right\rfloor \geq 2$. The machine speeds $s_1, \ldots, s_k$ are uniformly drawn from $[0, 1/\phi]$, whereas the machine speeds $s_{k+1}, \ldots, s_{2k}$ and $s_{2k+1}, \ldots, s_m$ are drawn from $\left(\frac{1}{\sqrt{\phi}} - \frac{1}{\phi}, \frac{1}{\sqrt{\phi}}\right)$ and $\left[\frac{\phi-1}{\phi}, 1\right]$, respectively. The jobs are partitioned in two classes $J = J_1 \dot\cup J_2$ with $J_1 = \{1, \ldots, k\}$ and $|J_2| = k\left\lceil\sqrt{\phi}\right\rceil$. All jobs have size 1. Job $j \in J_1$ is only allowed to run on the machines j and $k + j$ while every job in J_2 is allowed to run on the machines $1, \ldots, 3k$.

First consider a schedule σ', which assigns each job $j \in J_1$ to machine $k + j$. The jobs in J_2 are distributed evenly over the k machines $2k + 1, \ldots, 3k$. It follows that

$$C_{\max}(I, \sigma') \leq \max\left\{\left(\frac{1}{\sqrt{\phi}} - \frac{1}{\phi}\right)^{-1}, \frac{k\left\lceil\sqrt{\phi}\right\rceil}{k} \cdot \frac{\phi}{\phi - 1}\right\} \leq \max\left\{\frac{\phi}{\sqrt{\phi} - 1}, 2\sqrt{\phi}\right\} \leq 2\sqrt{\phi},$$

where the inequalities follow from $\phi > 4$, i.e., $\sqrt{\phi} - 1 > \sqrt{\phi} - \sqrt{\phi}/2 = \sqrt{\phi}/2$ and

$$\frac{\phi}{\phi-1}\left\lceil\sqrt{\phi}\right\rceil \leq \frac{4}{3}\left(\sqrt{\phi}+\left(\left\lceil\sqrt{\phi}\right\rceil-\sqrt{\phi}\right)\right) \leq \frac{4}{3}\sqrt{\phi}+\frac{4}{3} \leq 2\sqrt{\phi}.$$

We now construct a bad jump optimal schedule σ: Let i_0 be the slowest machine. Since $\frac{1}{\phi} \leq \frac{1}{\sqrt{\phi}} - \frac{1}{\phi}$, we know that $i_0 \leq k$. Let $\mathcal{F}$ be the event that $s_{i_0} \notin \left[\frac{1}{k\phi}, \frac{2}{k\phi}\right]$. If $\mathcal{F}$ occurs, set σ to an arbitrary schedule. Otherwise assign job $i_0 \in J_1$ to machine i_0 and $\left\lceil \frac{s_{k+i_0}}{s_{i_0}} - 1 \right\rceil$ jobs from J_2 to machine $k+i_0$. Distribute the remaining jobs in the same way as in σ'. This procedure is well-defined, as $\left\lceil \frac{s_{k+i_0}}{s_{i_0}} - 1 \right\rceil \leq \frac{s_{k+i_0}}{s_{i_0}} \leq \frac{1/\sqrt{\phi}}{1/(k\phi)} = k\sqrt{\phi} \leq k\left\lceil\sqrt{\phi}\right\rceil = |J_2|$. Note that this way no further jobs are assigned to the machines i_0 and $k+i_0$ and that every other machine has not a greater load than in σ'. Because of

$$L_{k+i_0} = \frac{\left\lceil \frac{s_{k+i_0}}{s_{i_0}} - 1 \right\rceil}{s_{k+i_0}} < \frac{\frac{s_{k+i_0}}{s_{i_0}}}{s_{k+i_0}} = \frac{1}{s_{i_0}} = L_{i_0} \leq \frac{\left\lceil \frac{s_{k+i_0}}{s_{i_0}} \right\rceil}{s_{k+i_0}} = L_{k+i_0} + \frac{p_{i_0}}{s_{k+i_0}}$$

and

$$L_{i_0} = \frac{1}{s_{i_0}} \geq \frac{k\phi}{2} \geq \phi > 2\sqrt{\phi} \geq C_{\max}(I,\sigma') \geq L_i \qquad \forall i \in M \setminus \{i_0, k+i_0\},$$

machine i_0 is the only critical machine and σ is jump optimal with $C_{\max}(I,\sigma) = \frac{1}{s_{i_0}} \geq \frac{k\phi}{2}$. Therefore, $\mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\mathrm{Jump}(I)} \frac{C_{\max}(I,\sigma)}{C^*_{\max}(I)}\right] \geq 1 \cdot \mathbf{Pr}\left[\mathcal{F}\right] + \frac{k\phi/2}{2\sqrt{\phi}} \cdot \mathbf{Pr}\left[\overline{\mathcal{F}}\right] \geq \frac{k\sqrt{\phi}}{36} = \Omega\left(m\sqrt{\phi}\right)$, where we used $\mathbf{Pr}\left[\overline{\mathcal{F}}\right] \geq \frac{1}{9}$ due to Lemma 3.1. □

References

[1] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3), pp. 486-504, 1997.

[2] R. Beier, B. Vöcking. Random knapsack in expected polynomial time. *Journal of Computer and System Sciences*, 69(3), pp. 306-329, 2004.

[3] T. Brunsch, H. Röglin, C. Rutten, T. Vredeveld. Smoothed Performance Guarantees for Local Search. *Proc. of the 19th ESA*, pp. 192-206, 2011.

[4] Y. Cho, S. Sahni. Bounds for list schedules on uniform processors. *SIAM Journal on Computing*, 9, pp. 91-103, 1980.

[5] A. Czumaj, B. Vöcking. Tight bounds for worst-case equilibria. *ACM Transactions on Algorithms*, 3(1), 2007.

[6] D. Recalde, C. Rutten, P. Schuurman, T. Vredeveld. Local search performance guarantees for restricted related parallel machine scheduling. *LATIN 2010: Theoretical Informatics*, volume 6034 of *LNCS*, pp. 108-119. Springer, Berlin, 2010.

[7] P. Schuurman, T. Vredeveld. Performance Guarantees of Local Search for Multiprocessor Scheduling. *Informs Journal on Computing*, 19(1), pp. 52-63, 2007.

[8] D. A. Spielman, S. H. Teng. Smoothed Analysis of Algorithms: Why The Simplex Algorithm Usually Takes Polynomial Time. *Journal of the ACM*, 51(3), pp. 385-463, 2004.

Observation and Evolution of Finite-dimensional Markov Systems

Ulrich Faigle[1] and Alexander Schönhuth[2]

[1]University of Cologne
[2]CWI Amsterdam

1 Markov systems

A *system* $\mathfrak{S}$ is an entity that can be in one of several *states*. Let $\mathcal{S}$ be the set of states of $\mathfrak{S}$. An *n-dimensional Markov representation* is an injective map $\rho : \mathcal{S} \to \mathcal{Q}$ onto an affine hyperplane $\mathcal{Q}$ of an n-dimensional Hilbert space $\mathcal{H}$ over $\mathbb{R}$. We denote the inner product in $\mathcal{H}$ by $\langle x|y\rangle$ and assume

$$\mathcal{Q} = \{x \in \mathcal{H} \mid \tau(x) = 1\},$$

where $\tau : \mathcal{H} \to \mathbb{R}$ is a linear functional. Given the representation ρ, we identify $\mathcal{S}$ with $\mathcal{Q}$ and speak of $\mathcal{Q}$ as the collection of *(Markov) states* of $\mathfrak{S}$.

An n-dimensional Markov system $\mathcal{S}$ admits a *standard* representation $\sigma : \mathcal{S} \to Q$ into the euclidean coordinate space $\mathbb{R}^n$ with inner product

$$\langle x|y\rangle = x^T y = \sum_{i=1}^{n} x_i y_i \quad \text{for all } x^T = (x_1, \ldots, y_n), y^T = (y_1, \ldots, y_n) \in \mathbb{R}^n.$$

and, with $\mathbf{1}^T := (1, 1, \ldots, 1)$, the affine hyperplane

$$Q = \{x \in \mathbb{R}^n \mid \tau(x) = \mathbf{1}^T x = x_1 + \ldots + x_n = 1\}.$$

However, also other representations are of interest to the mathematical modeler:

1.1 Quantum Markov systems

Motivated by the classical model of m-dimensional quantum systems, consider the (complex) Hilbert space $\mathbb{C}^{m\times m}$ of complex $(m \times m)$-matrices with inner product

$$\langle C|D\rangle = \mathrm{tr}(D^* C),$$

where D^* is the conjugate transpose of D and $\mathrm{tr}(A)$ denotes the trace of a matrix A. Recall that a matrix C is *self-adjoint* (or *hermitian*) if $C = C^*$ and let $\mathcal{H}$ denote the collection of all self-adjoint $(m \times m)$-matrices C. It is not difficult to see that $\mathcal{H}$ forms a real(!) Hilbert space of dimension $n = m^2$. Letting I denote the identity matrix of $\mathbb{C}^{m\times m}$, we call the members of the hyperplane

$$\mathcal{D} = \{D \in \mathcal{H} \mid \mathrm{tr}(D) = \langle D|I\rangle = 1\}$$

Markov density matrices and refer to a system with states corresponding to Markov density matrices a *Markov quantum system*.

1.2 Quantum activity systems and quantum bits

While classical computation is based on boolean bits, quantum computation (see, *e.g.*, [8]) models activities by *quantum bits* ("*qbits*"), where one qbit has the form

$$q = \alpha|0\rangle + \beta|1\rangle \quad \text{with } \alpha, \beta \in \mathbb{C} \text{ s.t. } |\alpha|^2 + |\beta|^2 = 1.$$

The qbit q has has the interpretation that $|0\rangle$ is observed with probability $|\alpha|^2 \geq 0$ and $|1\rangle$ with probability $|\beta|^2 = 1 - |\alpha|^2 \geq 0$.

An n-dimensional quantum activity system is the n-fold tensor product $\mathcal{A} = \mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_n$ of 1-dimensional quantum activity systems $\mathcal{A}_i$. An n-dimensional quantum activity state ("*n-qbit*") is therefore of the form

$$q = \sum_{k \in \{0,1\}^n} \alpha_k |k\rangle \quad \text{with } \alpha_k \in \mathbb{C} \text{ and } \textstyle\sum_k |\alpha|^2 = 1 \tag{1}$$

and corresponds to the parameter vector $v = (\alpha_k | k \in \{0,1\}^n) \in \mathbb{C}^{2^n}$ with (squared) norm

$$\|v\|^2 = v^* v = |\alpha_1|^2 + \ldots + |\alpha_n|^2 = 1.$$

Note that an n-qbit q in the form (1) cannot directly be interpreted a Markov state in standard form. The associated matrix $Q = vv^*$ is self-adjoint with trace

$$\mathrm{tr}(vv^*) = v^* v = |\alpha_1|^2 + \ldots + |\alpha_n|^2 = 1$$

and hence a Markov density (in fact, a classical quantum density).

1.3 Pseudo-boolean functions and cooperative games

A real-valued set function $v : 2^N \to \mathbb{R}$ is a *pseudo-boolean function* (see [6]). Identifying the subsets $K \subseteq N$ with their associated boolean states $|k\rangle$, a pseudo-boolean function v can be viewed as a formal linear combination

$$v = \sum_{k \in \{0,1\}^n} \alpha_k |k\rangle$$

with the coefficients $\alpha_k = v(K)$.

From a game theoretic point of view, the pair $\Gamma = (N, v)$ is a *cooperative game* with *characteristic function* v. The parameter $v(K)$ is thought to reflect the "value" of the coalition $K \subseteq N$ in a given economic context. It is reasonable to assume that the game Γ is scaling-invariant. So we might equally well study the normalized game $(N, \tilde{v})$, where

$$\tilde{v} = \begin{cases} 0 & \text{if } v \equiv 0 \\ v/\|v\|^2 & \text{if } \|v\|^2 = \sum_{K \subseteq N} v(K)^2 \neq 0 \end{cases}$$

and think of a non-trivial cooperative game as a qbit with real coefficients.

Remark 1.1. *The* Hadamard transformation *H of a a 1-qbit is the linear transformation*

$$|k_1 \ldots k_n\rangle \mapsto H|k_1\rangle \otimes \cdots \otimes H|k_n\rangle \quad (k_1 \ldots k_n \in \{0,1\}^2). \tag{2}$$

The Hadamard coefficients *$\hat{\alpha}_k$ of v correspond to the* Banzhaf indices *(see [2]), well-known in social choice theory. (See,* e.g.*, [7] for more applications of the Hadamard transformation to social choice problems and [5] for more on interaction indices).*

2 Observables and measurements

Returning to the general Markov state model with the n-dimensional Hilbert space $\mathcal{H}$ and $\mathcal{Q} = \{v \in \mathcal{H} \mid \tau(v) = 1\}$ relative to the system $\mathfrak{S}$, let us fix a particular basis $\mathcal{B} \subseteq \mathcal{Q}$.

Remark 2.1. *We think of $\mathcal{B}$ as the set of representatives of the "ground states" of $\mathfrak{S}$.*

We call a function $X : \mathcal{B} \to \{0, 1\}$ an *information function.* So X models a "property" ground states $b \in \mathcal{B}$ may or may not have. Extending X linearly to all of $\mathcal{H}$, X corresponds to an element $x \in \mathcal{H}$ such that

$$\langle x|b\rangle = X(b) \quad \text{for all } b \in \mathcal{B}.$$

Assume that $\mathfrak{S}$ happens to be in the Markov state $q = \sum_{b\in\mathcal{B}} q_b b$ and define

$$\pi^q(r) = \sum_{b\in\mathcal{B}:X(b)=r} q_b \quad (r = 0, 1).$$

We call X *(statistically) observable in the state* q if $\pi^q(r) \geq 0$ holds for $r = 0, 1$.

3 Evolution of Markov systems

A *Markov (evolution) operator* relative to the Markov system $\mathfrak{S}$, represented as the hyperplane $\mathcal{Q}$ of the Hilbert space $\mathcal{H}$ is a linear transformation $\mu : \mathcal{H} \to \mathcal{H}$ such that $\mu(q) \in \mathcal{Q}$ holds for all $q \in \mathcal{Q}$.

A *(generalized) Markov chain* is a pair $(\mu, q^{(0)})$ where μ is a Markov operator and q a Markov state. The pair $(\mu, q^{(0)})$ stands short for the *Markov evolution* of states in discrete time when the Markov system $\mathfrak{S}$ is in state $q^{(0)}$ at time $t = 0$:

$$q^{(t)} = \mu(q^{(t-1)}) = \mu^t(q^{(0)}) \quad \text{for } t = 1, 2, \ldots.$$

Examples of Markov chains relative to the standard representation are, of course, classical Markov chains, where μ is represented by a probability transition matrix.

Other examples arise from the Schrödinger wave evolution in quantum activity systems.

3.1 Evolution and measurement

The concept of a measurement can be naturally be put into context with evolution. We call a family $X = \{\mu_r \mid r \in \mathcal{R}\}$ of linear operators $\mu_r : \mathcal{H} \to \mathcal{H}$ a *Markov measurement* with (finite) scale $\mathcal{R}$ iff

$$\mu_X := \sum_{r\in\mathcal{R}} \mu_a \quad \text{is a Markov operator.} \tag{3}$$

In light of (3), we write (X, q) as a unifying notation for both a Markov measurement X and an associated Markov chain (μ_X, q) and refer to it as a *Markov measurement chain.* A Markov measurement chain is *invariant* if $\mu_X(q) = q$.

Now consider concatenating measurements ($w := r_1 ... r_n$)

$$\mu_w(q) := \mu_{r_n}(...(\mu_{r_1}(q))...)$$

and observe that, by multinomial expansion, $\mu_X^t = \sum_{w \in \mathcal{R}^t} \mu_w$. We call a Markov measurement chain (X, q) *(statistically) observable* iff

$$\tau(\mu_w(q)) \geq 0 \quad \text{for all} \quad w \in \mathcal{R}^*.$$

3.2 Equivalence and minimality of Markov measurements

We call two Markov measurement chains

$$X_1 = (\{\mu_r : \mathcal{H}_1 \to \mathcal{H}_1 \mid r \in \mathcal{R}\}, q_1) \quad \text{and} \quad X_2 = (\{\rho_r : \mathcal{H}_2 \to \mathcal{H}_2 \mid r \in \mathcal{R}\}, q_2)$$

where, possibly, $\dim \mathcal{H}_1 \neq \dim \mathcal{H}_2$, *equivalent* iff

$$\tau_1(\mu_{\bar{r}}(q_1)) = \tau_2(\mu_{\bar{r}}(q_2)) \quad \text{for all} \quad \bar{r} \in \mathcal{R}^* = \sum_{t \geq 0} \mathcal{R}^t.$$

We write

$$(X_1, q_1) \sim (X_2, q_2)$$

in that case.

We call a Markov measurement chain (X, q) on $\mathcal{H}$ *minimal* iff $\dim \mathcal{H}$ is minimal among all Markov measurement chains that are equivalent to (X, q). (See also [4] for details on how to perform equivalence tests efficiently.)

3.3 Decomposition of Markov measurements

We present the following new theorem:

Theorem 3.1 (Decomposition of invariant Markov measurement chains)**.** *Let* $X = (\{\mu_r : \mathcal{H} \to \mathcal{H} \mid r \in \mathcal{R}\}, q)$ *be a minimal, observable, invariant Markov measurement chain. Let* $d := \dim(\mathrm{Eig}_{\mu_X}(1)$*. Then there are minimal, observable, invariant Markov measurement chains*

$$X_i := (\{\mu_r^{(i)} : \mathcal{H}_i \to \mathcal{H}_i \mid r \in \mathcal{R}\}, q_i) \quad i = 1, ..., d$$

such that

(i) $q = q_1 + ... + q_d$

(ii) $(X, q_i) \sim (X_i, q_i)$

(iii) $\dim(\mathrm{Eig}_{\mu_{X_i}}(1)) = 1.$

(iv) $\mathcal{H} \cong \mathcal{H}_1 \otimes ... \otimes \mathcal{H}_d.$

Remark 3.2. $\dim \mathrm{Eig}_{\mu_X}(1) \geq 1$*, see [3].*

One may perceive this theorem as a building block for a unifying theory of classification for, for example, hidden Markov processes, quantum random walks and action-based cooperation systems emerging from game theory [10].

References

[1] J.-P. Aubin: *Cooperative fuzzy games*. Math. of Operations Research 6 (1981), 1-13.

[2] J.F. Banzhaf: *Weighted voting does not work: A mathematical analysis*. Rutgers Law Review 19 (1965), 317-347.

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

[4] U. Faigle and A. Schönhuth: *Efficient tests for equivalence of hidden Markov processes and quantum random walks*, IEEE Trans. Information Theory, 57 (2011), 1746-1753.

[5] M. Grabisch, J.-L. Marichal, R. Mesiar and E. Pap, *Aggregation Functions*. Encyclopedia of Mathematics and its Applications vol. 127, Cambridge, 2009.

[6] P.L. Hammer and S. Rudeanu, *Boolean Methods in Operations Research and Related Areas*. Springer-Verlag, 1968.

[7] G. Kalai: *A Fourier-theoretic perspective on the Condorcet paradox and Arrow's theorem*. Advances in Applied Mathematics 29 (2002), 412-426.

[8] A. Yu. Kitaev, A.H. Shen and M.N. Vyalyi, *Classical and Quantum Computation*, Graduate Studies in Mathematics vol. 47, American Mathematical Society, 2002.

[9] A. Schönhuth, *Discrete-valued stochastic vector spaces*. Doctoral dissertation (in German), Universität zu Köln, 2006.

[10] J. Voss, *A System-theoretic Approach to Multi-Agent Models*. Doctoral dissertation (in German), Universität zu Köln, 2012.

Optimal Cost Sharing for Capacitated Facility Location Games

Philipp von Falkenhausen*[1] and Tobias Harks[2]

[1]Technical University Berlin
[2]Maastricht University

1 Introduction

In a capacitated facility location game, there is a set N of players that each need to connect to a facility from a set M. A user i of a facility r experiences a transport cost $t_{i,r}$ for the connection to r and additionally has to pay a share of the facility's cost. Specifically, given a strategy $x_i \in M$ for every player i, we denote the joint strategy by x. The users $i \in N_r(x)$ of facility r each impose a load d_i, such that the total load is $\ell_r(x) = \sum_{i \in N_r(x)} d_i$. The facility has load-dependent cost, which we denote interchangeably by $c_r(x) = c_r(\ell_r(x))$. Each player i pays a share $\xi_i(x)$ of the cost of facility x_i and the player's goal is to choose x_i such as to minimize $\xi_i(x) + t_{i,x_i}$ given the other players choices x_{-i}. We require the cost shares of the users of a facility to sum up to the facility's cost $\sum_{i \in N_r(x)} \xi_i(x) = c_r(x)$.

Our only assumption on the cost functions is that they are nondecreasing, therefore hard capacities can be modeled by a sharp increase in cost. The restriction that each player connects to only one facility is made only to improve clarity of the exposition, our results also hold for the case where players require a connection to multiple facilities with individual matroid support constraints.

Given the facility location model (N, M, d, t, c), in an ideal world facilities are allocated optimally, that is, an allocation minimizes the social costs. In distributed systems, however, players will selfishly select facilities for their demands based on their transport cost and the cost shares they have to pay. While physical transport cost is unavoidable, the ways the monetary cost of a facility is shared among its users - the cost sharing protocol ξ - determine the Nash equilibria of the strategic game induced. A pure Nash equilibrium is a strategy profile x such that no player can unilaterally improve by switching to a different strategy $r \in M$, i.e. $\xi_i(x) + t_{i,x_i} \leq \xi_i(r, x_{-i}) + t_{i,r}$.

The main objective of this paper is to design cost sharing protocols so as to minimize the efficiency loss of the equilibria induced. We consider the *price of anarchy* and the *price of stability* as the two prevailing performance metrics used in the literature. The price of anarchy (PoA) is defined as the worst-case ratio of the cost of a Nash equilibrium over the cost of a system optimum, while the price of stability (PoS) captures the ratio of the best possible Nash equilibrium over a system optimum.

*This research was supported by the Deutsche Forschungsgemeinschaft within the research training group 'Methods for Discrete Structures' (GRK 1408).

Table 1: The results without reference are derived in this paper, $\mathcal{H}_n = \sum_{i=1}^{n} \frac{1}{i}$.

	Player-spec. Matroids		Symmetric Matroids without Delays	
	PoS	PoA	PoS	PoA
general cost	$\mathcal{H}_n$	n	$\mathcal{H}_n$ [4]	$\mathcal{H}_n$ [4]
concave cost	1	n	1	$\leq n$
convex cost	1	1	1	1

Our Results. The first main result is a characterization of Nash equilibria in games with general cost functions and player-specific strategy spaces, showing that only a subclass of strategy profiles (called *decharged*) are candidates for being Nash equilibria. This allows us to give lower bounds by constructing instances where all decharged profiles are expensive. As a second contribution, we give an algorithm that constructs decharged profiles, thus establishing PoS and PoA matching our lower bounds for general cost functions. When the class of cost functions is restricted to be either concave or convex, we finally show a drastic improvement of PoS and PoA. Note that the protocols used for the positive results also fulfill the separability requirement from [1], i.e. when assigning the cost shares on a given facility, the protocol has no information about the load on other facilities. Our characterization of Nash equilibria using the notion of decharged profiles strictly generalizes a characterization introduced in our previous paper [4]. While in [4] we assumed symmetric strategy spaces, i.e., the same strategies are available to each player, we allow in this work *player-specific strategy spaces* and, additionally, individual transport cost for each player and resource. The results in comparison to those obtained in [4] are summarized in Table 1.

Related Work. Cost sharing approaches to facility location problems and network design problems were analyzed in [2, 3]. In these works it is only required that the total cost shares cover (approximately) total cost as the players pay for the service of being connected. In contrast in our work, we require the stricter notion that the cost of every individual resource is paid for by the players using it. Closer to the model in this paper are [1], which deals with cost sharing in network games assuming constant costs.

2 General Cost Functions

Characterizing Pure Nash Equilibria. The characterization of Nash equilibria will help to identify (cheap) strategy profiles around which we build our protocols.

Theorem 2.1. *Given a facility location game, a strategy profile x can be a PNE if and only if it is* decharged*, that is*

1. *Each player i is willing to pay the transport cost t_{i,x_i}:*

$$t_{i,x_i} \leq \min_{r \in M} (c_r(r, x_{-i}) + t_{i,r}). \tag{D1}$$

2. *The users $N_r(x)$ of each facility r, after having paid their transport cost, are willing to*

share the cost of the facility:

$$c_r(x) \leq \sum_{i \in N_r(x)} \min_{s \in M} (c_s(s, x_{-i}) + t_{i,s}) - t_{i,r}. \tag{D2}$$

Proof. A simple calculation shows it follows directly from the Nash condition that any PNE x is decharged. To prove that any decharged profile x can be a PNE, we define the x-enforcing cost sharing protocol. Its intuition is that in x, the cost of each facility is shared among the users proportional to their cheapest alternatives less the transport cost and that singly deviating players pay the entire cost.

Definition 2.2 (x-Enforcing Protocol)**.** *Given a facility location model (N, M, d, t, c), define for any profile z and resource r the set of* foreign players $N_r^1(z) := N_r(z) \backslash N_r(x)$ *and assign for all $i \in N$ the cost share functions*

$$\xi_i(z) := \begin{cases} \dfrac{\min\limits_{r \in M} (c_r(r, x_{-i}) + t_{i,r}) - t_{i,x_i}}{\sum\limits_{j \in N_{x_i}(x)} \min\limits_{r \in M} (c_r(r, x_{-j}) + t_{j,r}) - t_{j,x_i}} \cdot c_{x_i}(x), \\ \qquad\qquad \text{if } N_{x_i}(z) = N_{x_i}(x) \text{ and } c_{x_i}(x) > 0, \\ c_{z_i}(z), \qquad \text{if } N_{z_i}^1(z) \neq \emptyset \text{ and } i = \min N_{z_i}^1(z), \\ c_{z_i}(z), \qquad \text{if } N_{z_i}^1(z) = \emptyset,\ N_{z_i}(z) \subset N_{z_i}(x) \text{ and } i = \min N_{z_i}(z), \\ 0, \qquad \text{else.} \end{cases}$$

To establish that x is a PNE, observe that no player can improve by unilaterally changing his strategy, because in that case he would pay the entire cost of the resource which by (D2) is more expensive (incl. transport cost) than his cost share in x. □

An Optimal Protocol for the Price of Stability. Theorem 2.1 and the x-enforcing protocol reduce the Price of Stability question to finding (if existent) a cheap decharged strategy profile. We present an algorithm that 'decharges' any strategy profile, increasing the cost in the process by at most a factor $\mathcal{H}_n = \sum_{i=1}^{n} \frac{1}{i}$.

Theorem 2.3. *Every facility location model has decharged strategy profile x at cost $C(x) \leq \mathcal{H}_n \cdot C(y)$, where y is the cost-optimal strategy profile.*

Proof. Algorithm 1 returns for every instance (N, M, d, t, c) and input profile y a decharged profile x that costs no more than $\mathcal{H}_n \cdot C(y)$.

Algorithm 1 Find decharged profile x

Input: Facility location model (N, M, d, t, c), profile y

1: **while** current strategy profile has charged facilities **do**
2: pick a *certain* player on the most expensive charged resource r
3: move this player to the cheapest alternative, leaving the remainder of the profile as is
4: **end while**

Output: Decharged profile x

The key to the algorithm's performance is to move the players in the right order. Doing so, we can guarantee that after a number of loops, the cost of the strategy profile strictly decreases with every loop. Hence, the algorithm terminates. Also, the first loops increase the cost of the profile by at most a factor of $\mathcal{H}_n$, such that $C(x) \leq \mathcal{H}_n \cdot C(y)$. □

An Optimal Protocol for the Price of Anarchy. The Price of Anarchy measures the cost of the most expensive Nash equilibrium. A simple instance (not included in this abstract) where the optimal strategy profile costs 1 and some other profile with cost n is a Nash equilibrium under any budget-balanced protocol shows that the Price of Anarchy is at least n. The x-enforcing protocol matches this lower bound.

Theorem 2.4. *The Price of Anarchy for cost sharing in facility location games is at most n.*

Proof. Given a facility location model, let x be the decharged strategy profile returned by Algorithm 1 with the optimal profile as input. Then, the x-enforcing protocol induces a game where the most expensive Nash equilibrium costs at most n times as much as the optimal profile. □

3 Concave and Convex Costs

Restricting the class of cost functions allows to design highly efficient protocols.

Theorem 3.1. *For concave cost functions, an optimal strategy profile is decharged, thus, the price of stability is one by Theorem 2.1.*

For convex cost functions, we introduce the the opt-enforcing protocol.

Definition 3.2 (opt-enforcing protocol)**.** *Given a facility location model (N, M, d, t, c), choose some optimal profile y and assign for all $i \in N$ the cost share functions*

$$\xi_i(z) := \begin{cases} d_i \cdot \dfrac{c_{z_i}(z)}{\ell_{z_i}(z)}, & \text{if } N_{z_i}(z) \subseteq N_{z_i}(y) \\ c_{z_i}(z), & \text{if } N_{z_i}(z) \not\subseteq N_{z_i}(y) \text{ and } i = \min\left(N_{z_i}(z) \setminus N_{z_i}(y)\right) \\ 0, & \text{else.} \end{cases}$$

In a game induced by this protocol, a profile is a Nash equilibrium if and only if it is socially optimal. Hence, the Price of Stability and the Price of Anarchy for games with convex cost functions is 1.

References

[1] H.-L. Chen, T. Roughgarden, and G. Valiant. Designing network protocols for good equilibria. *SIAM J. Comput.*, 39(5):1799–1832, 2010.

[2] J. Könemann, S. Leonardi, G. Schäfer, and S. van Zwam. A group-strategyproof cost sharing mechanism for the steiner forest game. *SIAM J. Comput.*, 37(5):1319–1341, 2008.

[3] M. Pál and É. Tardos. Group strategyproof mechanisms via primal-dual algorithms. In *FOCS*, pages 584–593, 2003.

[4] P. von Falkenhausen and T. Harks. Optimal cost sharing for resource selection games. *Mathematics of Operations Research*, 38(1):184–208, February 2013.

Exact and heuristic algorithms for the green vehicle routing problem

Ángel Felipe Ortega[1], M. Teresa Ortuño Sánchez[1], Gregorio Tirado Domínguez[1], and Giovanni Righini[2]

[1]Departamento de Estadística e Investigación Operativa, Universidad Complutense de Madrid
[2]Università degli Studi di Milano

1 The Green VRP

We consider a variation of the vehicle routing problem (VRP), where the fleet is made of electrical vehicles. The main distinguishing feature of this model is the need for frequent recharge. Electrical vehicles have limited autonomy: therefore a vehicle may need to recharge its batteries in order to be able to serve all the customers assigned to its route. This means that the computation of an optimal set of routes must also take into account the problem of deciding where and when to recharge each vehicle and how much. A battery recharge operation can be done in different ways with different technologies, implying different recharging time and cost. We also take into account the option of partial recharge that may be useful to save time in order to satisfy maximum route duration especially by the end of a route. On the other hand we assume that all vehicles start will fully charged batteries, because it is common to recharge electric vehicles during the night and because today technology already allows to do it with normal power supply means provided that enough time is available.

Our work elaborates on a paper by Erdogan and Miller-Hooks [1] where the problem was proposed under the name of Green VRP and tackled by a heuristic algorithm. We generalize the model presented in [1] by also considering (a) the possibility of partial recharge, (b) the availability of different recharge technologies, (c) the cost due to battery amortization.

The green vehicle routing problem (GVRP) can be formulated as follows. Let $\mathcal{G} = (\mathcal{N} \cup \mathcal{R}, \mathcal{A})$ be a given weighted undirected graph whose vertex set is the union of a set $\mathcal{N}$ of customers and a set $\mathcal{R}$ of points of recharge. One distinguished point of recharge in $\mathcal{R}$ is the depot, numbered 0, where vehicle routes start and terminate.

All customer vertices in $\mathcal{N}$ must be visited by a single vehicle; split delivery is not allowed. Each customer $i \in \mathcal{N}$ is characterized by a demand q_i (expressed in kilograms).

Vertices in $\mathcal{R}$ can be visited at any time if needed. Multiple visits to them is allowed (also simultaneously) and partial recharge is also allowed. We consider a set $\mathcal{H}$ of different technologies for battery recharge. For each technology $h \in \mathcal{H}$ we assume a given recharge speed ρ_h (expressed in Watthour per minute), and a given recharge unit cost γ_h (expressed in Euro per Watthour).

For notational convenience in the remainder we assume to split each recharge station into as many copies as the number of different recharge technologies available. The copies of a same

station are not connected with each other, so that no feasible solution is allowed to include a recharge partially made with two ore more technologies during the same visit at the same station. Therefore we can associate technologies with stations and we can replace index $h \in \mathcal{H}$ with index $i \in \mathcal{R}$ accordingly.

All vertices $i \in \mathcal{R} \cup \mathcal{N}$ are also characterized by a service time s_i (expressed in minutes). In the case of customer vertices it represents the time taken by delivery operations; in the case of recharge vertices it represents a fixed time to be spent to set up the recharge operations, independent of the amount of recharge.

Non-negative coefficients d_a and v_a are associated with each edge $a \in \mathcal{A}$, representing respectively the distance (expressed in kilometers) and the average speed (expressed in km/minutes) for traveling along a in either direction.

We consider a fleet made of V identical vehicles with given capacity Q (expressed in kilograms), and equipped with batteries of capacity B (expressed in Watthour). The energy consumption is assumed to be proportional to the distance traveled through a given coefficient π (expressed in Watthour per kilometer). The duration of each route is required not to exceed a given limit T (expressed in minutes) representing the duration of drivers' work shifts.

A feasible route is a cycle complying with the following set of constraints:

- the route must include the depot;
- the overall amount of goods delivered along the route must not be larger than the vehicle capacity Q; the amount of goods delivered is given by the sum of the demands q_i of the visited customers $i \in \mathcal{N}$;
- the total duration of the route must not exceed the total allowed duration T; the route duration is given by three terms:
  - traveling time, i.e. the sum of the terms $\frac{d_a}{v_a}$ for each edge $a \in \mathcal{A}$ in the route;
  - service time, i.e. the sum of the terms s_i for each vertex $i \in \mathcal{N} \cup \mathcal{R}$ visited along the route;
  - recharge time at the stations (which is a decision variable, owing to the possibility of partial recharge);
- the level of battery charge must be kept between 0 and B at any time, taking into account that:
  - whenever the vehicle travels along edge $a \in \mathcal{A}$ the associated energy consumption is given by πd_a;
  - the amount of energy recharged at any station $i \in \mathcal{R}$ is given by ρ_i times the variable recharge time at $i \in \mathcal{R}$.

A set of feasible routes is a feasible solution if all customers are visited once and no more than V vehicles are used.

The objective to be optimized is given by the overall recharge cost, consisting of a fixed cost and a variable cost. Since batteries allow for a limited number of recharge cycles during their operational life, we associate a fixed cost with each recharge operation; this cost, indicated by f (expressed in Euro), is given by the cost of a battery divided by the estimated number of recharge cycles after which it must be replaced. The variable cost associated with a recharge

operation at any station $i \in \mathcal{R}$ is proportional to the amount of energy stored, but it also depends on the recharge technology through the coefficient γ_i.

In the talk we outline some observations on key properties of the problem. Then we present two reformulations, indicated by A and B, that can be solved with column generation. The two master problems are equivalent in the space of their discrete variables, but the linear relaxation of master problem B is a relaxation of the linear relaxation of master problem A. Hence column generation with formulation B is likely to be faster than with formulation A and in general it provides a looser lower bound.

2 Formulation A

Formulation A is based on routes, i.e. cycles including the depot. We indicate with Ω the set of all feasible routes. We associate a binary variable x_r with each feasible route $r \in \Omega$: x_r takes value 1 if and only if route r is selected to be part of the solution. Binary coefficients y_{ir} take value 1 if and only if customer $i \in \mathcal{N}$ is visited along route $r \in \Omega$. We indicate by c_r the cost of each route $r \in \Omega$.

With these definitions and notation we obtain the following integer linear programming model:

$$\text{minimize} \sum_{r \in \Omega} c_r x_r \tag{1}$$

$$\text{s.t.} \sum_{r \in \Omega} y_{ir} x_r \geq 1 \qquad \forall i \in \mathcal{N} \tag{2}$$

$$\sum_{r \in \Omega} x_r \leq V \tag{3}$$

$$x_r \in \{0, 1\} \qquad \forall r \in \Omega. \tag{4}$$

3 Formulation B

Formulation B is based on *legs*, i.e. sequences of customers visited by the same vehicle between two recharge stations (including the depot). We indicate with Λ the set of all feasible legs and with $\Lambda_{[u,v]}$ the set of all feasible legs between nodes $u \in \mathcal{R}$ and $v \in \mathcal{R}$.

Formulation B can be interpreted as the problem of selecting a suitable set of edges on a symmetric graph $\mathcal{M}$ with a vertex subset for each recharge station (including the depot) and a weighted edge for each leg. The graph does not include self-loops, i.e. a leg cannot start and end at the same station. We assume that each station corresponds to many coincident vertices, representing different visits of the same vehicle to the same station, possibly using different recharge technologies at each visit.

For each leg, that is for each column in the master problem, we also need some additional information. Coefficients q^l indicate the amount of goods picked-up/delivered along leg l. Coefficients t^l indicate the traveling and service time spent along leg l. Coefficients e^l indicate the energy consumption along leg l. We indicate by c^l the fixed cost at the depot for each leg $l \in \Lambda_{[u,v]}$. We associate a binary variable θ_{lk} with each feasible leg $l \in \Lambda$ and each vehicle k: θ_{lk} takes value 1 if and only if leg l is selected to be part of the solution and is assigned to vehicle k. It is necessary to have as many copies of the leg variables as the number of different vehicles

in order not to allow capacities, time and energy to be traded between vehicles. Unfortunately this introduces symmetry, i.e. dual degeneracy, in the master problem.

The master problem of Formulation B reads as follows.

$$\text{minimize} \quad \sum_{l\in\Lambda}\sum_{k} c^l\theta_{lk} + \sum_{k}\sum_{u\in\mathcal{R}} \gamma_u\delta_{uk} \tag{5}$$

$$\text{s.t.} \quad \sum_{v\in\mathcal{R}\setminus\{u\}}\sum_{l\in\Lambda_{[u,v]}} \theta_{lk} = 2\omega_{uk} \qquad \forall u\in\mathcal{R},\forall k \tag{6}$$

$$\omega_{0k} = 1 \qquad \forall k \tag{7}$$

$$S.E.C. \tag{8}$$

$$\sum_{k}\sum_{l\in\Lambda} y_i^l\theta_{lk} \geq 1 \qquad \forall i\in\mathcal{N} \tag{9}$$

$$\sum_{l\in\Lambda} q^l\theta_{lk} \leq Q \qquad \forall k \tag{10}$$

$$\sum_{l\in\Lambda} t^l\theta_{lk} + \sum_{u\in\mathcal{R}} \frac{\delta_{uk}}{\rho_u} \leq T \qquad \forall k \tag{11}$$

$$\delta_{uk} \leq B\omega_{uk} \qquad \forall u\in\mathcal{R},\forall k \tag{12}$$

$$\delta_{uk} \geq \sum_{v\in\mathcal{R}}\sum_{l\in\Lambda_{[u,v]}} e^l\theta_{lk} - B \qquad \forall u\in\mathcal{R},\forall k \tag{13}$$

$$\delta_{uk} + \delta_{vk} \leq 2B - (B - e^l)\theta_{lk} \qquad \forall u,v\in\mathcal{R},\forall l\in\Lambda_{[u,v]}\forall k \tag{14}$$

$$\sum_{u\in\mathcal{R}} \delta_{uk} \geq \sum_{l\in\Lambda} e^l\theta_{lk} \qquad \forall k \tag{15}$$

$$\delta_{uk} \geq 0 \qquad \forall u\in\mathcal{R},\forall k \tag{16}$$

$$\theta_{lk} \in \{0,1\} \qquad \forall l\in\Lambda,\forall k \tag{17}$$

$$\omega_{uk} \in \{0,1\} \qquad \forall u\in\mathcal{R},\forall k. \tag{18}$$

The main problem with this formulation comes from (i) multiple visits of the same vehicle to the same station and (ii) multiple use of the same (empty) leg between two stations by the same vehicle.

In order to allow for multiple visits ω_{uk} of a same vehicle to a same station, it is necessary to duplicate each station into many vertices. In this way it is possible to impose degree 2 to each visited vertex with constraints (6); moreover constraints (13) are necessary and sufficient to correctly define feasible solutions. Unfortunately this would imply an enormous increase in the size of the master problem as well as a high degree of symmetry in it. If s distinct vertices are used for each station, s^2 distinct leg variables must be used for each leg.

On the other hand if we allow for multiple visits to the same vertex, the same variable δ_{uk} represents the overall amount of energy recharged by vehicle k at station u. In this case however constraints (13) do not guarantee feasibility any longer, i.e. they are necessary but not sufficient.

References

[1] S. Erdogan, E. Miller-Hooks, *A Green Vehicle Routing Problem*, Transportation Research Part E 48 (2012) 100-14

Fooling-sets and rank in nonzero characteristic

Mirjam Friesen[1] and Dirk Oliver Theis[2]

[1]Faculty of Mathematics, Otto von Guericke University Magdeburg, Germany
[2]Faculty of Mathematics and Computer Science, University of Tartu, Estonia

1 Introduction

An $n \times n$ matrix M over some field $\Bbbk$ is called a *fooling-set matrix* of size n if

$$M_{kk} \neq 0 \qquad \text{for all } k \text{ (its diagonal entries are all nonzero), and} \tag{1a}$$
$$M_{k,\ell}\, M_{\ell,k} = 0 \qquad \text{for all } k \neq \ell. \tag{1b}$$

Note that the definition depends only on the zero-nonzero pattern of M.

In Communication Complexity and Combinatorial Optimization, one is interested in finding a large fooling-set (sub-)matrix contained in a given matrix (permutation of rows and columns is allowed), and in upper-bounding the size of a fooling-set matrix one may hope for in terms of easily computable properties of the given matrix. (The complexity status of the decision problem for fooling-set submatrix containment is open; it is however equivalent to finding a large clique in a certain type of graphs).

Dietzfelbinger, Hromkovič, and Schnitger ([3, Thm. 1.4], or see [10, Lemma 4.15]; cf. [8, 4]) shows that the rank of a fooling-set matrix of size n is at most $\sqrt{n}$, i.e.,

$$n \leq (\mathrm{rk}_{\Bbbk}\, M)^2. \tag{2}$$

This inequality gives such an upper bound on the largest fooling-set submatrix in terms of the easily computable rank.

However, it is an open question whether the exponent on the rank in the right-hand side of (2) can be improved or not. Dietzfelbinger et al. [3, Open Problem 2] specifically asked this question for 0/1-matrices, which where of particular interest in Communication Complexity at that time. Klauck and de Wolf [8] have pointed out the importance for Communication Complexity of the question regarding general matrices.

Currently, the examples (attributed to M. Hühne in [3]) of 0/1 fooling-set matrices M with lowest rank are such that $n \approx (\mathrm{rk}_{\mathbb{Q}}\, M)^{\log_4 6}$ ($\log_4 6 = 1.292\ldots$); for general matrices, Klauck and de Wolf [8] have given examples with $n \approx (\mathrm{rk}_{\mathbb{Q}}\, S)^{\log_3 6}$ ($\log_3 6 = 1.63\ldots$).

In our paper, we settle the question for fields $\Bbbk$ of nonzero characteristic p. For a prime number p, we denote by $\mathbb{F}_p$ the finite field with p elements. We will prove the following.

Theorem 1.1. *For every prime number p, there is a family of fooling-set matrices $(M^{(t)})_{t=1,2,3,\ldots}$ of size $n^{(t)}$ over $\mathbb{F}_p$, with $n^{(t)} \to \infty$, which have the property*

$$\frac{n^{(t)}}{(\mathrm{rk}_{\mathbb{F}_p} M^{(t)})^2} \longrightarrow 1,$$

In other words, inequality (2) is tight if the characteristic of $\Bbbk$ is nonzero. It is particularly striking that not only is the exponent on the rank in inequality (2) best possible, but so is the constant (one) in front of the rank.

We note that, for characteristic $p > 2$, Dietzfelbinger et al.'s *original* question regarding 0/1-matrices remains open.

Organization of this extended abstract. In the remainder of this section we will explain some of the connections of the fooling-set vs. rank problem with Combinatorial Optimization and Graph Theory concepts. In Section 2, we will sketch the construction of the matrices $M^{(t)}$. While the rank-bound is easily verified, for the proof that these matrices are indeed fooling-set matrices, we refer to the full paper [5]. In the final section, we point to some questions which remain open.

1.1 Fooling-set matrices in other areas

While being of interest in their own right in as a minimum-rank type problem in Combinatorial Matrix Theory, fooling-set matrices are connected to other areas of Mathematics and Computer Science.

In Polytope Theory, given a polytope P, sizes of fooling-set submatrices of appropriately defined matrices provide lower bounds to the number of facets of any polytope Q which can be mapped onto P by a projective mapping ([13], cf. [4]). Similarly, in Combinatorial Optimization, sizes of fooling-set matrices are lower bounds to the minimum sizes of Linear Programs for combinatorial optimization problems ([13]). For example, it is an open question whether Edmond's matching polytope for a complete graph on n vertices admits a fooling-set matrix whose size grows quicker in n than the dimension of the polytope. Such a fooling-set matrix would yield a fairly spectacular improvement on the currently known lower bounds of sizes of Linear Programming formulations for the matching problem. See [4] for bounds based on fooling sets for a number of combinatorial optimization problems, including Bipartite Matching.

In the Polytope Theory / Combinatorial Optimization applications, we typically have $\Bbbk = \mathbb{Q}$, and the rank of the large matrix A is known. However, since the definition of a fooling-set matrix depends only on the zero-nonzero pattern, changing the field from $\mathbb{Q}$ to $\Bbbk'$ and replacing the nonzero rational entries of A by nonzero numbers in $\Bbbk'$ may yield a different rank and hence a different upper bound on the size of a fooling-set matrix.

In Computational Complexity, fooling-set matrices provide lower bounds for the communication complexity of Boolean functions (see, e.g., [10, 11, 3, 8]), and for the number of states of an automaton accepting a given language (e.g., [6]).

In Graph Theory, a fooling-set matrix (up to permutation of rows and columns) can be understood as the incidence matrix of a cross-free matching. Recall that a cross-free matching of size n is a bipartite graph H where each of the two sets in the bipartition contains n vertices and such that there is a perfect matching between the two sets which is cross-free, i.e., no two matching edges induce a C_4 subgraph in H.

Cross-free matchings are best known as a lower bound on the size of biclique coverings of graphs (e.g. [2, 7]). A *biclique covering* of a graph G is a collection of complete bipartite subgraphs of G such that each edge of G is contained in at least one of these bipartite subgraphs. If a cross-free matching of size n is contained as a subgraph in G, then at least n bicliques are needed to cover all edges of G. (For some classes of graphs, this is a sharp lower bound on the biclique covering number [2, 12]).

In Matrix Theory, the maximum size of a fooling-set sub-matrix is known under a couple of different names, e.g. as independence number [1, Lemma 2.4]), or as the intersection number. For some semirings, this number provides a lower bound for the so-called factorization rank of the matrix over the semiring.

In each of these areas, fooling-set matrices are used as lower bounds. Upon embarking on a search for a big fooling-set matrix in a large, complicated matrix A, one is interested in an *a priori* upper bound on their sizes and thus the potential usefulness of the lower bound method.

2 Construction of the matrices

We now describe the construction of our matrices. Let p be a prime number and $r \geq 2$ an integer. Define the function $f\colon \mathbb{Z} \to \mathbb{F}_p$ by the recurrence relation

$$f(k+r) = -f(k) - f(k+1) \quad \text{for all } k \in \mathbb{Z} \tag{3a}$$

and the initial conditions

$$f(0) = 1, \text{ and } f(1) = \ldots = f(r-1) = 0. \tag{3b}$$

Fix an integer $n > r$. From the sequence, we define an $(n \times n)$-matrix as follows. For ease of notation, the matrix indices are taken to be in $\{0, \ldots, n-1\} \times \{0, \ldots, n-1\}$. We let

$$M_{k,\ell} = f(k - \ell). \tag{4}$$

It is fairly easy to see that $\operatorname{rk} S \leq r$.

Lemma 2.1. *The rank of S is at most r.*

Proof. From (3a), for $k \geq r$, we deduce the equation $M_{k,\cdot} = -M_{k-r,\cdot} - M_{k-r+1,\cdot}$. Hence, each of the rows $M_{k,\cdot}$, $k \geq r$, is a linear combination of the first r rows of S. □

It can be seen that the rank is, in fact, equal to r: The top-left $r \times r$ sub-matrix is regular because it is upper-triangular with non-zeros along the diagonal.

Next, we just reduce the fooling-set property (1) to a property of the function f.

Lemma 2.2. *If*

$$f(k)f(-k) = 0 \quad \text{for all } k \in \{1, \ldots, n-1\} \tag{5}$$

then M is a fooling-set matrix.

Proof. It is clear from (3b) and (4) that $M_{j,j} = f(0) = 1$ for all $j = 0, \ldots, n-1$, so it remains to verify (1b). Since

$$M_{i,j}M_{j,i} = f(i-j)f(j-i) = f(i-j)f(-(i-j)),$$

if $f(k)f(-k) = 0$ for all $k = 1, \ldots, n-1$, then $M_{i,j}M_{j,i}$ is zero whenever $i \neq j$. This proves (1b). □

Given appropriate conditions on r and n (depending on p), this condition on f can indeed be verified. For the details, we refer to the full paper [5]. At this point, suffice it to say that, for a given positive integer t, in the notation of Theorem 1.1, we have

$$n = n^{(t)} = p^t(p^t + 1) + 1.$$

3 Conclusion

Dietzfelbinger et al.'s original question regarding the tightness of inequality (2) for 0/1-matrices remains open in characteristic $p > 2$. For these matrices, it may still be possible that the exponent on the rank in the inequality (2) can be improved.

For characteristic zero, Klauck and de Wolf [8] have given examples of fooling-set matrices of size 6^k together with $\{0, \pm 1\}$-matrices of rank 3^k with the same support ($k = 1, 2, 3, \ldots$). Thus, the exponent on the rank in inequality (2) with $\Bbbk := \mathbb{Q}$ for general matrices is at least $\log_3 6 = 1.63\ldots$, while the best known bound for 0/1-matrices is $\log_4 6 = 1.292\ldots$.

We would like to point out the possibility that, in characteristic zero, the minimum possible rank on the right hand side of inequality (2) may depend not only on the characteristic, but on the field $\Bbbk$ itself. Indeed, there are examples of zero-nonzero patterns for which the minimum rank of a matrix with that zero-nonzero pattern differs between $\Bbbk = \mathbb{Q}$ and $\Bbbk = \mathbb{R}$, see e.g. [9]. Hence, for characteristic zero, we ask the following weaker version of Dietzfelbinger et al.'s question.

Question 3.1. *Is there a field $\Bbbk$ (of characteristic zero) over which the fooling-set matrix size vs. rank inequality in* (2) *can be improved?*

As mentioned in the introduction, another problem in characteristic zero comes from polytope theory. Let P be a polytope. Let A be a matrix whose rows are indexed by the facets of P and whose columns are indexed by the vertices of P, and which satisfies $A_{F,v} = 0$, if $v \in F$, and $A_{F,v} \neq 0$, if $v \notin F$. For any fooling-set submatrix of size n of A, the following inequality follows from (2) (cf. [4]):

$$n \leq (\dim P + 1)^2. \tag{6}$$

The following variant of Dietzfelbinger et al.'s question is thus of pertinence in polytope theory and combinatorial optimization.

Question 3.2. *Can the fooling-set size vs. dimension inequality* (6) *be improved (for polytopes)?*

To our knowledge, the best known lower bound for the best possible exponent on the dimension in inequality (6) is 1. Finally, the following conjecture is, to the best of our knowledge, still open.

Conjecture 3.3. *The Fooling-Set-Submatrix problem*
Input: *Integers n, m and $m \times m$ 0/1-matrix A*
Output: *"Yes", if a fooling-set submatrix of size n of A exists, "No" otherwise.*
is NP-hard.

References

[1] Joel E. Cohen and Uriel G. Rothblum, *Nonnegative ranks, decompositions, and factorizations of nonnegative matrices*, Linear Algebra Appl. **190** (1993), 149–168. MR 1230356 (94i:15015)

[2] Milind Dawande, *A note on cross-perfect bipartite graphs*, Inform. Process. Lett. **88** (2003), no. 4, 143–147. MR 2009283 (2004g:05118)

[3] Martin Dietzfelbinger, Juraj Hromkovič, and Georg Schnitger, *A comparison of two lower-bound methods for communication complexity*, Theoret. Comput. Sci. **168** (1996), no. 1, 39–51, 19th International Symposium on Mathematical Foundations of Computer Science (Košice, 1994). MR 1424992 (98a:68068)

[4] Samuel Fiorini, Volker Kaibel, Kanstantin Pashkovich, and Dirk Oliver Theis, *Combinatorial bounds on nonnegative rank and extended formulations*, arXiv:1111.0444) (to appear in *Discrete Math.*), 2013+.

[5] Mirjam Friesen and Dirk Oliver Theis, *Fooling sets (aka cross-free matchings) and rank in nonzero characteristic*, arXiv:1208.2920, 2012+.

[6] Hermann Gruber and Markus Holzer, *Finding lower bounds for nondeterministic state complexity is hard (extended abstract)*, Developments in language theory, Lecture Notes in Comput. Sci., vol. 4036, Springer, Berlin, 2006, pp. 363–374. MR 2334484

[7] S. Jukna and A. S. Kulikov, *On covering graphs by complete bipartite subgraphs*, Discrete Math. **309** (2009), no. 10, 3399–3403. MR 2526759 (2010h:05231)

[8] Hartmut Klauck and Ronald de Wolf, *Fooling one-sided quantum protocols*, arXiv:1204.4619, 2012.

[9] Swastik Kopparty and K. P. S. Bhaskara Rao, *The minimum rank problem: a counterexample*, Linear Algebra Appl. **428** (2008), no. 7, 1761–1765. MR 2388655 (2009a:15002)

[10] Eyal Kushilevitz and Noam Nisan, *Communication complexity*, Cambridge University Press, Cambridge, 1997. MR 1426129 (98c:68074)

[11] L. Lovász and M. Saks, *Möbius functions and communication complexity*, Proc. 29th IEEE FOCS, IEEE, 1988, pp. 81–90.

[12] José A. Soto and Claudio Telha, *Jump number of two-directional orthogonal ray graphs*, Integer programming and combinatorial optimization, Lecture Notes in Comput. Sci., vol. 6655, Springer, Heidelberg, 2011, pp. 389–403. MR 2820923 (2012j:05305)

[13] Mihalis Yannakakis, *Expressing combinatorial optimization problems by linear programs*, J. Comput. System Sci. **43** (1991), no. 3, 441–466. MR 1135472 (93a:90054)

Coloring of Paths into Forests *

Giulia Galbiati[1] and Stefano Gualandi[2]

[1]Department of Industrial and System Engineering, University of Pavia, Italy
[2]Department of Mathematics, University of Pavia, Italy

1 The problem

We present a new *path coloring* problem that naturally comes into play when optimization problems that arise in the context of the Multiple Spanning Tree Protocol are solved using a Column Generation approach. Network routing protocols based on a single spanning tree have several limitations in terms of robustness and fairness, whereas those based on multiple trees allow to better distribute the demands within the network and mitigate such limitations. In [4] routing problems under the Multiple Spanning Tree Protocol are presented as Integer Linear Programs and are solved using a Column Generation approach. The approach leads to solve a subproblem equivalent to the path coloring problem that is the subject of this paper and that we now present.

In graph theory *path coloring* usually refers to the problem of finding, given a graph and a collection of vertex pairs, a set of paths connecting each pair and a coloring of these paths in such a way that any two paths that share an edge receive different colors. We need here a different notion of *path coloring*.

A *path k-forest coloring* of a set P of paths of a graph $G = (V, E)$ is a coloring of the paths with at most k colors so that paths with the same color form a forest of G. If paths with the same color share an edge we intend that no monochromatic cycle is formed. This coloring partitions P into at most k disjoint sets, each set containing paths that do not form cycles in G.

The problem that we study is the *path k-forest coloring* problem (Path-k-Forest-Color), whose *instances* consist of an undirected graph $G = (V, E)$ and a set P of paths of G and the *question* is whether there exists a *path k-forest coloring* of P.

We prove in Theorem 2.1 that Path-2-Forest-Color is NP-complete and in Theorem 2.3 that Path-k-Forest-Color is NP-complete for each $k \geq 3$, hence showing that Path-k-Forest-Color is NP-complete for each $k \geq 2$, and that coloring a set of paths of an undirected graph G with a minimum number of colors so that paths with the same color form a forest of G is NP-hard.

We note that both theorems are proved in the restricted case where the length of any path in P is at most 2. If set P does instead contain only paths of lenght 1, then the problem, which we may call in this case Edge-k-Forest-Color, is known to be solvable in polynomial time since it is a partition problem of a graphic matroid. Using matroid techniques [2] one can find

*We began investigating this topic together with Francesco Maffioli. His sudden and premature death has deprived us of his enthusiasm and support. His memory will remain in our hearts and with all those who had the fortune of knowing him.

in polynomial time a partition of the edges of the graph into a minimum number of forests, a number known as the *arboricity* of the graph. However, it has recently been proved in [1] that partitioning the edges of a graph into a minimum number of trees (call the recognition form of this problem Edge-k-Tree-Color) is instead NP-hard. This shows that the request of connectivity of the elements of the partition of the edges does make a difference and increases the complexity of the problem.

Therefore the results obtained here with paths of length at most 2 are tight, showing that in this case even if the connectivity of the elements of the partition is not requested the problem is NP-hard.

Of course it would be important to find interesting cases where Path-k-Forest-Color can be solved in polynomial time. When $k = 2$, G is a grid and all paths are of length at most 2, the problem is trivially solvable in polynomial time.

In Corollary 2.2 and Corollary 2.4 we present instead other hardness results for the problem formulated on planar graphs, always with P consisting of paths of length at most 2. The first result is for Path-3-Forest-Color, the second for Path-2-Tree-Color, where with the name Path-k-Tree-Color we refer to the Path-k-Forest-Color problem with the additional request that the paths in any element of the partition form a tree of G.

2 The results

In this section we present two theorems and two corollaries. The first theorem uses a reduction from the well-known NP-complete [3] Not-All-Equal-3Sat problem, whose *instances* consist of a set $U = \{x_1, \ldots, x_n\}$ of boolean variables, a collection $C = \{c_1, \ldots, c_m\}$ of clauses over U, with each clause having exactly 3 literals, a literal being a variable or a negated variable. The problem asks whether there exists a satisfying truth assignment for U such that each clause in C has at least one true literal and one false literal.

Theorem 2.1. Path-2-Forest-Color *is NP-complete even when P consists of paths of length at most 2.*

Proof. The reduction that we now describe, from Not-All-Equal-3Sat to Path-2-Forest-Color, is illustrated in Figure 1 on a particular instance. We let this reduction use paths of length up to 4. At the end of the proof we will show that any path of length greater than 2 can be replaced by a set of paths of length 2 without altering the conclusions of the proof.

Given an instance I of Not-All-Equal-3Sat, in the corresponding instance I' of Path-2-Forest-Color graph $G = (V, E)$ has V containing three nodes z_i, y_i, t_i, for each $i \in \{1, \ldots, n\}$, and nine nodes for each clause, three nodes "labelled" with the literals l_1, l_2, l_3 of the clause and six "inner" nodes. The inner nodes, say $a_1, b_1, a_2, b_2, a_3, b_3$ are connected in a cycle, and nodes a_i, b_i are adjacent to l_i, $i = 1, \ldots, 3$. Moreover set E contains, for each $i \in \{1, \ldots, n\}$, three edges between the three vertices z_i, y_i, z_i and an edge between t_i and every vertex labelled x_i or $\overline{x}_i$. The set P contains 9 paths for each clause; precisely, if variable x_i appears unnegated (resp., negated) in clause c where it represents label l_j, then P contains the paths $\{\{z_i, y_i\}, \{y_i, t_i\}, \{t_i, x_i\}, \{x_i, a_j\}\}$ and $\{\{z_i, y_i\}, \{y_i, t_i\}, \{t_i, x_i\}, \{x_i, b_j\}\}$ (resp., $\{\{z_i, t_i\}, \{t_i, \overline{x}_i\}\}, \{\overline{x}_i, a_j\}\}$ and $\{\{z_i, t_i\}, \{t_i, \overline{x}_i\}\}, \{\overline{x}_i, b_j\}\}$). For each clause, set P also contains the three "inner" paths of length 2, i.e. $\{\{a_1, b_1\}, \{b_1, a_2\}\}$, $\{\{a_2, b_2\}, \{b_2, a_3\}\}$ and $\{\{a_3, b_3\}, \{b_3, a_1\}\}$. Finally, P contains, for each $i \in \{1, \ldots, n\}$, the paths $\{z_i, t_i\}$ of length 1 and the paths $\{\{z_i, y_i\}, \{y_i, t_i\}\}$ of length 2. Now if there exists a satisfying truth assignment

for U such that each clause in C has at least one true and one false literal, consider a coloring of the paths that sets to blue (resp., red) all paths reaching inner vertices passing through a vertex corresponding to a literal set to true (resp., to false) by the assignment. Moreover color the paths $\{\{z_i, y_i\}, \{y_i, t_i\}\}$ and $\{z_i, t_i\}$ blue and red (resp., red and blue) if variable x_i is set true (resp., false) in U, for each $i \in \{1, \ldots, n\}$. The inner paths are colored in the only way that makes it possible to avoid monochromatic cycles in G. It is straightforward to see that all paths having the same color form a forest of G. Conversely if all paths in P can be colored with two colors so that paths with the same color do not form cycles in G, notice that all paths reaching nodes labelled with the same literal, say x_i wlog, must have the same color, otherwise a monochromatic cycle would arise coloring either path $\{z_i, t_i\}$ or path $\{\{z_i, y_i\}, \{y_i, t_i\}\}$. Notice also that, for each clause, the three inner paths cannot use the same color since this would create a monochromatic cycle and therefore the 6 paths reaching the inner nodes nodes cannot be monochromatic. Hence if we set to true the literals reached via blue paths and to false the remaining ones, the corresponding assignment for U satisfies all clauses, each clause having at least one true and one false literal.

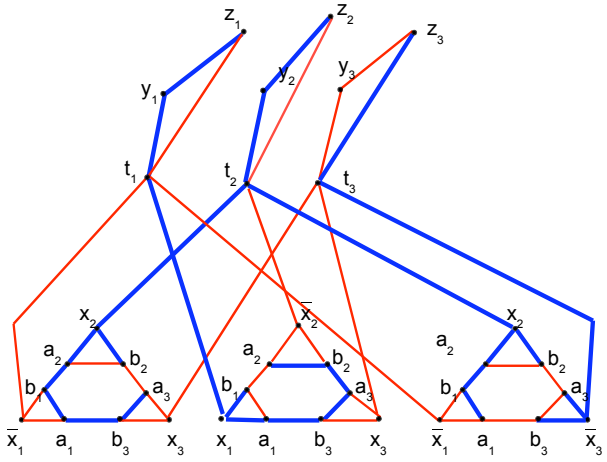


Fig 1. Graph G corresponding to an instance of NOT-ALL-EQUAL-3SAT having $C = \{\{\overline{x}_1 \vee x_2 \vee x_3\}, \{x_1 \vee \overline{x}_2 \vee x_3\}, \{\overline{x}_1 \vee x_2 \vee \overline{x}_3\}\}$ and a path-2-forest coloring corresponding to the assignment $\{1, 1, 0\}$.

In order to conclude the proof we must show that any path p of P of length 3 or 4 can be replaced by a set of paths of length 2 without altering the conclusions of the proof. Let $l \geq 3$ be the length of p and let $q_0, q_1, \ldots, q_l$ be the vertices on this path. In the reduction replace p by the paths $\{\{q_i, q_{i+1}\}, \{q_{i+1}, q_{i+2}\}\}$, $i = 0, \ldots, l-2$ and the paths $\{\{q_{i+1}, w_{i+1}\}, \{w_{i+1}, q_{i+2}\}\}$, $i = 0, \ldots, l-3$, with the w_j being new vertices. See Figure 2 for an example of path of length 4. It is easy to see that, among these paths, those that do not touch any vertex w_j must be colored with the same color in order to avoid monochromatic cycles and therefore they can be used together for reaching the same conclusions that the color of path p would have implied. □

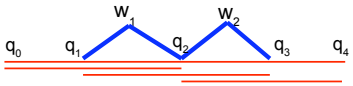


Fig 2

If in the reduction of Theorem 2.1 we contract all vertices t_i, $i = 1, \ldots, n$ into a unique node,

say t_0, then the resulting graph is planar and every set of paths with the same color forms a tree in G. Hence the following holds.

Corollary 2.2. PATH-2-TREE-COLOR *is NP-complete on planar graphs even when P consists of paths of length at most 2.*

Before addressing the next theorem we observe that EDGE-2-TREE-COLOR has recently been addressed and proved NP-complete on general graphs in [1]; unfortunately, the proof in [1] does not extend to planar graphs and, to our knowledge, the complexity of EDGE-2-TREE-COLOR on planar graphs is unknown. If EDGE-2-TREE-COLOR were solvable in polynomial time, then the result in Corollary 2.2 would be tight.

Theorem 2.3. PATH-K-FOREST-COLOR *is NP-complete for every $k \geq 3$ even when P consists of paths of length at most 2.*

Proof. (Sketch of) We reduce from VERTEX-K-COLOR, a problem asking whether or not the vertices of an undirected graph G can be colored with k colors so that adjacent vertices receive different colors. This problem is NP-complete [3] for $k \geq 3$. As in the preceding theorem we let the reduction use paths of length 3 but they could be replaced by paths of length 2. Let $G = (V = \{1, \ldots, n\}, E)$ be an instance of this problem. For each vertex u of G we add to G a vertex x_u and $k-1$ vertices $\{y_1, \ldots, y_{k-1}\}$; we also add the edges $\{x_u, y_i\}, \{y_i, u\}$, for each $i = 1, \ldots, k-1$ and the edge $\{x_u, u\}$. For each edge $e = \{u, v\}$ of G we add to G a vertex $x_{u,v}$ and the edges from this vertex to u and v. The set P of paths is defined as follows. For each vertex $u \in V$ it includes the paths $\{\{x_u, y_i\}, \{y_i, u\}\}$, for each $i = 1, \ldots, k-1$ and for each vertex $v \in V$ adjacent to u it includes the path $\{\{x_u, u\}, \{u, x_{u,v}\}, \{x_{u,v}, v\}\}$ if $u < v$, and the path $\{\{x_u, u\}, \{u, v\}\}$ otherwise. Now, given a vertex k-coloring that assigns color c to vertex u we obtain a path k-forest-coloring by assigning color c to all paths that issue from x_u and end in a vertex of $V - \{u\}$ and by assigning the remaining $k-1$ colors to the $k-1$ paths that issue from x_u and end in u. Conversely, given a path k-coloring, we observe that all paths that issue from x_u and end in a vertex of $V - \{u\}$ must have the same color, otherwise a monochromatic cycle would arise among the $k-1$ paths that issue from x_u and end in u. This induces an obvious vertex k-coloring. □

Since VERTEX 3-COLOR is NP-complete on planar graphs [3] and the reduction in Theorem 2.3 preserves the planarity of the graph the following holds.

Corollary 2.4. PATH-3-FOREST-COLOR *is NP-complete even on planar graphs and with P consisting of paths of length at most 2.*

References

[1] T Biedl and F Brandenburg. Partitions of graphs into trees. In *Graph Drawing*, vol LNCS 4372, pages 430–439. Springer, 2007.

[2] H N Gabow and H H Westermann. Forests, frames, and games: algorithms for matroid sums and applications. *Algorithmica*, 7(1):465–497, 1992.

[3] M R Gary and David S Johnson. Computers and intractability: A guide to the theory of NP-completeness, 1979.

[4] F Maffioli G Galbiati, S Gualandi. Routing with multiple spanning tree protocol: A column generation approach with partition of paths into forests, 2013. Submitted.

A linear kernel for planar red-blue dominating set *

Valentin Garnero[1], Ignasi Sau[1], and Dimitrios M. Thilikos[1,2]

[1]AlGCo project-team, CNRS, LIRMM, Montpellier, France.
[2]Department of Mathematics, National & Kapodistrian University of Athens, Greece.

In the Red-Blue Dominating Set problem, we are given a bipartite graph $G = (V_B \cup V_R, E)$ and an integer k, and asked whether G has a subset $D \subseteq V_B$ of at most k 'blue' vertices such that each 'red' vertex from V_R is adjacent to a vertex in D. We provide the first explicit linear kernel for this problem on planar graphs.

Keywords: parameterized complexity, planar graphs, linear kernels, domination.

1 Introduction

The field of parameterized complexity (see [4]) deals with algorithms for decision problems whose instances consist of a pair (x, k), where k is known as the *parameter*. A fundamental concept in this area is that of *kernelization*. A kernelization algorithm, or *kernel*, for a parameterized problem takes an instance (x, k) of the problem and, in time polynomial in $|x| + k$, outputs an equivalent instance (x', k') such that $|x'|, k' \leq g(k)$ for some function g. The function g is called the *size* of the kernel and may be viewed as a measure of the "compressibility" of a problem using polynomial-time preprocessing rules. A natural problem in this context is to find polynomial or linear kernels for problems that admit such kernelization algorithms.

A celebrated result in this area is the linear kernel for Dominating Set on planar graphs by Alber *et al.* [2], which gave rise to an explosion of (meta-)results on linear kernels on planar graphs [8] and other sparse graph classes [3, 5, 9]. Although of great theoretical importance, these meta-theorems have two important drawbacks from a practical point of view. On the one hand, these results rely on a problem property called *Finite Integer Index*, which guarantees the *existence* of a linear kernel, but it is still not yet clear how and when such a kernel can be effectively *constructed*. On the other hand, at the price of generality one cannot hope that general results of this type may directly provide explicit reduction rules and small constants for particular graph problems.

In this article we follow this research avenue and focus on the Red-Blue Dominating Set problem (RBDS for short) on planar graphs. In the Red-Blue Dominating Set problem, we are given a bipartite graph $G = (V_B \cup V_R, E)$ and an integer k, and asked whether G has

*This work was supported by the ANR project AGAPE (ANR-09-BLAN-0159) and the Languedoc-Roussillon Project "Chercheur d'avenir" KERNEL. The third author was co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: "Thales. Investing in knowledge society through the European Social Fund.

a subset $D \subseteq V_B$ of at most k 'blue' vertices such that each 'red' vertex from V_R is adjacent to a vertex in D. From a (classical) complexity point of view, finding a red-blue dominating set of minimum size is NP-complete on planar graphs [1]. From a parameterized complexity perspective, RBDS parameterized by the size of the solution is $W[2]$-complete on general graphs and FPT on planar graphs [4].

The fact that RBDS involves a *coloring* of the vertices of the input graph makes it unclear how to make the problem fit into the general frameworks of [3, 5, 8, 9]. In this article we provide the first explicit (and quite simple) polynomial-time data reduction rules for Red-Blue Dominating Set on planar graphs, which lead to a linear kernel for the problem.

Theorem 1. Red-Blue Dominating Set *parameterized by the solution size has a linear kernel on planar graphs. More precisely, there exists a poly-time algorithm that for each positive planar instance (G, k) returns an equivalence instance (G', k) such that $|V(G')| \leq 48 \cdot k$.*

This result complements several explicit linear kernels on planar graphs for other domination problems such as Dominating Set [2], Edge Dominating Set [8], Efficient Dominating Set [8], Connected Dominating Set [7, 11], or Total Dominating Set [6]. We stress that our constant is considerable smaller than most of the constants provided by these results. Since one can easily reduce the Face Cover problem on a planar graph to RBDS (without changing the parameter)[1], the result of Theorem 1 also provides a linear *bikernel* for Face Cover (i.e., a polynomial-time algorithm that given an input of Face Cover, outputs an equivalent instance of RBDS with a graph whose size is linear in k). To the best of our knowledge, the best existing kernel for Face Cover is quadratic [10]. Our techniques are much inspired from those of Alber *et al.* [2] for Dominating Set, although our reduction rules and analysis are slightly simpler.

2 A linear kernel for planar red-blue dominating set

We first propose several reduction rules and then we analyze the size of the obtained graph.

Reduction rules. We start with an elementary rule that turns out to be helpful in simplifying the instance, and then we present the rules for a single vertex and a pair of vertices. For simplicity, we will use the shorthand rbds to denote a red-blue dominating set in a graph.

Rule 1. *Iteratively remove blue vertices whose neighborhood is included into the neighborhood of another blue vertex. Similarly, remove red vertices whose neighborhood includes the neighborhood of another red vertex.*

Definition 1. *Let $G = (V_B \cup V_R, E)$ be a graph. The* neighborhood *of a vertex $v \in V_B \cup V_R$ is the set $N(v) = \{u : \{v, u\} \in E\}$. The* private neighborhood *of a blue vertex b is the set $P(b) = \{r \in N(b) : N(N(r)) \subseteq N(b)\}$.*

Rule 2. *Let $b \in V_B$ be a blue vertex. If $|P(b)| > 1$, remove $P(b)$ from G and add a new red vertex r and the edge $\{b, r\}$.*

Definition 2. *Let $G = (V_B \cup V_R, E)$ be a graph. The* neighborhood *of a blue pair of vertices $b, c \in V_B$ is the set $N(b, c) = N(b) \cup N(c)$. The* private neighborhood *of a blue pair of vertices $b, c \in V_B$ is the set $P(b, c) = \{r \in N(b, c) : N(N(r)) \subseteq N(b, c)\}$.*

[1] Just consider the *radial graph* corresponding to the input graph G and its dual G^*, and color the vertices of G (resp. G^*) as red (resp. blue).

Rule 3. *Let b, c be two distinct blue vertices. If $|P(b,c)| > 2$ and there is no blue vertex $d \neq b, c$ which dominates $P(b,c)$:*

1. *if $P(b,c) \not\subseteq N(b)$ and $P(b,c) \not\subseteq N(c)$:*
   - *remove $P(b,c)$ from G,*
   - *add two new red vertices r_b, r_c and the edges $\{b, r_b\}, \{c, r_c\}$;*
2. *if $P(b,c) \subseteq N(b)$ and $P(b,c) \subseteq N(c)$:*
   - *remove $P(b,c)$ from G,*
   - *add a new red vertex r and the edges $\{b, r\}, \{c, r\}$;*
3. *if $P(b,c) \subseteq N(b)$ and $P(b,c) \not\subseteq N(c)$:*
   - *remove $P(b,c)$ from G,*
   - *add a new red vertex r and the edge $\{b, r\}$;*
4. *if $P(b,c) \not\subseteq N(b)$ and $P(b,c) \subseteq N(c)$:*
   - *symmetrically to Case 3.*

Lemma 1. $[\star]$[2] *Let $G = (V_B \cup V_R, E)$ be a graph. If G' is the graph obtained from G by the application of Rules 1, 2, or 3, then there is a rbds in G of size at most k if and only if there is one in G'.*

Analysis of the kernel size. We will show that a graph *reduced* under our rules (i.e., a graph for which none of the rules can be applied anymore) has size linear in $|D|$, the size of a solution. To this aim we assume that the graph is *plane* (that is, given with a fixed embedding) and we will define a notion of region adapted to our definition of neighborhood. Then we will show that, given a solution D, there is a maximal region decomposition $\Re$ such that:

- $\Re$ has $O(|D|)$ regions,
- $\Re$ covers all vertices but $O(|D|)$ of them,
- each region of $\Re$ has size $O(1)$.

The three following propositions treat respectively each of the above claims.

Definition 3. *Let $G = (V_B \cup V_R, E)$ be a plane graph and let $v, w \in V_B$. A* region *$R(v,w)$ between v and w is a closed subset of the plane such that:*

- *the boundary of $R(v,w)$ is formed by two simple paths connecting v and w, each of them having at most 4 edges;*
- *all vertices (strictly) inside $R(v,w)$ belong to $N(v,w)$ or $N(N(v,w))$.*

Definition 4. *Let $G = (V_B \cup V_R, E)$ be a plane graph and let $D \subseteq V_B$. A* D-decomposition *of G is a set of regions $\Re$ between pairs of vertices in D such that:*

- *any region between v,w does not contain vertices in $D \setminus \{v, w\}$;*
- *any two regions have only the boundary in common.*

We note $V(\Re) = \bigcup_{R \in \Re} V(R)$. A D-decomposition is maximal *if there is no region $R \notin \Re$ such that $\Re \cup \{R\}$ is a D-decomposition with $V(\Re) \subsetneq V(\Re \cup \{R\})$.*

[2] The proofs of the results marked with '$[\star]$' are omitted in this extended abstract.

Proposition 1. [⋆] *Let G be a reduced plane graph and let D be a rbds in G. There is a maximal D-decomposition of G such that $|\Re| \leq 3 \cdot |D| - 6$.*

Proposition 2. [⋆] *Let $G = (V_B \cup V_R, E)$ be a reduced plane graph and let D be a rbds in G. If $\Re$ is a maximal D-decomposition, then $|V \setminus (V(\Re) \cup D)| \leq 2 \cdot |D|$.*

Proposition 3. [⋆] *Let $G = (V_B \cup V_R, E)$ be a reduced plane graph, let D be a rbds in G, and let $v, w \in D$. A region R between v and w contains at most 15 vertices distinct from v, w.*

We are finally ready to piece everything together and prove Theorem 1.

Proof of Theorem 1. Let G be the plane input graph and let G' be the reduced graph obtained from G. According to Lemma 1, G admits a rbds with size at most k if and only if G' admits one. It is easy to see that the same time analysis of [2] implies that our reduction rules can be applied in time $O(|V(G)|^3)$. According to Propositions 1, 2, and 3, if G' admits a rbds with size at most k, then G' has size at most $k + 15 \cdot (3k - 6) + 2k \leq 48k$. □

References

[1] J. Alber, H. Bodlaender, H. Fernau, and R. Niedermeier. Fixed parameter algorithms for planar dominating set and related problems. In *Proc. of the 7th Scandinavian Workshop on Algorithm Theory (SWAT)*, volume 1851 of *LNCS*, pages 97–110, 2000.

[2] J. Alber, M. Fellows, and R. Niedermeier. Polynomial-Time Data Reduction for Dominating Set. *Journal of the ACM*, 51(3):363–384, 2004.

[3] H. L. Bodlaender, F. V. Fomin, D. Lokshtanov, E. Penninkx, S. Saurabh, and D. M. Thilikos. (Meta) Kernelization. In *Proc. of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 629–638. IEEE Computer Society, 2009.

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

[5] F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In *Proc. of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 503–510. SIAM, 2010.

[6] V. Garnero and I. Sau. A linear kernel for planar total dominating set. Manuscript available at `arxiv.org/abs/1211.0978`, 2012.

[7] Q. Gu and N. Imani. Connectivity is not a limit for kernelization: Planar connected dominating set. In *Proc. of the 9th Latin American Symposium on Theoretical Informatics (LATIN)*, volume 6034 of *LNCS*, pages 26–37, 2010.

[8] J. Guo and R. Niedermeier. Linear problem kernels for NP-hard problems on planar graphs. In *Proc. of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 4596 of *LNCS*, pages 375–386, 2007.

[9] E. J. Kim, A. Langer, C. Paul, F. Reidl, P. Rossmanith, I. Sau, and S. Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. Manuscript available at `arxiv.org/abs/1207.0835`, to appear in *Proc. of ICALP'13*, 2012.

[10] T. Kloks, C.-M. Lee, and J. Liu. New Algorithms for k-Face Cover, k-Feedback Vertex Set, and k Disjoint Cycles on Plane and Planar Graphs. In *Proc. of 28th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 2573 of *LNCS*, pages 282–295, 2002.

[11] D. Lokshtanov, M. Mnich, and S. Saurabh. A linear kernel for planar connected dominating set. *Theoretical Computer Science*, 23(412):2536–2543, 2011.

The double projection method for some domination related parameters in Cartesian product graphs*

Ismael González Yero[1] and Amaurys Rondón Aguilar[2]

[1]Department of Mathematics, University of Cádiz, Spain
ismael.gonzalez@uca.es
[2]Filial Universitaria Municipal Media Luna, Universidad de Granma, Cuba
arondona@udg.co.cu

A set S of vertices of a graph G is a dominating set for G if every vertex of G is adjacent to at least one vertex of S. The domination number $\gamma(G)$, of G, is the minimum cardinality of a dominating set in G. Vizing's conjecture for the domination number of a Cartesian product graph $G\Box H$ of the graphs G and H, states that $\gamma(G\Box H) \geq \gamma(G)\gamma(H)$. In this sense, Clark and Suen [The Electronic Journal of Combinatorics, 2000] obtained that $2\gamma(G\Box H) \geq \gamma(G)\gamma(H)$ by using a technique which has been called after the "double projection method". This technique is also useful while investigating other domination related parameters in Cartesian product graphs. A domination parameter of G is related to those sets of vertices of a graph satisfying some domination property together with other conditions on the vertices of G. In this paper we use the double projection method to obtain some Clark and Suen bound-type results for the k-domination number, the Roman domination number and the global offensive alliance number of Cartesian product graphs.

Keywords: Domination number; k-domination number; Roman domination number; global offensive alliances; Cartesian product graphs.
AMS Subject Classification Numbers: 05C69; 05C76; 05C78.

1 Introduction

One of the most interesting open problems about domination in graphs is related to bounding the domination number of Cartesian product graphs [10]. Vizing's conjecture states that the domination number of the Cartesian product of two graphs is greater than or equal to the product of the factor graphs. Several partial results and numerous Vizing-like results related to other graph invariants have appeared in the literature. Nevertheless, only one general result, proved by Clark and Suen in [2], is known. There was proved that for every Cartesian product graph $G\Box H$ it follows, $\gamma(G\Box H) \geq \frac{\gamma(G)\gamma(H)}{2}$. A slightly improvement of this result was presented in [9] where the authors showed that $\gamma(G\Box H) \geq \frac{\gamma(G)\gamma(H)}{2} + \frac{1}{2}\min\{\gamma(G), \gamma(H)\}$. In

*The research was partially done while the first author was at University of Maribor, Slovenia, supported by "Ministerio de Educación, Cultura y Deporte", Spain, under the "Jose Castillejo" program for young researchers. Reference number: CAS12/00267.

both articles [2, 9], the authors used a similar technique, frequently called "double projection method" which is particularly useful while proving some other Vizing-like results. The article [1] surveys the most important contributions about Vizing's conjecture. Moreover, in this survey were presented some new results and, for instance, the so called double projection method was applied to claw-free graphs to obtain a better lower bound for them than in the general case. The proofs of our results are relatively similar among them, due to the technique which was used. Nevertheless, it is necessary to introduce some changes during the development of the technique in each proof, which are related to the style of the studied domination related parameter.

We begin with the establishment of the principal terminology and notation which will be used throughout the article. Hereafter $G = (V, E)$ denotes a finite simple graph. For two adjacent vertices u and v of G we use the notation $u \sim v$ and, in this case, we say that uv is an edge of G, *i.e.*, $uv \in E$. For a vertex v of G, $N(v) = \{u \in V : u \sim v\}$ denotes the set of neighbors that v has in G. $N(v)$ is called the *open neighborhood of* v and the *close neighborhood of* v is defined as $N[v] = N(v) \cup \{v\}$. For a set $D \subseteq V$, the *open neighborhood* is $N(D) = \cup_{v \in D} N(v)$ and the *closed neighborhood* is $N[D] = N(D) \cup D$. The maximum degree of G is denoted by Δ. A set D is a *dominating set* if $N[D] = V$. The *domination number* $\gamma(G)$ is the minimum cardinality of a dominating set in G. We say that a set S is a $\gamma(G)$-set if it is a dominating set and $|S| = \gamma(G)$.

We recall that given two graphs G and H with set of vertices $V_1 = \{v_1, v_2, ..., v_{n_1}\}$ and $V_2 = \{u_1, u_2, ..., u_{n_2}\}$, respectively, the Cartesian product of G and H is the graph $G\Box H = (V, E)$, where $V = V_1 \times V_2$ and two vertices $(v_i, u_j), (v_k, u_l)$ are adjacent in $G\Box H$ if and only if ($v_i = v_k$ and $u_j \sim u_l$), or ($v_i \sim v_k$ and $u_j = u_l$) [6].

In order to present the results we need to introduce the following notation. Given two graphs $G = (V_1, E_1)$, $H = (V_2, E_2)$ and a set X of vertices of $G\Box H = (V, E)$, the projections of X over the graphs G and H, respectively, are the following ones.

$$P_G(X) = \{u \in V_1 : \exists u \in V_1 ; (u, v) \in X\}, \;\; P_H(X) = \{v \in V_2 : \exists u \in V_1 ; (u, v) \in X\}$$

Moreover, given a set $C \subset V_1$ of vertices of G and a vertex $v \in V_2$, a $G(C, h)$-*cell* in $G\Box H$ is the set $C^h = \{(u, v) \in V : (u, v) \in C \times \{v\}\}$. A v-*fiber* is the copy of G corresponding to the vertices in $\{v\} \times V_2$.

2 k-domination

For any positive integer $k \leq \Delta$, the set S is a k-dominating set in G if for every vertex outside of S we have that $|N(v) \cap S| \geq k$. The k-domination number $\gamma_k(G)$, of G, is the minimum cardinality of a k-dominating set in G. We say that the set S is a $\gamma_k(G)$-set if it is a k-dominating set and $|S| = \gamma_k(G)$. k-domination in graphs was introduced first in [4]. For more terminology and notation we follow [5].

Theorem 2.1. *Let G be any graph and let H be a graph of maximum degree Δ. For any positive integer $k \leq \Delta$,*

$$\gamma_k(G\Box H) \geq \frac{\gamma(G)\gamma_k(H)}{2}.$$

Proof. Let V_1 and V_2 be the vertex sets of the graphs G and H, respectively. Let $S = \{u_1, ..., u_{\gamma(G)}\}$ be a dominating set for G. Let $\Pi = \{A_1, A_2, ..., A_{\gamma(G)}\}$ be a vertex partition of

G such that $u_i \in A_i$ and $A_i \subseteq N[u_i]$. Let $\{\Pi_1, \Pi_2, ..., \Pi_{\gamma(G)}\}$ be a vertex partition of $G\Box H$, such that $\Pi_i = A_i \times V_2$ for every $i \in \{1, ..., \gamma(G)\}$.

Let D be a $\gamma_k(G\Box H)$-dominating set. Now, for every $i \in \{1, ..., \gamma(G)\}$, let $D_i = P_H(D \cap \Pi_i)$. Suppose D_i is not a k-dominating set in H. So, there exists a vertex $w \notin D_i$ such that $|N_{D_i}(w)| < k$ (w is not k-dominated by D_i), which means that for every vertex v belonging to the $G(A_i, w)$-cell A_i^w it is satisfied that $|N_{D\cap\Pi_i}(v)| < k$. Thus, for every vertex $v \in A_i^w$ we have that v has at least one neighbor from D in the other $G(A_i, w)$-cell A_j^w, $j \neq i$, belonging to the same w-fiber. Now, if for every $i \in \{1, ..., n\}$, X_i denotes the set of vertices of H which are not k-dominated by D_i, then $D_i \cup X_i$ is a k-dominating set in H and we have that $\gamma_k(H) \leq |D_i| + |X_i|$. Hence,

$$\gamma_k(G\Box H) = |D| \geq \sum_{i=1}^{\gamma(G)} |D_i| \geq \sum_{i=1}^{\gamma(G)} (\gamma_k(H) - |X_i|) = \gamma(G)\gamma_k(H) - \sum_{i=1}^{\gamma(G)} |X_i|.$$

So, we obtain that

$$\gamma_k(G\Box H) \geq \gamma(G)\gamma_k(H) - \sum_{i=1}^{\gamma(G)} |X_i|. \tag{1}$$

On the other hand, for every $v \in V_2$, let l_v be the number of $G(A_{j_i}, v)$-cells $A_{j_1}^v, A_{j_2}^v, ..., A_{j_{l_v}}^v$ for which every vertex belonging to each $G(A_{j_i}, v)$-cell is not k-dominated by $D \cap \Pi_{j_1}, D \cap \Pi_{j_2}, ..., D \cap \Pi_{j_{l_v}}$, respectively. Now, let $Y_v = S - \{u_{j_1}, u_{j_2}, ..., u_{j_{l_v}}\}$. Notice that, as we mention before, each vertex of each one of these $G(A_{j_i}, v)$-cells has a neighbor from D not in the same $G(A_{j_i}, v)$-cells which itself belongs. Thus, they are dominated by vertices of D belonging to the same v-fiber but not to the same $G(A_{j_i}, v)$-cell. Hence,

$$\sum_{v\in V_2} l_v = \sum_{i=1}^{\gamma(G)} |X_i|. \tag{2}$$

Now, for every $v \in V_2$ let D_v be the set of vertices of D belonging to the same v-fiber. Since Y_v dominates $V_1 - \bigcup_{i=1}^{l_v} A_{j_i}^v$ and D_v dominates the union of all $G(A_{j_i}, v)$-cells $\bigcup_{i=1}^{l_v} A_{j_i}^v$ we have that $S_v = D_v \cup Y_v$ is a dominating set in G. If $l_v > |D_v|$, then we have

$$|S_v| = |D_v| + |Y_v| = |S| - l_v + |D_v| = \gamma(G) - l_v + |D_v| < \gamma(G),$$

which is a contradiction. So, we have $l_v \leq |D_v|$ and we obtain that

$$\sum_{v\in V_2} l_v \leq \sum_{v\in V_2} |D_v| = \gamma_k(G\Box H), \tag{3}$$

Thus, by (1), (2) and (3) we deduce $\gamma_k(G\Box H) \geq \gamma(G)\gamma_k(H) - \gamma_k(G\Box H)$, and the result follows. □

3 Global offensive alliances

A nonempty set $S \subseteq V$ is a *global offensive alliance* in G if $\delta_S(v) \geq \delta_{\overline{S}}(v) + 1$, for every $v \in \overline{S}$. Note that every global offensive alliance is a dominating set [7]. The *global offensive alliance number* of G, denoted by $\gamma_o(G)$, is defined as the minimum cardinality of a global offensive alliance in G. A global offensive alliance of cardinality $\gamma_o(G)$ is called a $\gamma_o(G)$-set.

Theorem 3.1. *For any graphs G and H, $\gamma_o(G\Box H) \geq \frac{1}{2}\max\{\gamma(G)\gamma_o(H), \gamma_o(G)\gamma(H)\}$.*

4 Roman domination

A map $f : V \to \{0, 1, 2\}$ is a *Roman dominating function* for G if for every vertex v with $f(v) = 0$, there exists a vertex $u \in N(v)$ such that $f(u) = 2$ [3, 8]. The *weight* of a Roman dominating function is given by $f(V) = \sum_{u \in V} f(u)$. The minimum weight of a Roman dominating function for G is called the *Roman domination number* of G and it is denoted by $\gamma_R(G)$. A function f is a $\gamma_R(G)$-function if it is a Roman dominating function and $f(V) = \gamma_R(G)$.

Theorem 4.1. *For any graphs G and H, $\gamma(G\Box H) \geq \frac{\gamma(G)\gamma_R(H)}{3}$.*

In [11] was proved that for any graphs G and H, $\gamma_R(G\Box H) \geq \gamma(G)\gamma(H)$, which leads to $\gamma(G\Box H) \geq \frac{\gamma(G)\gamma(H)}{2}$ since for any graph G, it follows that $\gamma(G) \geq \frac{\gamma_R(G)}{2}$. Notice that this bound is the same one obtained in [2]. Now, if $\gamma_R(H) > \frac{3\gamma(H)}{2}$, then the bound of Theorem 4.1 is better than the bound of [2] and [11]. There are several examples of graphs satisfying that $\gamma_R(H) > \frac{3\gamma(H)}{2}$. For instance, *Roman graphs* by definition satisfy that $\gamma_R(H) = 2\gamma(H)$. For Roman graphs Theorem 4.1 leads to the following consequence.

Corollary 4.2. *For any graphs G and any Roman graph H, $\gamma(G\Box H) \geq \frac{2\gamma(G)\gamma(H)}{3}$.*

References

[1] B. Brešar, P. Dorbec, W. Goddard, B. L. Hartnell, M. A. Henning, S. Klavžar, D. F. Rall, Vizing's conjecture: a survey and recent results, *Journal of Graph Theory*, **69** (2012) 46–76.

[2] W. E. Clark, S. Suen, An inequality related to Vizing's conjecture, *The Electronic Journal of Combinatorics* **7** (2000), no. 1, Note 4, 3 pp.

[3] E. J. Cockayne, P. A. Dreyer, S M. Hedetniemi, S. T. Hedetniemi, Roman domination in graphs, *Discrete Mathematics* **278** (1-3) (2004) 11–22.

[4] J. F. Fink, M. S. Jacobson, n-domination in graphs. In: *Graph Theory with Applications to Algorithms and Computer Science*. John Wiley & Sons, New York, 1985, 283–300.

[5] T. W. Haynes, S. T. Hedetniemi, P. J. Slater, *Fundamentals of Domination in Graphs*, Marcel Dekker, Inc. New York, 1998.

[6] R. Hammack, W. Imrich, S. Klavžar. *Handbook of product graphs*. CRC press, Taylor & Francis Group. Boca Raton, Fl, USA. Second Edition. 2011.

[7] P. Kristiansen, S. M. Hedetniemi, S. T. Hedetniemi, Alliances in graphs, *Journal of Combinatorial Mathematic and Combinatorial Computing* **48** (2004) 157–177.

[8] I. Stewart, Defend the Roman Empire, *Scientific American*, December (1999) 136–138.

[9] S. Suen, J. Tarr, An improved inequality related to Vizing's conjecture, *The Electronic Journal of Combinatorics* **19** (2012) #P8.

[10] V. G. Vizing, The Cartesian product of graphs, *Vyčisl. Sistemy* **9** (1963) 30–43.

[11] Y. Wu, An Improvement on Vizing's conjecture, manuscript. *arxiv.org/pdf/0909.3695.pdf*

Two-Dimensional Optimal Mechanism Design for a Single Machine Scheduling Problem

Ruben Hoeksma[1] and Marc Uetz[1]

[1] r.p.hoeksma@utwente.nl, m.uetz@utwente.nl. Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands.

We consider an optimal mechanism design problem with two-dimensional private data for a non-auction problem, namely for the classic problem of scheduling n jobs with weights w_j and processing times p_j on a single machine. The same problem with single dimensional private data was addressed before by Heydenreich et al. [4]. In line with Myerson's seminal result on optimal auctions for selling a single item [6], they show that the optimal mechanism is Smith's rule with respect to "virtual" weights. In particular, it can be computed and implemented in polynomial time. Indeed, it is fair to say that optimal mechanism design with single dimensional private data is pretty well understood, while algorithmic results for optimal mechanism design with multi dimensional private data have been obtained only recently, e.g. [1, 2]. For single machine scheduling, Heydenreich et al. leave it as an open problem "to identify (closed formulae for) optimal mechanisms for the 2-d case" [4]. We solve this problem, as our main result is:

Theorem 1 ([5])**.** *An optimal randomized mechanism for the single machine mechanism design problem with two-dimensional private data can be computed and implemented in polynomial time.*

Several technical contributions form the basis of this result: We formulate the mechanism design problem as an exponential size integer linear program based on the well known linear ordering formulation [8]. The corresponding LP relaxation represents randomized mechanisms, yet is still exponential in size. We can show that this LP can be compactified such that only a polynomial number of variables and constraints remain, without degradation of the solution. By solving the compactified LP we find the optimal objective value and so-called interim schedules. Implementing the corresponding solution finally asks for an algorithm which, for any given vector of private data, outputs a lottery over at most n different schedules. To this end we propose a geometric decomposition algorithm which, for any point in the scheduling polytope [7], computes a decomposition into a convex combination of $\leq n$ vertices, i.e., schedules.

Problem Definition and LP-compactification

The problem is defined as follows: A single machine is owned by the mechanism designer who needs to decide in which order to process the jobs. Each of the jobs has a private type, $t_j = (w_j, p_j)$, specifying cost for waiting w_j and processing requirement p_j. While the types

of the jobs are private to the job itself, we assume that each job has a discrete type space, T_j, with at most m types, and a probability distribution $\varphi_j : T_j \to [0,1]$ over that type space, public to both the mechanism designer and all jobs. By $T = T_1 \times \ldots \times T_n$ we denote the type space of all jobs, with $t = (t_1, \ldots, t_n) \in T$. Moreover, (t_j, t_{-j}) denotes a type vector where t_j is the type of job j and t_{-j} are the types of all jobs except j, with $t_{-j} \in T_{-j} := \prod_{k \neq j} T_k$. For given $t \in T$ and $K \subseteq N$, we also define the shorthand notation $\varphi(t_K) := \prod_{k \in K} \varphi_k(t_k)$ for the product distribution of the types of jobs in K, particularly $\varphi(t_{-j}) := \prod_{k \neq j} \varphi_k(t_k)$.

Each job must be scheduled on the machine and the jobs must be reimbursed for their waiting cost with a payment, π_j. The mechanism designer aims to minimize the total expected payment made to the jobs, $\sum_t \varphi(t) \sum_j \pi_j(t)$. The jobs however aim to maximize their expected utility, $\pi_j(t_j) - w_j(t_j) Es_j(t_j)$. Here $Es_j(t_j)$ represents the expected starting time of job j given that it has type t_j, the expectation taken over all possible types of the other jobs. The jobs may of course lie about their types to improve their expected utility. However, Meyerson's revelation principle states that w.l.o.g. we may restrict to direct mechanisms where the equilibrium strategies of all jobs are to be truthful. Then we obtain the following LP-formulation for optimal randomized mechanisms in the standard Bayes-Nash setting.

$$\min \sum_{j \in N} \sum_{i \in T_j} \varphi_j^i \pi_j^i \tag{1}$$

$$\pi_j^i \geq w_j^i Es_j^i \qquad \forall j, i \tag{2}$$

$$\pi_j^i \geq \pi_j^{i'} - w_j^i (Es_j^{i'} - Es_j^i) \qquad \forall j, i, i' \tag{3}$$

$$Es_j^i = \sum_{t_{-j} \in T_{-j}} \varphi(t_{-j}) \sum_{k \in N} d_{kj}(t_j^i, t_{-j}) p_k(t_j^i, t_{-j}) \qquad \forall j, i \tag{4}$$

$$d_{jj}(t) = 0 \qquad \forall j, t \tag{5}$$

$$d_{kj}(t) + d_{jk}(t) = 1 \qquad \forall j, k, t \; j \neq k \tag{6}$$

$$d_{jk}(t) \in [0,1] \qquad \forall j, k, t \tag{7}$$

Here, we use shorthand notation φ_j^i for $\varphi_j(t_j^i)$, with type $t_j^i \in T_j$. The same for π_j^i, w_j^i, and Es_j^i. Individual rationality constraints (2) ensure that all jobs have non-negative expected utility. The Bayes-Nash incentive compatibility constraints (3) ensure that for each job it is best in expectation to be truthful. Variables $d_{kj}(t)$ are linear ordering variables for jobs k and j, given a vector of types t. The LP-formulation (1)-(7) can have $\Omega(\, m^n \,)$ variables, while the input size of the problem is $O(nm)$.

We reduce the number of variables by considering variables $d_{kj}(t_k, t_j)$ instead of $d_{kj}(t)$. This translates to choosing d_{kj} only based on the types t_j and t_k of those jobs, instead of letting d_{kj} depend on the types of all jobs. We refer to this reduction as LP-compactification and show:

Lemma 2 ([5])**.** *The optimal mechanism design problem described by* (1)-(7) *can be equivalently described by a compactified LP, where variables* $d_{kj}(t)$ *are replaced by variables* $d_{kj}(t_k, t_j)$*.*

The LP-compactification results in $O(n^2 m^2)$ variables and therefore can be solved in time polynomial in the size of the input.

Implementation

The optimal solution of the compactified LP consists of a payment and an expected starting time for each job and type of job, as well as the corresponding vector of linear ordering variables

d. To obtain the actual implementation for this so-called interim solution, for any given vector of types t we first calculate the starting times for each job

$$s_j(t) = \sum_{k \neq j} d_{kj}(t_k, t_j) p_k(t_k) \ ,$$

where t_k and t_j are the types of jobs k and j in t. The vector $s(t)$ is by construction a point in the scheduling polytope Q(t), parametrized along types t [7]. We show:

Lemma 3 ([5]). *For any point in the single machine scheduling polytope, a decomposition of this point into a convex combination of at most n schedules can be computed in $O(n^3 \log n)$ time.*

Our algorithm follows the geometric construction as proposed by Grötschel et al. in [3, Thm. 6.5.11]. Using this construction, the result follows from a combinatorial, $O(n^2 \log n)$ time algorithm to compute the maximum intersection of the scheduling polytope with a line.

References

[1] S. Alaei, H. Fu, N. Haghpanah, J. Hartline, and A. Malekian (2012). *Bayesian Optimal Auctions via Multi- to Single-agent Reduction.* In: Proc. 13th EC, 2012, p. 17

[2] Y. Cai, C. Daskalakis and S.M. Weinberg (2012). *Optimal Multi-Dimensional Mechanism Design: Reducing Revenue to Welfare Maximization.* In: Proc. 53rd FOCS, 2012, pp. 130-139.

[3] M. Grötschel, L. Lovász and A. Schrijver (1988). *Geometric algorithms and combinatorial optimization.* Algorithms and combinatorics, 1988, Springer.

[4] B. Heydenreich, D. Mishra, R. Müller and M. Uetz (2008). *Optimal Mechanisms for Single Machine Scheduling.* In: Proc. WINE 2008, LNCS 5385, 414-425.

[5] R. Hoeksma and M. Uetz (2013). *Two Dimensional Optimal Mechanism Design for a Sequencing Problem.* In: Proc. IPCO 2013, LNCS 7801, pp. 242-253.

[6] R.B. Myerson (1981). *Optimal Auction Design.* Math. OR 6, 1981, 58-73.

[7] M. Queyranne (1993). *Structure of a simple scheduling polyhedron.* Math. Prog. 58, 1993, 263-285.

[8] M. Queyranne and A.S. Schulz Polyhedral Approaches to Machine Scheduling. TU Berlin Technical Report 408/1994.

Application of the descent with mutations (DWM) metaheuristic to the computation of a median equivalence relation

Olivier Hudry[1]

[1]Telecom ParisTech, 46, rue Barrault, 75634 Paris Cedex 13, France,
olivier.hudry@telecom-paristech.fr

1 Introduction

We deal with a metaheuristic (for references on metaheuristics, see for instance [5]) called "descent with mutations" (DWM). This method looks like the usual descent, but with random elementary transformations which are performed, from time to time, in a blind way, in the sense that they are accepted whatever their effects on the function f to optimize (such an elementary transformation performed without respect to its effect on f will be called a *mutation* in the sequel). The density of performed mutations decreases during the process, so that the process at its end is the same as a classic descent.

We apply DWM to two problems, both arising from the field of classification or clustering, more precisely from the aggregation and the approximation of symmetric relations: Régnier's problem [9] and Zahn's problem [11] (see also for instance [2] for references on these problems in classification). We compare DWM with a simulated annealing method improved by ingredients coming from the noising methods (as done in [3] and in [4] for the Travelling Salesman Problem; note that Tabu search was performed in [1] and compared for these problems with simulated annealing: the results provided by these two methods are qualitatively the same).

2 Principle of DWM

As the other metaheuristics, DWM is not designed to be applicable to only one combinatorial problem, but to many of them. Such a problem can be stated as follows:

$$\text{Minimize } f(s) \text{ for } s \in S,$$

where S is a finite set and f is a function defined on S; the elements s of S will be called *solutions*. As many other metaheuristics, DWM is based on elementary transformations. A *transformation* is any operation changing a solution into another solution. A transformation will be considered as *elementary* (or *local*) if, when applied to a solution s, it changes one feature of s without modifying its global structure much. For instance, if s is a binary string, a possible elementary transformation would be to change one bit of s into its complement.

Thanks to the elementary transformations, we may define the neighbourhood $N(s)$ of a solution s: $N(s)$ is the set of all the solutions that we can obtain from s by applying an

elementary transformation to s. Then, we may define an iterative improvement method, or *descent* for a minimization problem (it is the case for the problems considered here), as follows. A descent starts with an initial solution s_0 (which can be for instance randomly computed, or found by a heuristic) and then generates a series of solution s_1, s_2, ..., s_i, ..., s_q such that:

1. for any $i \geq 1$, s_i is a neighbour of s_{i-1}: $s_i \in N(s_{i-1})$;
2. for any $i \geq 1$, s_i is better than s_{i-1} with respect to f: $f(s_i) < f(s_{i-1})$;
3. no neighbour of s_q is better than s_q: $\forall s \in N(s_q)$, $f(s) \geq f(s_q)$.

Then s_q is the solution returned by the descent. The descent is over and the final solution s_q provided by the descent is (at least) a local minimum of f with respect to the adopted elementary transformation. The whole method may stop here, or restarts a new descent from a new initial solution (to get repeated descents).

In such a descent, the process is not blind in the sense that the elementary transformations are adopted only if they improve the value taken by f. In DWM, we also apply the basic process of a descent but, from time to time, we apply and accept the considered elementary transformation, whatever its effect on f: we say that we have a *blind elementary transformation*, or simply a *mutation*. Thus, the only thing to specify in order to apply DWM (in addition to what must be defined to apply a descent, like the elementary transformation) is when a mutation is adopted.

3 Application of DWM to Régnier's and Zahn's problems

3.1 Statement of Régnier's and Zahn's problems as the partition of a graph into disjoint cliques

Régnier's problem [9], which arises for example in cluster analysis, consists in the aggregation of equivalence relations into a unique equivalence relation summarizing the initial equivalence relations as accurately as possible. More precisely, we consider a set X of n objects and a set of m criteria; each criterion is assumed to define an equivalence relation (or, equivalently, a partirion) on X; the aim is to find a unique equivalence relation defined on X which summarizes the m criteria as accurately as possible by minimizing the total number of disagreements with respect to the given criteria. This problem is NP-hard when m is not fixed [6] (its complexity is unknown in the general case when m is fixed).

Zahn's problem [11], which arises in social sciences, consists in approximating a given symmetric relation R defined on a set X by an equivalence relation E defined also on X which is at minimum distance from R with respect to the symmetric difference distance (which measures the number of disagreements between R and E, see [7]). This problem is NP-hard too, as shown by M. Krivanek and J. Moravek [8].

These two problems can be represented (see for instance [10]) by the following clique partitioning problem (CPP in what follows). In this CPP, we consider a weighted undirected graph $G = (X, U, w)$ with n vertices; G is complete; an integer (which can be positive, or negative, or equal to 0) $w(x, y) = w(y, x)$ is associated with each edge $\{x, y\} \in U (x \neq y)$; then CPP consists in finding a partition of X into $k(G)$ disjoint cliques C_1, C_2 , ..., $C_{k(G)}$ (note that the number $k(G)$ of cliques is not fixed a priori and depends on the considered graph; for this reason and because of the signs of the weights, CPP is not the famous clique partitioning

problem sometimes known as the k-cut problem, though the formulations of the two problems are near each other) in order to minimize the sum of the weights of the edges with their two extremities in a same clique, i.e., in order to minimize the function f defined for any partition $(C_1, C_2, ..., C_k)$ of X by:

$$f(C_1, C_2, ..., C_k) = \frac{1}{2} \sum_{i=1}^{k} \sum_{(x,y) \in C_i^2, x \neq y} w(x, y).$$

To formulate Régnier's problem and Zahn's problem as instances of CPP, we build a weighted complete graph as follows. The vertex set will be the set X of Régnier's or Zahn's problems. For Régnier's problem, the weight of an edge $\{x, y\}$ (with $x \neq y$) is given by the difference $m - 2m_{xy}$, where m_{xy} denotes the number of equivalence relations for which x and y are together in a same class (this weight is also the difference between the number, equal to $m - m_{xy}$, of equivalence relations for which x and y are not together in a same class, and m_{xy}). For Zahn's problem, let R be the symmetric relation of the instance; the weight of an edge $\{x, y\}$ (with $x \neq y$) is -1 if x and y are in relation with respect to R and $+1$ otherwise. Then the search of an equivalence relation which is a solution of Régnier's or Zahn's problems consists in both cases in partitioning the weighted complete graph into disjoint cliques in order to minimize the sum of the weights of the edges with their two extremities in a same clique, i.e. CPP.

3.2 Application of the descent (without mutations) to CPP

To define a descent for CPP, we apply the following elementary transformation: we choose a vertex v and we move v from its current clique into another clique or alone in a new clique; in this last case, we say that we put v in the empty clique. To apply a descent, we begin from a randomly chosen partition and we consider the vertices one after the other in a cyclic way; a neighbour better than the current solution is accepted as soon as it has been discovered. More precisely, for each vertex v and for each clique C (including the empty one) of the current solution s, we compute the variation $\Delta f(s, C)$ of f when v is moved from its current clique to C; the clique C^* for which the variation $\Delta f(s, C^*)$ is minimum is called the *best clique for* v (with respect to s); then, if moving v from its current clique to its best clique (with respect to s) involves a strict improvement (i.e., if $\Delta f(s, C^*) < 0$), we do move v from its current clique to C^* and thus we get a new solution s' from which we apply the same process; otherwise, we keep v in its current clique and we consider the next vertex. When there is no vertex that can be moved towards a clique better than its current clique, the descent is over.

3.3 Application of DWM to CPP

In DWM, we apply almost a descent but, from time to time, instead of moving the considered vertex to its best clique, we move it to a clique chosen randomly. More precisely, when a vertex v is considered, we have two possibilities: with a probability p, we choose a clique randomly, with a uniform probability on the cliques (including the empty one) of the current solution; or, with a probability $1 - p$, we compute the best clique for v; in both cases, we move v to the chosen clique.

In our experiments, p decreases arithmetically from a maximum value p_0 down to 0 (it is also possible to apply a geometrical decrease, as in simulated annealing, but then not down to 0); this parameter p_0 gives the probability to perform a mutation at the beginning of the whole

process. Then the method requires the specification of only two parameters: p_0 and the total number of performed transformations, which is directly related to the CPU time that the user wishes to spend to solve his or her problem. With this respect, DWM is much easier to tune than some other metaheuristics like simulated annealing, which involves several parameters.

4 Experiments

Experiments are not detailed here. They show that DWM may provide good results, with about the same quality (or even better) as the one provided by an improved and sharply tuned version of simulated annealing, within the same CPU time (or less), while, above all, it is very easy to design and to apply DWM to problems like CPP, and usually easier to tune than a simulated annealing method.

Indeed, the main advantage of DWM with respect to standard metaheuristics is that, aside the CPU time, there is only one parameter to tune: p_0. The sensibility analysis shows that the tuning of p_0 is not a crucial point if the value of p_0 is not too low. It is even possible to choose $p_0 = 1$, so that there is no parameter to tune!...

References

[1] S.G. de Amorim, J.-P. Barthélemy, C.C. Ribeiro, Clustering and clique partitioning: simulated annealing and tabu search approaches, Journal of Classification 9, 1992, 17-42.

[2] F. Brucker, J.-P. Barthélemy, Éléments de classification, Hermès, 2007.

[3] I. Charon, O. Hudry, Mixing different components of metaheuristics, in: I.H. Osman and J.P. Kelly (eds), Metaheuristics: Theory and Applications, Kluwer Academic Publishers, Boston, 1996, 589-603.

[4] I. Charon, O. Hudry, Application of the noising methods to the Travelling Salesman Problem, European Journal of Operational Research 125 (2), 2000, 266-277.

[5] M. Gendreau, J.-Y. Potvin (eds.), Handbook of Metaheuristics, Springer, 2010.

[6] O. Hudry, NP-hardness of the computation of a median equivalence relation in classification (Régnier's problem), Mathematics and Social Sciences 197, 2012, 83-97.

[7] O. Hudry, B. Leclerc, B. Monjardet, J.-P. Barthélemy, Metric and latticial medians, in Concepts and methods of decision-making process, D. Bouyssou, D. Dubois, M. Pirlot and H. Prade (eds), Wiley, 2009, 771-812.

[8] M. Krivanek, J. Moravek, NP-hard problems in hierarchical-tree clustering, Acta Informatica 23, 1986, 311-323.

[9] S. Régnier, Sur quelques aspects mathématiques des problèmes de classification automatique, I.C.C. Bulletin 4, 1965, 175-191. Reprint: Mathématiques et Sciences humaines 82, 1983, 13-29.

[10] Y. Wakabayashi, Aggregation of binary relations: algorithmic and polyhedral investigations, PhD thesis, Augsbourg, 1986.

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

Relaxation of 3-partition instances

Sebastiaan J.C. Joosten[1,3,4] and Hans Zantema[2,4]

[1]Department of Computer Science, Open University, Heerlen, The Netherlands
[2]Department of Computer Science, TU Eindhoven, Eindhoven, The Netherlands
[3]Department of Applied Mathematics, University of Twente, Enschede, The Netherlands
[4]Institute for Computing and Information Sciences, Radboud University Nijmegen, Nijmegen, The Netherlands

The 3-partition problem admits a straightforward formulation as a 0-1 Integer Linear Program (ILP). We investigate problem instances for which the half-integer relaxation of the ILP is feasible, while the ILP is not. We prove that this only occurs on a set of at least 18 elements, and in case of 18 elements such an instance always contains an element of weight ≥ 10. These bounds are sharp: we give all 14 instances consisting of 18 elements all having weight ≤ 10. Our approach is based on analyzing an underlying graph structure.

1 Introduction

Given a set of $3k$ elements $A = \{a_1, \ldots, a_{3k}\}$ with weights $w(a_1), \ldots, w(a_{3k})$. We ask if there is a partition into k triples, each with sum $c = \sum_{a \in A} w(a)/k$. This problem is called the 3-partition problem (3-PART), and is well-known to be NP-complete [2].

A straightforward approach to answer this is: first determine the set $\mathcal{C}$ of all *candidate sets*. That is, all sets $C \subseteq A$ such that: the set has three elements $|C| = 3$ and the sum of the set $\sum_{a \in C} w(a)$, which we will abbreviate with $w(C)$, is c. We denote all such sets as $\mathcal{C}$. Now 3-PART can be reformulated as finding k of these candidate sets in such a way that every element occurs exactly once in a chosen candidate set.

A solution of 3-PART is defined to be a selection of candidate sets represented by a mapping

$$f : \mathcal{C} \to \{0, 1\} \text{ such that: } \sum_{C \ni a} f(C) = 1 \text{ for all } a \in A, \tag{1}$$

in which the sum runs over all $C \in \mathcal{C}$ containing a.

This expresses the NP-hard problem 3-PART as an integer linear program (ILP). If we extend the range of f to the real interval $[0, 1]$, we obtain a linear program (LP) in polynomial time, which is polynomially solvable. Assuming $P \neq NP$, there exist 3-PART instances that have a solution to the LP, but are not feasible. Finding such an instance was stated as an open problem in [6]. We found several such 3-PART instances. Most of them had half-integral solutions, that is, the range of f is contained in $\{0, \frac{1}{2}, 1\}$. Such instances without integer but having half-integral solution we will call *nearly-feasible*. Every half-integral solution coincides with a solution of the following ILP problem (by multiplying by 2): find a mapping

$$g : \mathcal{C} \to \{0, 1, 2\} \text{ such that: } \sum_{C \ni a} g(C) = 2 \text{ for all } a \in A, \tag{2}$$

in which again the sum runs over all $C \in \mathcal{C}$ containing a.

Definition 1.1. *We say that a 3-PART instance is* nearly-feasible *if $\mathcal{C}$ is such that the problem (2) has a solution, while the problem (1) does not.*

We will present all nearly-feasible instances for which $k = 6$ elements and all numbers are ≤ 10, and prove:

- Every nearly-feasible instance has $k \geq 6$.
- Every nearly-feasible instance with $k = 6$, contains a number being at least 10.

The analysis of nearly-feasible instances is guided by its underlying graph structure. In Section 2, the underlying graph structure is analyzed, yielding the lower bound $k \geq 6$ for nearly-feasible instances. Moreover, for all 14 possible multigraphs for $k = 6$, corresponding instances are given. In Section 3, minimal values for the case $k = 6$ are investigated.

2 Nearly-feasible instances with minimal k

In this section, we prove that every nearly-feasible instance has $k \geq 6$ and show that this bound is tight. For this, we construct a multigraph (V, E) corresponding to any solution g of (2) defined by

$$V = \{C \in \mathcal{C} \mid g(C) = 1\}, \quad E(\{C, C'\}) = \left|C \cap C'\right| \tag{3}$$

So the vertices are the candidate sets for which $g(C) = 1$; since (1) has no solution by definition of nearly-feasible, there are such candidate sets and $V \neq \emptyset$. The number of edges between two such vertices is given by the number of elements the two candidate sets have in common. This multigraph is called the *solution graph* of g.

Lemma 1. *Let g be a solution to a nearly-feasible instance with minimal k, and (V, E) the solution graph of g. Then:*

1. *(V, E) is cubic, that is: $\forall C \in V.\ \sum_{C' \neq C} E(\{C, C'\}) = 3$.*
2. *(V, E) is connected*
3. *$|V| = 2k$*
4. *The candidate sets corresponding to any $k - 1$ vertices are not disjoint*

For a proof we refer to our full paper [5].

We investigated the graphs with the properties of Lemma 1 using `genbg` from the `nauty`-package to generate connected cubic graphs. Details about this tool can be found in [7]. The generated graphs were tested the graph for independent sets using a Haskell program that ran in just a few seconds. This yielded the following lemma:

Lemma 2. *Every connected cubic multigraph on $2k$ points has an independent set of size $k-1$ for $k \leq 5$. There are, up to isomorphism, 14 connected cubic multigraphs on 12 points with no independent set of size 5.*

By Lemma 2 and Lemma 1, we obtain:

Theorem 1. *Every nearly-feasible instance has $k \geq 6$.*

For each of the 14 multigraphs we found an instance for which the 12 candidate sets corresponding to the vertices in the graph are the only candidate sets, that is, every other triple of elements has a sum unequal to c. We found these within seconds of runtime with the SMT-solver `Yices` using the theory of linear inequalities. Details about `Yices` can be found in [1]. We required $w(C) = 1$, and scaled the solution back to integers. The found solutions can be drawn as described by (3). This gives a nice graphical representation which can be found in our full paper [5].

3 Nearly-feasible instances with low weights

We prove that every nearly-feasible instance with $k = 6$ contains an element with weight at least 10, by enumerating 3-PART instances with 18 elements and all weights ≤ 10. We selected those where (1) has no solution while (2) does, using the following observations to reduce the computation time:

1. Without loss of generality, we generate instances with weights in increasing order, starting at 0, and their sum a multiple of $k = 6$.
2. If m is the highest weight in an instance in which the sum per set is c, replacing weight $w(i)$ with $m - w(i)$ for all elements i creates another instance in which the sum per set is $3m - c$. Therefore, we only generate instances where $2c \leq 3m$.
3. In a nearly-feasible instance of 18 elements, every element occurs in at least 2 candidate sets. Therefore, we only proceed with instances for which this holds.

This way, 701827 instances remained to be checked. We run simplex (using the GLPK package), which should give a solution for the instances for which (2) has a solution. Most of them have no LP-solution, by which 197110 instances remain. Inspecting the LP-solutions, we remove the instances where a 1 occurs in a half-integral solution, as by Theorem 1 those instances are not nearly-feasible. Of the remaining instances, only 7 remained for which (1) has no solution. All of these were nearly-feasible. We show these instances in the following table on the left, and on the right those found using the trick stated in number 2.

0,0,1,1,1,2,2,2,4,4,4,5,5,5,8,8,10,10	0,0,2,2,5,5,5,6,6,6,8,8,8,9,9,9,10,10
0,0,1,1,1,2,3,3,4,4,4,4,4,6,6,9,10,10	0,0,1,4,4,6,6,6,6,6,7,7,8,9,9,9,10,10
0,0,1,1,2,2,2,2,4,4,4,5,5,5,8,8,9,10	0,1,2,2,5,5,5,6,6,6,8,8,8,8,9,9,10,10
0,1,1,1,2,2,2,4,4,4,4,4,5,5,5,8,10,10	0,0,2,5,5,5,6,6,6,6,6,8,8,8,9,9,9,10
0,1,1,1,2,2,3,3,3,4,4,4,6,6,6,6,10,10	0,0,4,4,4,4,6,6,6,7,7,7,8,8,9,9,9,10
0,1,1,1,2,3,3,3,4,4,4,4,4,6,6,6,10,10	0,0,4,4,4,6,6,6,6,6,7,7,7,8,9,9,9,10
0,1,1,2,2,2,2,4,4,4,4,4,5,5,5,8,9,10	0,1,2,5,5,5,6,6,6,6,6,8,8,8,8,9,9,10

These and all its permutations are the only nearly-feasible instances of 18 elements with weights ≤ 10. In particular, we proved the following.

Theorem 2. *Every nearly-feasible instance with $k = 6$, has an element with weight* 10 *or higher.*

In the seven nearly-feasible instances on the left, candidate sets have a sum of $c = 12$. In the right half, $c = 18$. We have verified that there are no feasible instances with $k = 6$ featuring

candidate sets with a sum of $c = 11$ using the approach mentioned here. Hence in the sense of c, the seven left instances are minimal as well.

A logical generalization to (2) parametrized by $M \geq 2$ is: find a mapping

$$g : \mathcal{C} \to \{0, \ldots, M\} \text{ such that: } \sum_{C \ni a} g(C) = M \text{ for all } a \in A, \tag{4}$$

The nearly-feasible instances presented until now turn out not to satisfy (4) for M odd. However, instances exist that have a solution to (4) for every $M \geq 2$, but not to (1) (or $M = 1$), for example: $0, 0, 0, 1, 1, 2, 3, 3, 4, 4, 4, 4, 4, 6, 6, 9, 10, 11$. This was checked by showing that (4) has a solution for both $M = 2$ and $M = 3$ (and not for $M = 1$); a solution to (4) for any $M > 3$ is obtained by taking a linear combination of the solutions for $M = 2$ and $M = 3$.

Also instances exist that have a solution to (4) for $M = 3$, but not for $M \leq 2$, for example: $0, 0, 1, 1, 2, 2, 4, 4, 5, 5, 5, 5, 5, 7, 7, 11, 13, 13$. For such instances, the structure is not well understood, and it is unknown what the least size of such instances could be.

Our notion of nearly-feasible also applies to two other NP-complete problems taken from the book of Garey and Johnson [3]. Every instance of 3-PART can be seen as an instance of exact cover by 3-sets. An exact cover by 3-sets instance is given by a collection $\mathcal{C}$ of 3-element subsets of some set A with $|A| = 3k$. The instance is feasible if one can pick k of the 3-element subsets in $\mathcal{C}$ such that every element in A is picked exactly once. For such problems we can define the same notion of nearly-feasible instance. In this setting, already nearly-feasible sets for $k = 2$ exists. For example, taking 6 elements $1, \ldots, 6$ and four candidate sets $\{1, 3, 5\}, \{1, 4, 6\}, \{2, 3, 6\}, \{2, 4, 5\}$, then the full set is not the union of two of these sets, but by choosing all four candidate sets all elements are chosen exactly twice. The instance just given also happens to be a 3-Dimensional matching instance. For more details and a weighted 3-Dimensional matching instance, we refer to [4].

References

[1] B. Dutertre and L. de Moura. The YICES SMT solver. Tool paper at http://yices.csl.sri.com/tool-paper.pdf, August 2006.

[2] M. R. Garey. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal of Computing*, 4:397–411, 1975.

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

[4] S.J.C. Joosten. Relaxations of the 3-partition problem. Master's thesis, University of Twente, Enschede, The Netherlands, December 2011.

[5] S.J.C. Joosten and H. Zantema. Relaxation of 3–partition instances. Technical Report ICIS–R13002, Radboud University Nijmegen, University of Twente, TU Eindhoven, February 2013.

[6] W. Kern and X. Qiu. Improved taxation rate for bin packing games. In *TAPAS 2011, Rome, Italy*, volume 6595 of *LNCS*, pages 175–180, Berlin, 2011. Springer Verlag.

[7] B.D. McKay. `nauty` *User's Guide (Version 2.4)*. Australian National University, ACT 0200, Australia, November 2009.

Semi blowup and blowup snarks and Berge-Fulkerson Conjecture*

K. Karam[1] and D. Sasaki[2]

[1]Institute of Computing, University of Campinas, Brazil
[2]COPPE, Federal University of Rio de Janeiro, Brazil

J. Hägglund recently described two constructions of snarks. We review both of them and consider an infinite family of snarks obtained with each construction. Then, we establish that each family satisfies the well-known Berge-Fulkerson Conjecture, which says that every bridgeless cubic graph has six perfect matchings such that every edge belongs to precisely two of them.

1 Introduction

Let G be a simple graph with *vertex set* $V(G)$ and *edge set* $E(G)$. If every vertex of G has degree three, then G is *cubic*. An edge $e \in E(G)$ is a *bridge* if $G - e$ has more components than G. A *matching* M of G is a subset of $E(G)$ such that no two edges of M are adjacent. If every vertex of G is incident with an edge of M, then M is a *perfect matching*. A *double cover by six perfect matchings* of G is a collection of six perfect matchings such that every edge of G belongs to exactly two of them. The challenging Berge-Fulkerson Conjecture was independently formulated by C. Berge and D. R. Fulkerson [7], and first published by Fulkerson [4].

Conjecture 1 (Berge-Fulkerson Conjecture)**.** *Every bridgeless cubic graph admits a double cover by six perfect matchings.*

An *edge-colouring* of G is a mapping $\pi\colon E(G) \to C$ such that for any two adjacent $e, f \in E(G)$, we have that $\pi(e) \neq \pi(f)$. If $|C| = k$, then π is a *k-edge-colouring*. For each *colour* $c \in C$, the set $\{e \in E(G)\colon \pi(e) = c\}$ is a *colour class*. In a 3-edge-coloured cubic graph, a double cover by six perfect matchings is obtained by duplicating each colour class. Therefore, it remains to verify the conjecture for bridgeless cubic graphs not admitting a 3-edge-colouring. Also notice that a minimal counterexample to this conjecture would be a cubic graph not admitting a 3-edge-colouring and cyclically 4-edge-connected (every edge-cutset of cardinality less than four consists of three edges incident to one vertex) [2]. Such graphs are called *snarks*. P. G. Tait [9] showed the equivalence between the famous Four Colour Conjecture and the following statement: every bridgeless cubic planar graph has a 3-edge-colouring. The pursuit for a counterexample to the Four Colour Conjecture brought great importance to snarks. Furthermore, several other well-known conjectures would also have snarks as minimal counterexamples,

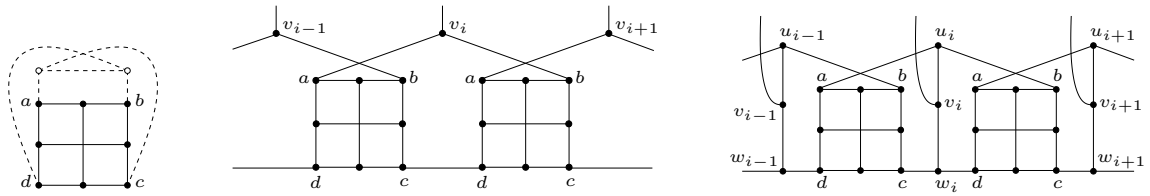
*Supported by CNPq.

among them Tutte's 5-Flow Conjecture and the Cycle Double Cover Conjecture [2]. Snarks have also proved their potential as counterexamples by refuting eight published conjectures [1]. Some families of snarks have been shown to verify Berge-Fulkerson Conjecture, such as the Flower snarks [2, 3, 6], the Goldberg snarks [3, 6], the generalized Blanuša snarks [3], the Szekeres snark [3], a family constructed with the dot product [2], and a family of Loupekine Snarks [8].

2 Semi blowup and blowup snarks

J. Hägglund [5] described two constructions of snarks, which are reviewed in this section.

Let B be the graph obtained by removing two adjacent vertices from the Petersen graph, as indicated in Figure 1(a). The degree-two vertices of B are a, b, c, d. Let G be a bridgeless cubic graph, with a 2-regular subgraph S. Take a cycle C of S, and let $C = v_1 v_2 \dots v_k$. Remove the edges of C. Take k copies $B_1, \dots, B_k$ of B. Indices greater than k are taken modulo k. In order to construct a *semi blowup* G' of (G, S), proceed as follows. For each $i \in \{1, 2, \dots, k\}$, make v_i adjacent to vertex a of B_i and to vertex b of B_{i+1}. Also, make vertex c of B_i adjacent to vertex d of B_{i+1}. Repeat this operation, illustrated in Figure 1(b), for every cycle of S. The resulting graph G' is denoted by *SemiBlowup*(G, S), adopting Hägglund's notation. Now, to construct a *blowup* G'' of (G, S), proceed in a slightly different manner. For each $i \in \{1, 2, \dots, k\}$, add vertices u_i and w_i, and edges $u_i v_i$ and $v_i w_i$. From B_i, make vertex a adjacent to u_i, and make vertex c adjacent to w_i. From B_{i+1}, make vertex b adjacent to u_i, and make vertex d adjacent to w_i. Repeat this process, illustrated in Figure 1(c), for every cycle of S. The resulting graph G'' is denoted by *Blowup*(G, S).



(a) Graph B. (b) Construction of a semi blowup snark. (c) Construction of a blowup snark.

Figure 1: Illustration of Hägglund's constructions.

3 Main Results

In this work, we consider two infinite families of semi blowup and blowup snarks constructed with the well-known generalized Petersen graphs $G(n, 1)$, $n \geq 5$, also known as n-Prisms [10]. Consider the generalized Petersen graphs $G(5, 1)$ and $G(6, 1)$ and their respective outer cycles C_5 and C_6. Figure 4 shows SemiBlowup$(G(5, 1); C_5)$ and SemiBlowup$(G(6, 1); C_6)$, while Figure 5 shows Blowup$(G(5, 1); C_5)$ and Blowup$(G(6, 1); C_6)$. We verify that both families satisfy the Berge-Fulkerson Conjecture.

Theorem 1. *For every $n \geq 3$, all SemiBlowup$(G(n, 1); C_n)$ such that C_n is the outer cycle of $G(n, 1)$ satisfy the Berge-Fulkerson Conjecture.*

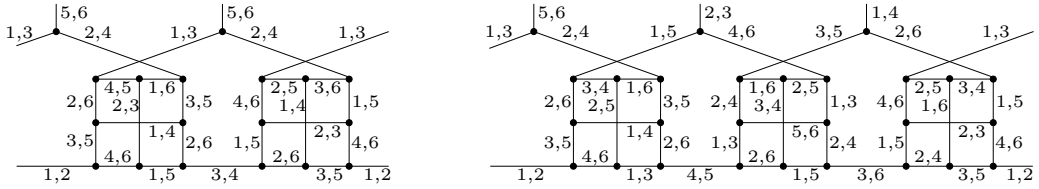
Proof sketch. Let G be a semi blowup as stated in the hypothesis. The proof consists of constructing a labeling of G using labels $\{1, 2, 3, 4, 5, 6\}$, such that each edge has two labels

and, for each label, every vertex is incident with an edge carrying the label. The construction uses the labelings of the pieces of a semi blowup depicted in Figure 2. The graph G can be constructed by joining copies of these pieces cyclically, and then joining them with a cycle of n vertices. If n is even, then it is enough to use only copies of the piece of graph of Figure 2(a). Otherwise, it is necessary to use one copy of the piece of graph of Figure 2(b). While joining the pieces, the labels are maintained. The labeling is easily extended to the central cycle of G. The labeling of the edges of G thus obtained induces a double cover by six perfect matchings. □

Theorem 2. *For every $n \geq 3$, all Blowup$(G(n,1); C_n)$ such that C_n is the outer cycle of $G(n,1)$ satisfy Berge-Fulkerson Conjecture.*

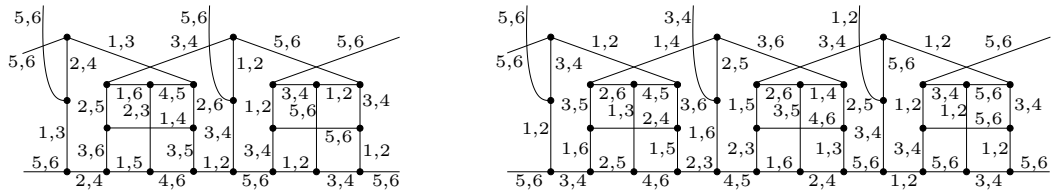
Proof sketch. Similar to the proof of Theorem 1, using the labelings of Figure 3 instead. □

Notice that it is possible to extend this result to other families of semi blowup and blowup snarks. For example, the families obtained by using the generalized Petersen graphs $G(4n, 2)$.



(a) (b)

Figure 2: Semi blowup coverings.



(a) (b)

Figure 3: Blowup coverings.

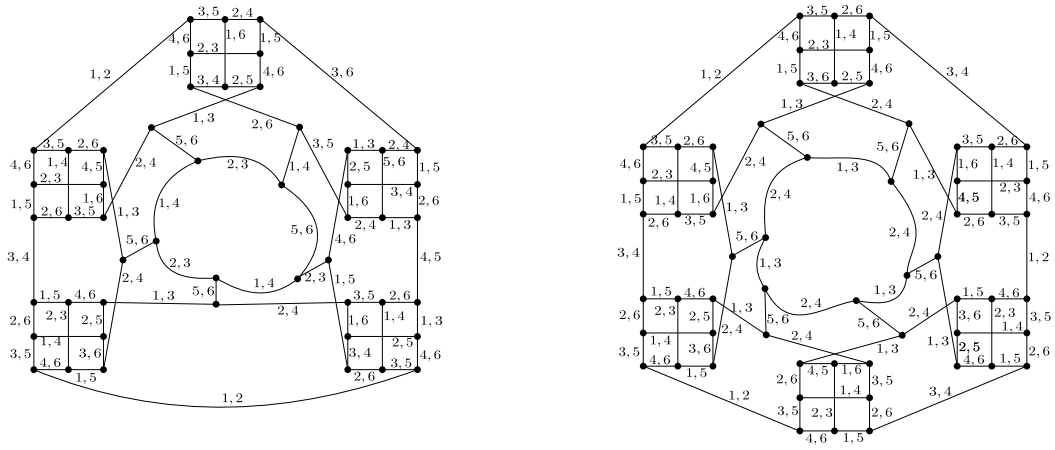


Figure 4: Examples of Theorem 1: SemiBlowup$(G(5,1); C_5)$ and SemiBlowup$(G(6,1); C_6)$.

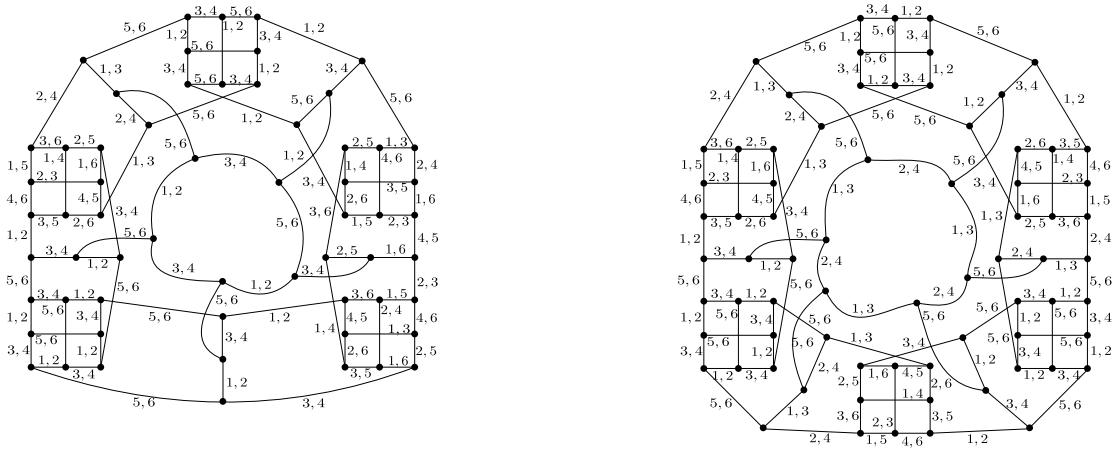


Figure 5: Examples of Theorem 2: Blowup$(G(5,1);C_5)$ and Blowup$(G(6,1);C_6)$.

Concluding Remarks In this work, we considered two infinite families of blowup and semi blowup snarks constructed with the well-known generalized Petersen graphs $G(n,1)$. We contributed with the celebrated Berge-Fulkerson Conjecture by establishing that both families verify this conjecture, providing further evidences together with other snark families, such as the generalized Blanuša snarks [3] and a family of Loupekine snarks [8]. Furthermore, we could extend this result to other families of semi blowup and blowup snarks by using, for example, the generalized Petersen graphs $G(4n,2)$.

Acknowledgments The authors would like to thank Myriam Preissmann for the fruitful comments on a preliminar version of this work.

References

[1] G. Brinkmann, J. Goedgebeur, J. Hägglund, and K. Markström. Generation and properties of snarks. To appear in J. Comb. Theory B, 2011.

[2] U. A. Celmins. A study of three conjectures on an infinite family of snarks. Technical Report CORR-79-19, Dept. of Combinatorics and Optimization, University of Waterloo, 1979.

[3] J. L. Fouquet and J. M. Vanherpe. On Fulkerson conjecture. *Discuss. Math. Graph Theory*, 31:253–272, 2011.

[4] D. R. Fulkerson. Blocking and anti-blocking pairs of polyhedra. *Math. Programming*, 1:168–194, 1971.

[5] J. Hägglund. On snarks that are far from being 3-edge colorable. `arXiv:1203.2015`, 2012.

[6] R. Hao, J. Niu, X. Wang, C.-Q. Zhang, and T. Zhang. A note on Berge-Fulkerson coloring. *Discrete Math.*, 309:4235–4240, 2009.

[7] T. R. Jensen and B. Toft. *Graph Coloring Problems.* Wiley Interscience, 1995.

[8] K. Karam and C. N. Campos. Fulkerson's Conjecture and Loupekine's Snarks. Proc. 11^{th} CTW 2012, Munich, Germany, 2012.

[9] P. G. Tait. Remarks on colouring of maps. *Proc. Royal Soc. Edinburgh Ser. A*, 10:729, 1880.

[10] M. E. Watkins. A theorem on Tait colorings with an application to the generalized Petersen graphs. *J. Combin. Theory*, 6:152–164, 1969.

An extension of the Collatz function

Roland Kaschek[1] and Alexander Krumpholz[2,3]

[1]The University of the Faroe Islands
[2]CSIRO, Canberra
[3]Australian National University, Canberra

1 The Collatz function and conjecture

Triggered by Lothar Collatz one has come to study the Collatz function $C : \mathbb{N} \to \mathbb{N}$, $n \mapsto n/2$, if n is even, and $n \mapsto 3n + 1$, else. The conjecture that for any $n \in \mathbb{N}$ there exists an $m \in \mathbb{N}$, such that $C^m(n) = 1$, among others, is called the Collatz conjecture. Related introductory and overview material for example is in [6, 8, 10, 13, 5]. According to [8], p. 2 it is unknown yet whether the Collatz conjecture is true. However, Schorer's proof attempts require closer scrutiny [11].

For every integer $z \neq 0$ we denote by $q(z)$ and $p(z)$ the largest odd divisor of z and the largest power of 2 that divides z, respectively. We define $p(0) = 0$ and $q(0) = 1$. Therefore $z = p(z)q(z)$ for any integer z. The strong Collatz function (see also [10]) f maps odd integers z to $3z + 1$, and even integers z to $q(z)$. The strong Collatz sequence S_z is the sequence $\{f^m(z)\}_{m\in\mathbb{N}}$. We call z its seed. The runway of S_z starts at z and goes to the first number that occurs in S_z repeatedly. The part of S_z between these two occurrences of that number is called cycle and that number is called a root ([1]) of it. The runway length is the number of iterations required to get from the seed to the root. The cycle length is the number of iterations required for any of its roots to return to it.

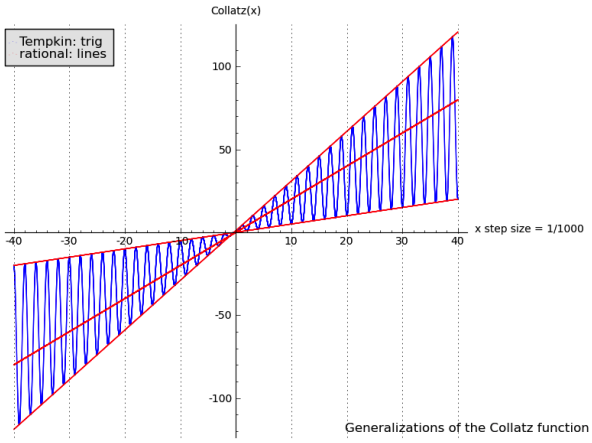
2 Observations

In [4, 3, 9] generalizations of the Collatz function have been discussed. We extend C to the rational numbers. For this we consider a rational number $r = \frac{a}{b}$ with $gcd(a, b) = 1$ as "even" if a or b is even and as "odd" otherwise. The predicates "odd" and "even" are well-defined.

Definition 1. For any rational number $r = \frac{a}{b} \in \mathbb{Q}$ with $gcd(a, b) = 1$ we define $f^*(r)$ as $\frac{3a+b}{b}$ if r is odd, as $\frac{q(a)}{b}$ if a is even and $\frac{a}{q(b)}$, else. We call $f^* : \mathbb{Q} \to \mathbb{Q}$, $r \mapsto f^*(r)$ the strong Collatz function on $\mathbb{Q}$.

The restriction of f^* to $\mathbb{N}$ equals f, i.e., $f^*|_{\mathbb{N}} = f$. Therefore we denote f^* by f. In the Figure 1 we show the graph of f and one of Tempkin's extension of C to $\mathbb{R}$. The function f attains values on the straight lines $y = 3x + 1, y = 2x$ and $y = x/2$ and Tempkin's function is defined by $T(x) = \frac{7x+2}{4} + \frac{5x+2}{4}\cos(\pi(x + 1))$. A similar extension of C to $\mathbb{R}$ namely $Ch(x) = x + \frac{1}{4} - \frac{2x+1}{4}\cos(\pi x)$ was proposed by Chamberland (see for all this [4]). We feel that our definition is closer in spirit to C than Tempkin's function. According to Chamberland

Figure 1: One of Tempkin's extensions of C to $\mathbb{R}$ as opposed to ours.



Tempkin, with regard to another, piece-wise defined, extension of C to $\mathbb{R}$ proves that it has divergent Series for seed of the form $\frac{k}{5}$ with $k \not\equiv 0\,(mod\,5)$. Our strong Collatz function does, however, not diverge on these seeds.

Remark 2. For any set S, set F of functions on S and any function $\varphi : S \to F$ we define $\Phi : S \to S$, $s \mapsto \phi(s)(s)$. Suppose $S = \mathbb{Q}$ and $F = \{r, s, t\}$ where r, s and t are functions on $\mathbb{Q}$ with $r(x) = 3x+1$, $s(x) = x/2$ and $t(x) = 2x$, respectively. If then ϕ maps the rational number $\frac{a}{b}$ with $gcd(a, b) = 1$ to r, s and t if a, b is odd, if a is even and b is even, respectively, then we get $\Phi = C$. Moreover we get an interesting function that enumerates the Collatz sequences if we define $\Phi^* : \mathbb{N}_0 \times \mathbb{Q} \to \mathbb{Q}$, that maps a pair (n, x) to x if $n = 0$ and to $\Phi(\Phi^*(n-1, x))$, otherwise. It is clear that $\Phi^*(n, x) = C^n(x)$, for any $n \in \mathbb{N}_0$ and $x \in \mathbb{Q}$. Note that we here refer to the original Collatz idea so that application of C, if at all, reduces the multiplicity of 2 as divisor by just 1.
For $x = \frac{1}{3}$, for example, we get $\Phi^*(0, \frac{1}{3}) = \frac{1}{3}$, $\Phi^*(1, \frac{1}{3}) = \varphi(\frac{1}{3})(\frac{1}{3}) = r(\frac{1}{3}) = 2$. It continues then with $\Phi^*(2, \frac{1}{3}) = \varphi(\Phi^*(1, \frac{1}{3}))(\Phi^*(1, \frac{1}{3})) = \varphi(2)(2) = s(2) = 1$. Now we are already in the loop since $\Phi^*(3, \frac{1}{3}) = \varphi(\Phi^*(2, \frac{1}{3}))(\Phi^*(2, \frac{1}{3})) = \varphi(1)(1) = r(1) = 4$ and finally $\Phi^*(4, \frac{1}{3}) = \varphi(\Phi^*(3, \frac{1}{3}))(\Phi^*(3, \frac{1}{3})) = \varphi(4)(4) = s(4) = 2$.

Remark 3. We have implemented a system of Python functions for computing strong Collatz sequences. Our related findings are: First (Figure 2) the runway-length as function of the seed appears as nearly space filling. Further studies seem to suggest that the blued area slowly, but steadily, extends towards larger ordinate values as the step-width decreases. Second (Figure 3) the cycle-length as a function of the seed is much more sparse and appears to occur in defined bands. Third (Figure 4) shows the cycles reached by the seed: $\frac{cell's\ column\ index}{cell's\ row\ index}$. The horizontal patterns indicate that the cycle is stronger effected by the denominator. Fourth, each seed seems to have an attracting cycle. Fifth, the runway and cycle sown by $\frac{255113845967565586}{85037948655855195}$ have the length 1114745 and 438480, respectively. Furthermore the runway and cycle sown by $\frac{11990350760475582496}{3996783586825194165}$ have the length 487429 and 4762800, respectively.

Acknowledgements

We thank Frederick Magata for an interesting discussion we had on the subject.

Figure 2: Runway-length as function of seed, step-width = 0.0001.

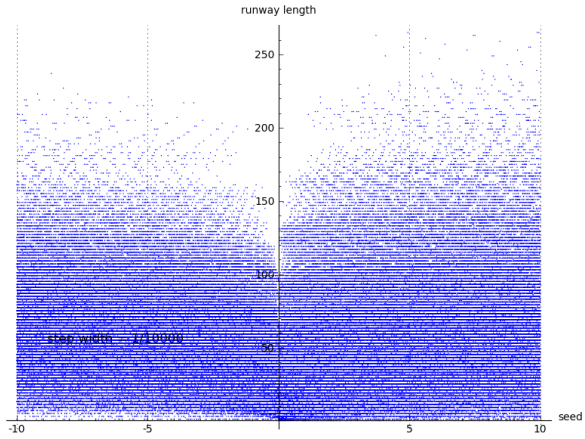


Figure 3: Cycle-length as function of seed, step-width = 0.0001.

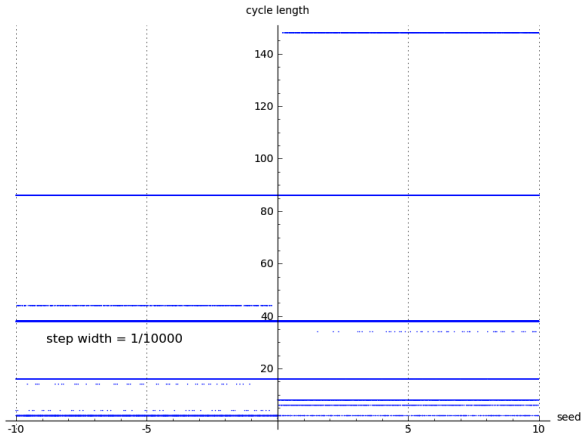
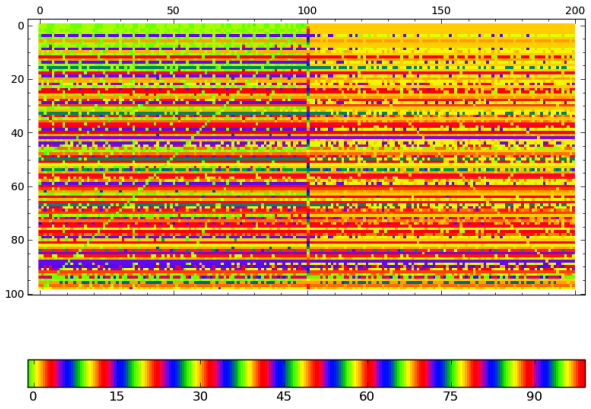


Figure 4: Cycle sown by a cell's column index divided by its row index.



References

[1] T. Allen: Characteristics of counter example loops in the Collatz conjecture. Master thesis, The University of Central Missouri, May 2012.

[2] S. Andreis, C. Masalagiu: About the Collatz conjecture. Acta Informatica 35 (1998): 167 - 179.

[3] B. Brent: 3x+1 dynamics on rationals with fixed denominator. arXiv:math/0204170v1, Apr 13th. 2002.

[4] M. Chamberland: An update on the 3x+1 problem. Internet publication, undated.

[5] J. Doe: Collatz conjecture. https://en.wikipedia.org/wiki/Collatz_conjecture, accessed: Mar 7th. 2013

[6] J. Lagarias: The 3x+1 problem and its generalizations. The American Mathematical Monthly, 92, 1 (1985):3 - 23.

[7] J. Lagarias: The 3x+1 problem: an annotated bibliography (1963 - 1999). arXiv:math/0309224v13, Jan 11th. 2011.

[8] J. Lagarias: The 3x+1 problem: an annotated bibliography, II. arXiv:math/0608208v6, Feb 12th. 2012

[9] J. Lesieutre: On a generalization of the Collatz conjecture. Research Science Institute, Jul 31st. 2007.

[10] P. Schorer: Are we near a solution to the 3x+1 problem? A discussion of several possible strategies.http://www.occampress.com/. Oct 26th. 2012.

[11] P. Schorer: A solution to the 3x+1 problem. http://www.occampress.com/. Feb 10th. 2013.

[12] M. Sinisalo: On the minimal number of cycle lengths of the Collatz sequences. Department of Mathematical Sciences, University of Oulu, preprint. Jun 2003.

[13] E. Weisstein: Collatz problem. MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/CollatzProblem.html, accessed on March 7th. 2013.

Constructing Strategies in Subclasses of McNaughton Games

Imran Khaliq[1] and Gulshad Imran[2]

[1]Department of Computer Science, [2]Department of Mathematics
The University of Auckland, New Zealand

We present algorithms for extracting winning strategies in subclasses of McNaughton games called update games and fully separated games. We also present an algorithm that solves update games with a bounded number of nondeterministic nodes in linear time which is an improvement over a quadratic time algorithm in update games.

Key words: Games on finite graphs, Müller games, finite state strategies

1 Introduction and Preliminaries

Recent years have seen a growing interest in two-player infinite games that are played on finite graphs. These games were first introduced by McNaughton in [1] motivated by the work of Gurevich and Harrington [2]. These games are not only natural models for reactive and concurrent systems [3] but also are used as a tool for analysis, synthesis and verification of such systems [4, 5]. Such games have also close connections in automata and logic [6].

In [1], McNaughton proved that winning strategies in his games can be simulated by a finite state automata. McNaughton games were further studied by Nerode, Rammel and Yakhnis and extracted winning strategies in concurrent programs [3]. Dinneen and Khoussainov investigated various subclasses of McNaughton games including update games and fully separated games [7, 8]. In both update games and fully separated games winner can be detected in polynomial time. In this paper we present procedures for extracting winning strategies in update games and fully separated games. We also show that winner can be detected in linear time in an update game with bounded number of nondeterministic nodes which is an improvement over the algorithm in [7].

The games we are interested in is a tuple (G, Ω), where $G = (V, E)$ is a finite directed bipartite graph with $V = V_0 \cup V_1$, $V_0 \cap V_1 = \emptyset$, $E \subseteq V_0 \times V_1 \cup V_1 \times V_0$ and $\Omega \subseteq 2^V$ is the winning condition. The elements of Ω are called target sets. We stipulate that a player can always make a move. Intuitively, the players play the game by moving a token along the edges of the graph. At initial round, the token is placed on a node $v_0 \in V$. At any round of the play, if the token is placed on a Player σ's node v, where $\sigma \in \{0, 1\}$, then Player σ chooses u such that $(v, u) \in E$, moves the token to u and the play continues on to the next round. Formally, *a play (starting from v_0)* is a sequence $v_0, v_1, v_2, \ldots$ such that $(v_i, v_{i+}) \in E$ for all $i \in \mathbb{N}$. We say Player 0 *wins a play* if the set containing all nodes that appear in the play infinitely often coincides with one of a target set in Ω; otherwise, Player 1 wins the play. These games are

called McNaughton games or games with Müller winning conditions. We simply refer to them as games and denote by Γ. *A strategy* for Player σ, $\sigma \in \{0,1\}$, is a function that takes as input initial segments of plays $v_0, v_1, \ldots v_k$ where $v_k \in V_\sigma$ and outputs some v_{k+1} such that $(v_k,\ v_{k+1}) \in E$. A game Γ is *determined* if one of the players has a winning strategy starting at any given node v of the game. *To solve a game* means to find all positions from which a given player wins. We define finite state strategies in the following.

Definition 1. *Let Γ be a game. (1) A* finite state strategy for Player σ *is given by a finite I/O automaton $\mathcal{S} = (Q, q_0, \delta)$ where Q is the finite set of states, $q_0 \in Q$ is the initial state, and $\delta : Q \times V_\sigma \to Q \times V_{1-\sigma}$ is the* transition function*. (2) A play $\rho = v_0, v_1, v_2, \ldots$ is* consistent *with $\mathcal{S}$ if there exists a sequence of states $q_0, q_1, q_2, \ldots$ such that for all $i \in \mathbb{N}$ we have the following. If $v_i \in V_\sigma$, then $\delta(q_i, v_i) = (q_{i+1}, v_{i+1})$; If $v_i \in V_{1-\sigma}$, then $q_{i+1} = q_i$. Thus, the strategy does not change its state when* Player $(1-\sigma)$ *makes moves. (3) The strategy $\mathcal{S}$ is a k-*state strategy *if $|Q| = k$. One state strategies are also called* memoryless strategies*. Thus, a memoryless strategy for* Player σ *is simply a function $\mathcal{S} : V_\sigma \to V_{1-\sigma}$.*

Let Γ be a game, $\sigma \in \{0,1\}$, and $X \subseteq V$. The *σ-attractor set of X*, denoted $\mathsf{Attr}_\sigma(X)$, is the set of all nodes $v \in V$ that satisfies the following: Player σ has a memoryless strategy $\mathcal{S}$ such that every play consistent with $\mathcal{S}$ that begins from v will eventually reach some node in X. The set $\mathsf{Attr}_\sigma(X)$ can be computed in time $O(|V| + |E|)$ [6]. If the tuple $(V_0' \cup V_1', E', \Omega')$, where $V_0' = V_0 \cap X$, $V_1' = V_1 \cap X$, $E' = E \cap (X \times X)$ and $\Omega' = \Omega \cap 2^X$, is a game then we call this a *subgame of Γ determined by X*.

2 Results

We focus on subclasses of McNaughton games called update games and fully separated games. An emphasis is placed on update games. An update game is a game where $\Omega = \{V\}$. If Player 0 is the winner of an update game then we say the update game is an *update network*. We investigate the following three interrelated problems:

PROBLEM 1: How hard is it to decide which players has a winning strategy?
PROBLEM 2: What is the size of a finite state winning strategy?
PROBLEM 3: How efficiently can update games be solved?

We give procedures for all the three problems. On of the key ideas, we use, is the concept of forced cycles, first introduced in [7]. A *forced cycle* is a cycle where Player 0 has full control whether to stay in the cycle or to leave it.

PROBLEM 1: To solve the first problem, a sequence of contracted games, that begins from the original update game, is constructed. For constructing this sequence we use the *contraction operator*. The operator to which an update game and a forced cycle in the game is input, outputs a new game. We call this new game the *contracted game*. The contracted game is constructed from the previous game by replacing the forced cycle by a forced cycle of length 2 called *spike*. The operator preserves the property of being an update network (or not being one). The sequence of contracted games is continued until no contraction of non-spike forced cycle can be made. If the last game in the sequence contains exactly one Player 0 node, then Player 0 wins the original game if and only if Player 0 wins the last game in the sequence. A game with exactly one Player 0 node u is an update network if and only if for every Player 1

node v there is an edge from u to v. We call such games *star-networks*. Thus, we obtain the following result.

Theorem 1 (Deciding Update Games)**.** *An update game is an update network if and only if the game can be transformed by a sequence of contractions of forced cycles into a star-network. Moreover, there exists an algorithm that constructs this sequence in quadratic time on the size of the underlying graph.* □

Problem 2: To solve the second problem we first prove that all update networks can be generated from star-networks by unfolding a series of spikes. For unfolding a spike we use *unfolding operator*. The unfolding operator is an inverse of contraction operator. Unfolding operator also preserves the property of being update network (or not being one). In the solution of the second problem we use the following theorem.

Theorem 2 (Building Update Networks)**.** *All update networks can be obtained by consecutively applying the unfolding operation to star-networks.* □

Given an update game, we build our finite state automaton that will determine a winning strategy for the winner. Assume a chain of contracted games is given. We prove that, in a star-network, the number of states of a winning strategy is equal to the number of Player 1's nodes in the star-network. This number of states increases by at most 1 for every unfolding of a spike in the sequence. If Player 1 wins the game, the strategy is even simpler, Player 1 has a memoryless winning strategy.

Theorem 3 (Extracting Winning Strategies)**.** *Suppose a sequence of contracted games of length k is given. (1) Assume that the last game in the sequence is a star-network with t number of* Player 1*'s nodes. Then, in the original game,* Player 0 *has a winning strategy with at most $t + k$ number of states. (2) Assume that the last game in the sequence is not a star-network. Then, in the original game,* Player 1 *has a memoryless winning strategy.* □

Problem 3: For the third problem we consider a special case of update games where at most k number of Player 1's nodes have more than one out going edges, where $k \geq 1$ is fixed. We call such nodes *nondeterministic* nodes. Thus, the update games we consider depend on the parameter k. We use a different approach to solve these games as opposed to the contraction of forced cycles. The iterated forced cycle contraction method may require quadratic time as it does not make use of nondeterministic nodes. In our procedure, certain type of strongly connected subgames are replaced by spikes in order to reduce the size of the game. The *derivative* of an update game is obtained from the update game by replacing all the strongly connected subgames in the update game by spikes. The procedure constructs a sequence of derivatives. We then prove that an update game is update network if and only if its derivative is so. We also prove that in an update network at least one of the nondeterministic nodes becomes deterministic in its derivative. Therefore, the sequence of derivatives stops at the stage where there is no nondeterministic node or the number of nondeterministic nodes in the previous game is equal to the number of nondeterministic nodes in the current game. As a result, we obtain the following theorem.

Theorem 4 (Deciding Update Games: Improved)**.** *Given an update game Γ with k nondeterministic nodes of* Player 1*, there exists an algorithm that constructs the sequence of derivatives in $O(|V| + |E|)$ time. Here the asymptotic constant is k. Moreover, the game Γ is an update*

network if and only if the last game in the sequence does not contain any nondeterministic node of Player 1 *and underlying graph to this game is strongly connected component.* □

Below we present a procedure for extract winning strategies for the winners in fully separated games. Our procedure embeds the above algorithm given in Theorem 3. Fully separated games have been studied by Hajimi Ishihara and Bakhadyr Khoussainov in [8]. Formally, a game Γ is called *fully separated game* if for each target set $T \in \Omega$ there is a node w_T, called *separator* such that $w_T \in T$ with $w_T \notin T'$ for all $T' \neq T$. The procedure that solves fully separated games is recursive one. One solves the game by constructing several smaller games. After solving these games, one then constructs the solution to the desired game in terms of the solutions of these smaller games. The procedure proceeds recursively as follows: It first identifies all the sets in winning condition such that the subgames over these sets form update networks. Using these subgames, a set N containing winning nodes for Player 0 is constructed. Note that N is of the form $\mathsf{Attr}_0(\mathsf{Attr}_0(T') \cup \mathsf{Attr}_0(T'') \cup \cdots \cup \mathsf{Attr}_0(T'''))$, where each target set is update network. A new game Γ' is obtained from the previous game by deleting N from the graph of the previous game. We declare a target set T in the previous game to be a target set in the new game if T is properly contained in V' of Γ'. The procedure stops when either N contains all the nodes in the current underlying graph or the newly constructed game does not contain a target set.

Theorem 5 (Deciding Fully Separated Games)**.** *There exists an algorithm that solves fully separated games in $O(k \cdot (|V| + |E|))$ time, where k is cardinality of the winning condition.* □

Before we present a procedure for extracting winning strategies in fully separated games we mention the following lemma that is true for any McNaughton game.

Lemma 1. *Let Γ be a McNaughton game. Then the following holds true. (1) Let $\mathcal{S}$ be t-state winning strategy from $X \subseteq V$ for* Player σ*, $\sigma \in \{0, 1\}$, in Γ. Then there exists a winning strategy $\mathcal{S}'$ with the same number t of states from $\mathsf{Attr}_\sigma(X)$ for* Player σ *in Γ. (2) Assume that* Player 0 *has winning strategies f and g such that f is s_1-state winning strategy from a 1-trap A and g is a s_2-state winning strategy from a 1-trap B. Then* Player 0 *has a $max\{s_1, s_2\}$-state winning strategy from $A \cup B$.* □

Theorem 3, Theorem 5 and Lemma 1 allows us to extract a winning strategy for the winner in a fully separated game through the following theorem.

Theorem 6 (Extracting Winning Strategies)**.** *Let Γ be fully separated game. (1) Assume* Player 0 *wins the game. Let N_1, N_2, ..., N_r be all sets constructed at the end of execution of the algorithm. Let $\mathcal{S}^i$ be t_i-state winning strategy from N_i for* Player 0 *in Γ, for all $i = 1, 2, \ldots, r$. Then* Player 0 *has a winning strategy with $max\{t_i \mid i = 1, 2, \ldots, r\}$ number of states from the union of all N_i. (2) Assume* Player 1 *wins the game. Then* Player 1 *has a memoryless winning strategy from the set $V \setminus \cup_{i=1}^{r} N_i$.* □

References

[1] McNaughton, R.: Infinite Games Played on Finite Graphs. *Annals of Pure and Applied Logic.* vol. 65, pp. 149 – 184, (1993)

[2] Gurevich, Y., Harrington, L.: Trees, Automata and Games. In: *Pro. of 14th ACM Symposium on the Theory of Computing.* STOCS. pp. 60 – 65 (1982)

[3] Nerode, A., Remmel, J., Yakhnis., A.: McNaughton Games and Extracting Strategies for Concurrent Programs. *Annals of Pure and Applied Logic.* vol. 78, pp. 203 – 242 (1996)

[4] Emerson, E.A., Jutla. C.S., Sistla, A.P.: On Model Checking for Fragments of μ-calculus. LNCS 697, pp. 385 – 396 (1993)

[5] Thomas, W.: Infinite games and verification. LNCS 2404, pp. 58 – 64, Springer (2002)

[6] Grädel, E., Thomas, W., Wilke, T.: Automata, logics, and infinite games: A Guide to Current Research. LNCS 2500, Springer, Heidelberg. (2002)

[7] Dinneen, M.J., Khoussainov, B.: Update Games and Update Networks. Journal of Discrete Algorithms. vol.1, issue 1. (2003)

[8] Ishihara, H., Khoussainov, B.: Complexity of Some Infinite Games Played on Finite Graphs. LNCS 2573, pp. 270 – 281, WG (2002)

Toward a precise integrality gap for triangle-free 2-matchings

Philipp Klodt[1] and Anke van Zuylen[2]

[1] Max-Planck-Institut für Informatik
[2]The College of William & Mary

We consider the traveling salesman problem (TSP) with edge costs that are symmetric and obey the triangle inequality. The integrality gap for the *subtour* LP relaxation is not known exactly, but Schalekamp, Williamson and Van Zuylen showed that the worst-case ratio of an optimal 2-matching to an optimal solution to the subtour LP is at most $\frac{10}{9}$. For the case that the support graph of a F2M is 2-connected, they show that the same ratio holds for the gap between 2-matchings and F2Ms, which is an even stronger statement.

We extend the latter result to *triangle-free* 2-matchings and are able to prove that the worst-case ratio is $\frac{7}{6}$ provided that the F2M is subtour-feasible. The triangle-free case can be seen as a first step towards an extension to the gap between the TSP and its subtour relaxation.

1 Introduction

We consider a complete undirected graph with vertex set V, edge set E and edge costs $c : E \to \mathbb{R}^+$ which are symmetric and obey the triangle inequality. The problem of finding the minimum-cost edge set ('tour') that visits each vertex exactly once is called the *traveling salesman problem* (TSP). Let $\delta(S)$ denote the set of edges with exactly one endpoint in $S \subset V$. Then the *subtour* LP relaxation of the TSP is given by the following linear program (LP).

$$
(SUBT)\quad
\begin{aligned}
&\text{minimize} && \textstyle\sum c(e)x(e) && && \\
&\text{subject to} && \textstyle\sum_{e\in\delta(i)} x(e) = 2 && \forall i \in V && (1)\\
& && \textstyle\sum_{e\in\delta(S)} x(e) \geq 2 && \forall S \subset V,\ 3 \leq |S| \leq |V| - 3 && (2)\\
& && 0 \leq x(e) \leq 1 && \forall e \in E && (3)
\end{aligned}
$$

Determining the precise integrality gap of the subtour LP relaxation for the TSP is an important open question, on which — despite significant research efforts — little progress has been made in over 30 years. Christofides's algorithm [3] finds a tour of cost at most $\frac{3}{2}$ times the optimum, and Wolsey [11] and Shmoys and Williamson [10] show that this implies an upper bound of $\frac{3}{2}$ on the integrality gap as well. Examples are known that show that the

integrality gap is at least $\frac{4}{3}$, and a famous conjecture (see for example Goemans [5]) states that the integrality gap is exactly $\frac{4}{3}$ for inputs that are symmetric and obey the triangle inequality.

Recently, progress has been made on approximating the TSP on graph metrics; in such metrics, there is an underlying unweighted graph, and the cost between i and j is equal to the length of the shortest path in this graph. The current best approximation guarantee is $\frac{7}{5} = 1.4$ by Sebő and Vygen [9]. This implies an upper bound of $\frac{7}{5}$ on the integrality gap for graph metrics as well. However, for general metrics, the status of the integrality gap of the subtour LP for the TSP has not changed.

Schalekamp, Williamson and Van Zuylen [8] resolve a conjecture by Boyd and Carr [2] by showing that the worst-case ratio of the minimum cost 2-matching compared to the optimal value of the subtour LP is equal to $\frac{10}{9}$. They remark that the worst-case ratio is attained for an instance in which the optimal subtour LP solution is a fractional 2-matching. A fractional 2-matching (F2M) is a basic solution to the LP obtained by dropping the subtour elimination constraints (2). It is known that fractional 2-matchings are half integral, and that the edges with $x(e) = \frac{1}{2}$ form vertex-disjoint odd cycles, connected by paths of edges with $x(e) = 1$ (Balinski [1]).

Schalekamp et al. [8] note that, even if we assume that the worst instances for the subtour LP integrality gap are F2Ms, then it is still not known what the exact integrality gap is. Markovic [7] considers the special case when the F2M consists of exactly two odd cycles of edges with $x(e) = \frac{1}{2}$ that have equal size, that are connected by paths of edges with $x(e) = 1$. Even for this very special case, it is not known whether the integrality gap is at most $\frac{4}{3}$, except for the case when the two odd cycles have at most 5 vertices each.

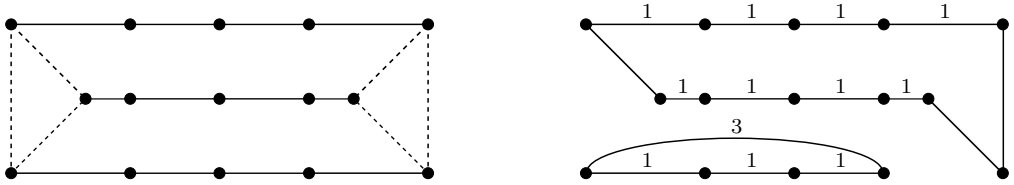
Given this state of affairs, we go back to the minimum cost 2-matching problem, i.e. the problem of finding a minimum cost subset of the edges such that every vertex is incident to exactly two edges. A k-cycle free 2-matching is a 2-matching in which each component has more than k vertices. Of course, a tour is a k-cycle free 2-matching for any $k \leq n - 1$. A natural question would thus be to ask for which values of k we can determine the worst-case ratio of the minimum cost of a k-cycle free 2-matching to the value of the subtour LP. The triangle-free 2-matching problem is an interesting problem in this context, as, contrary to the minimum cost 2-matching problem, it is unknown whether the problem is in P or not. The unweighted case has a polynomial time algorithm due to Hartvigsen [6].

We extend the results of Schalekamp et al. [8] and exactly determine the worst-case ratio of the minimum cost triangle-free 2-matching and a F2M that satisfies the subtour elimination constraints (2). We leave the question of determining the precise ratio of the minimum cost triangle-free 2-matching to the subtour LP (without any assumptions on the structure of the LP solution) as an intriguing open problem.

2 Gap between triangle-free 2-matchings and subtour-feasible F2Ms

2.1 Prerequisites

A *graphical 2-matching* (G2M) is a relaxaton of a 2-matching that allows vertices to have degree two or four, and edges to have zero, one or two copies. A G2M is called *triangle-free* if it does not contain any cycle of length 3. We will say a fractional 2-matching (F2M) is *subtour-feasible* if it satisfies the subtour elimination constraints (2).



(a) Graph with cost 1 on the solid edges and cost 0 on the dashed ones. The optimal F2M has x-value 1 on the solid edges and $\frac{1}{2}$ on the dashed ones, its cost is 12. Note that it satisfies the constraints (2).

(b) An optimal triangle-free 2-matching to the graph in Figure 1a with edge costs indicated beside the edges. The total cost is 14, so the example shows the gap is at least $\frac{14}{12} = \frac{7}{6}$.

Figure 1: Worst-case example

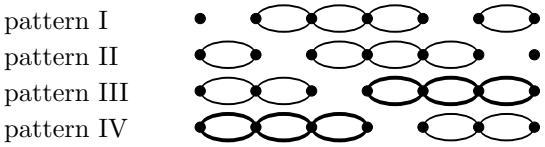


Figure 2: The 4 path patterns for $l \bmod 4 = 2$ with the 4-group highlighted at both ends

We will show that for a subtour-feasible F2M, a triangle-free 2-matching with cost at most $\frac{7}{6}$ times the F2M's cost exists. But before we do so, we have a look at the example graph in Figure 1 to convince ourselves that this gap cannot be less; in other words, this bound is tight.

2.2 Constructing the triangle-free 2-matching

To show that the gap is at most $\frac{7}{6}$, we consider a subtour-feasible F2M and remind the reader that an F2M consists of vertex-disjoint odd cycles with x-value $\frac{1}{2}$ whose nodes are connected via paths with x-value 1 along them (the optimal F2M in Figure 1a is an example).

We will show how to replace the cycle edges in the odd cycles and how to modify the paths to obtain a G2M of the desired cost. A natural extension of an idea in Schalekamp et al. [8] leads us to consider 4 different path patterns (see Figure 2), each of which may be used to replace one of the paths in the F2M.

In order to decide which pattern to use, and which cycle edges to remove, we define the following matching instance. Let G be the support graph of a fractional component. We define the cubic graph G^3 from G by replacing each path of edges with x-value 1 by a single edge. We now further define a graph G' by replacing each vertex of G^3 by 3 vertices in a row and each path by a *path gadget*. Its structure varies slightly according to the path length $l \bmod 4$ as the pattern starting with a group of 4 connected nodes is always associated to the central vertex in the path gadget (an example for $l \bmod 4 = 2$, i.e. the patterns in Figure 2, is given in Figure 3). Note that all these are extensions to the techniques of Schalekamp et al. [8]

The edge costs in G' are defined as follows. Internal edges have cost 0, cycle edges have the negative of their original costs and pattern edges have a cost equal to the sum of all edges in the pattern minus the cost of the original path.

Having defined G' entirely, we compute a minimum cost *perfect matching* (PM) on it and then construct a graph G_{G2M} as follows. We start with the original F2M, but increase the

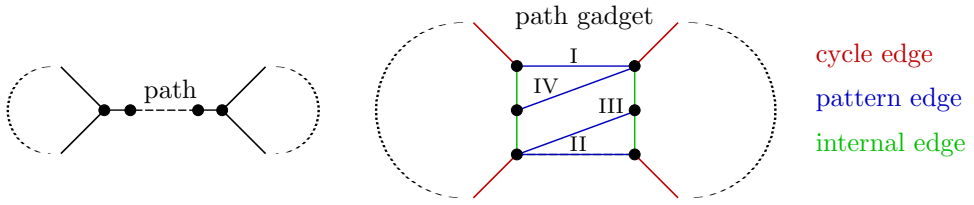


Figure 3: The path in G on the left and the resulting path gadget in G' on the right

edge value on the odd cycles to 1. We then remove every one of these edges that has its corresponding cycle edge in G' in the perfect matching. We replace each path that has one of its pattern edges in the PM by the corresponding path pattern (we can show that the PM contains at most 1 pattern edge of each pattern).

We then show that the resulting graph G_{G2M} is a triangle-free G2M, where the triangle-free property is the only part that does not follow easily from the techniques in Schalekamp et al. [8] The path patterns are designed to contain groups of 4 nodes, but special care is needed at the path ends as well as for cycles of size 3 in the F2M. Thanks to the triangle inequality for the edge costs we can transform G_{G2M} into a triangle-free 2-matching by *shortcutting* without increasing the total cost and thus if we show that the cost of G_{G2M} is at most $\frac{7}{6}$ the F2M's cost we complete the proof.

Finally we bound the cost of the optimal PM from above by defining a fractional edge-assignment y for G' that we will show to lie inside the perfect matching polytope as defined by Edmonds [4]. In particular, we let $y_e = \frac{1}{12}$ for all pattern edges, $y_e = \frac{5}{12}$ for cycle edges and for those internal edges adjacent to a node with degree 4 and finally $y_e = \frac{1}{2}$ for the remaining internal edges. This will give the desired bound of $\frac{7}{6}$ times the F2M's cost for the cost of the triangle-free 2-matching.

References

[1] M. L. Balinski. Integer programming: Methods, uses, computation. *Management Science*, 12:253–313, 1965.

[2] S. Boyd and R. Carr. A new bound for the ratio between the 2-matching problem and its linear programming relaxation. *Mathematical Programming*, 86:499–514, 1999.

[3] N. Christofides. Worst case analysis of a new heuristic for the traveling salesman problem. Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[4] J. Edmonds. Maximum matching and a polyhedron with (0,1) vertices. *J. Res. Nat. Bur. Standards Sect. B*, 69B:125–130, 1965.

[5] M. X. Goemans. Worst-case comparison of valid inequalities for the TSP. *Mathematical Programming*, 69:335–349, 1995.

[6] D. B. Hartvigsen. *Extensions of Matching Theory*. PhD thesis, Carnegie Mellon University, 1984.

[7] A. Markovic. Approximation algorithms for the undirected travelling salesman problem. Master's thesis, EPFL, 2011.

[8] F. Schalekamp, D. P. Williamson, and A. van Zuylen. A proof of the Boyd-Carr conjecture. In *Proceedings of the 23nd ACM-SIAM Symposium on Discrete Algorithms*, pages 1477–1486, 2012.

[9] A. Sebő and J. Vygen. Shorter tours by nicer ears: 7/5-approximation for graphic TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *CoRR*, abs/1201.1870, 2012.

[10] D. B. Shmoys and D. P. Williamson. Analyzing the Held-Karp TSP bound: A monotonicity property with application. *Information Processing Letters*, 35:281–285, 1990.

[11] L. A. Wolsey. Heuristic analysis, linear programming and branch and bound. *Mathematical Programming Study*, 13:121–134, 1980.

Handelman's hierarchy for the maximum stable set problem

Monique Laurent[1] and Zhao Sun[2]

[1]CWI and Tilburg University
[2]Tilburg University

1 Introduction

1.1 The Handelman hierarchy for polynomial optimization

Given real polynomials $p, g_1, \ldots, g_m \in \mathbb{R}[x]$ in n variables $x = (x_1, \ldots, x_n)$, we consider the *polynomial optimization problem*

$$p_{\max} = \max\ p(x)\ \text{ s.t. }\ x \in K = \{x \in \mathbb{R}^n : g_1(x) \geq 0, \ldots, g_m(x) \geq 0\} \tag{1}$$

i.e., maximizing p over the basic closed semialgebraic set K. This is an NP-hard problem, as it contains e.g. the maximum stable set problem and the maximum cut problem, which can both be formulated as instances of (1) with p square-free quadratic and $K = [0,1]^n$. Let $\mathcal{P}(K)$ denote the set of real polynomials that are nonnegative on K. Then (1) can be rewritten as

$$p_{max} = \min\ \lambda\ \text{ s.t. }\ \lambda - p \in \mathcal{P}(K). \tag{2}$$

A popular approach in the recent years is based on replacing the (hard to test) positivity condition $\lambda - p \in \mathcal{P}(K)$ by a tractable, sufficient condition for positivity. For instance, one may search for positivity certificates of the form $\lambda - p = \sum_{\alpha \in \mathbb{N}^m} c_\alpha g_1^{\alpha_1} \cdots g_m^{\alpha_m}$, where the multipliers c_α are nonnegative scalars, which leads to the Handelman hierarchy of linear programming relaxations for (1). When K is a polytope, the asymptotic convergence to $p_{\max}$ is guaranteed by a result of Handelman [1] showing that any polynomial strictly positive on K belongs to $\mathcal{P}(K)$. Alternatively one may search for positivity certificates of the form $\sum_{\alpha \in \mathbb{N}^m} s_\alpha g_1^{\alpha_1} \cdots g_m^{\alpha_m}$ (or simpler $s_0 + \sum_{j=1}^m s_j g_j$), where the multipliers s_α are now sums of squares of polynomials. This leads to the Lasserre hierarchy of semidefinite programming relaxations for (1), whose asymptotic convergence is guaranteed for K compact by results of real algebraic geometry.

Although the Lasserre hierarchy is stronger, it is more difficult to analyze and computationally more expensive as it relies on semidefinite programming. This motivates the study of LP based hierarchies which are generally easier to analyze, and might yet provide some insightful information, also for the SDP based hierarchies which they dominate. We refer e.g. to [2] for a study of LP hierarchies for polynomial optimization over the hypercube. Using Bernstein approximations, [2] gives error bounds for the Handelman hierarchy, which however apply only for order $t \geq n$. Recently, [4] gives refined error bounds that apply for any order $t \leq n$ for the case of square-free quadratic polynomials.

We consider here this setting of maximizing a quadratic square-free polynomial p over the hypercube $[0,1]^n$ (or, equivalently, on the boolean hypercube $\{0,1\}^n$). Then the Handelman hierarchy can be formulated as follows. For an integer $t \geq 1$, define the Handelman set

$$H_t = \left\{ \sum_{T \subseteq [n], I \subseteq T, |T| \leq t} c_{I,T} x^I (1-x)^{T \setminus I} : c_{I,T} \geq 0 \right\}$$

setting $x^I = \prod_{i \in I} x_i$ and $(1-x)^{T \setminus I} = \prod_{j \in T \setminus J}(1-x_j)$, and the Handelman bound of order t

$$\text{han}_t(p) = \inf\{\lambda : \lambda - p \in H_t\}. \tag{3}$$

We have that $p_{\max} \leq \text{han}_t(p)$, with equality at order $t = n$, in view of the following identity: $f(x) = \sum_{I \subseteq [n]} f(\chi^I) x^I (1-x)^{[n] \setminus I}$, which holds for any square-free polynomial f. Using a combinatorial version of Bernstein approximations, [4] shows the following error bound:

$$\frac{\text{han}_t(p)}{p_{\max}} \leq \frac{n}{t} \quad \text{for} \;\; 2 \leq t \leq n. \tag{4}$$

In this note we investigate the Handelman hierarchy applied to the maximum stable set problem, and answer some open questions posed in [4]. In particular we show a relation to fractional clique coverings, we give bounds for the rank of the Handelman hierarchy (i.e., the smallest order needed to obtain the true optimum), and we point out links to two other LP hierarchies (of Sherali-Adams and Lovász-Schrijver).

1.2 Application to the stable set problem

Let $G = (V = [n], E)$ be a graph with node weights $w \in \mathbb{R}^V_+$ (assumed to be nonnegative throughout). Then $\alpha(G,w)$ denotes the maximum weight $w(S) = \sum_{i \in S} w_i$ of a stable set S in G. Following [4] define the polynomial

$$p_{G,w} = \sum_{i \in V} w_i x_i - \sum_{i<j : \{i,j\} \in E} w_{ij} x_i x_j, \tag{5}$$

where we set $w_{ij} = \max\{w_i, w_j\}$ for $i, j \in V$. It is not difficult to verify that $\alpha(G,w)$ can be computed via the following optimization problem over the hypercube:

$$\alpha(G,w) = \max_{x \in [0,1]^n} p_{G,w}(x).$$

We can use the Handelman hierarchy (3) to bound the stability number and, as $p_{G,w}$ is square-free, the error bound (4) applies. In addition, let $\text{rk}_H(G,w)$ denote the smallest integer t for which $\text{han}_t(p_{G,w}) = \alpha(G,w)$, called the *Handelman rank* of (G,w). In the unweighted case (i.e., $w_i = 1 \; \forall i$), we simply omit the subscript w and write p_G, $\text{han}(p_G)$, $\alpha(G)$ and $\text{rk}_H(G)$. It is shown in [4] that $\text{han}_t(p_{G,w}) \geq (\sum_{i \in [n]} w_i)/t$, which implies $\text{rk}_H(G,w) \geq (\sum_{i \in [n]} w_i)/\alpha(G,w)$. For instance, the Handelman rank of the complete graph is largest possible: $\text{rk}_H(K_n) = n$.

2 Main results

2.1 Upper bound for the Handelman rank

We begin with a upper bound for the Handelman rank of general graphs, sharper bounds are given for specific graph classes thereafter. This bound is tight e.g. for the complete graph K_n.

Theorem 1. *For any weighted graph (G, w), $rk_H(G, w) \leq |V(G)| - \alpha(G) + 1$.*

Here is a sketch of proof. The starting point is the following identity, which holds for any square-free polynomial: $f(x) = (1 - x_n)f(\underline{x}, 0) + x_n f(\underline{x}, 1)$, where we set $\underline{x} = (x_1, \ldots, x_{n-1})$. Now observe that setting $x_n = 0$ in the polynomial $f_{G,w} = \alpha(G, w) - p_{G,w}$ corresponds to deleting node n, giving the graph $G \setminus n$, while setting $x_n = 1$ corresponds to deleting n and its neighbors, giving the graph $G \ominus n$ (roughly). More precisely, $f_{G,w}(\underline{x}, 0) = f_{G\setminus n,w}(\underline{x}) + \alpha(G, w) - \alpha(G \setminus n, w)$, while $f_{G,w}(\underline{x}, 1) = f_{G\ominus n,w}(\underline{x}) + q$, where $q \in H_2$. This implies:

$$\mathrm{rk}_H(G, w) \leq 1 + \max\{\mathrm{rk}_H(G \setminus n), \mathrm{rk}_H(G \ominus n), 2\}. \tag{6}$$

The result of Theorem 1 follows using induction on $|V(G)|$.

2.2 Link to fractional clique covers

Let (G, w) be a weighted graph. Given an integer t, a *fractional t-clique cover* of (G, w) is a collection of cliques C in G of size at most t together with scalars $\lambda_C \geq 0$ satisfying $\sum_C \lambda_C \chi^C = w$. Then the *fractional t-clique cover number* of (G, w) is the parameter

$$\rho_t(G, w) = \min\{\sum_C \lambda_C : \sum_C \lambda_C \chi^C = w,\ \lambda_C \geq 0\ \forall C \text{ clique of } G \text{ with } |C| \leq t\}. \tag{7}$$

Theorem 2. *$\rho_t(G, w) \geq \mathrm{han}_t(p_{G,w})$, with equality in the case $t = 2$.*

Theorem 3. *Given a graph G, $rk_H(G, w) \leq 2$ for all $w \in \mathbb{R}^n_+$ if and only if G is bipartite.*

This answers an open question of [4]. Note that there exist *nonbipartite* graphs with Handelman rank 2: Let G be obtained by taking the clique sum of t copies of K_{t+1} along a common K_t. Then, $\alpha(G) = t = \rho_2(G)$ (since one can cover all the nodes by t edges), and thus $\mathrm{rk}_H(G) = 2$.

2.3 Results for some graph classes

Perfect graphs. When G is a perfect graph, it is known that $\alpha(G, w) = \rho_t(G, w)$ for $t \geq \omega(G)$, where $\omega(G)$ is the clique number of G. Therefore, the Handelman relaxation of order $\omega(G)$ is exact: $\alpha(G, w) = han_{\omega(G)}(P_{G,w})$ and thus $\mathrm{rk}_H(G, w) \leq \omega(G)$ for any $w \in \mathbb{R}^n_+$. Moreover, if G is vertex-transitive, then $\mathrm{rk}_H(G) \geq n/\alpha(G) = \omega(G)$. Summarizing:

Theorem 4. *If G is perfect then $rk_H(G, w) \leq \omega(G)$, with equality in the unweighted case if G is vertex-transitive.*

Odd cycles and their complements. Let $G = C_{2n+1}$ be an odd cycle. As deleting a node creates a bipartite graph, we deduce using relation (6) combined with Theorem 3 that $\mathrm{rk}_H(C_{2n+1}, w) \leq 3$ (answering an open question of [4]). This implies that odd wheels have Handelman rank at most 4. Analogously, deleting a node in the complement of C_{2n+1} creates a perfect graph with clique number n, which implies $\mathrm{rk}_H(\overline{C_{2n+1}}) \leq n + 1$ (using again (6)). In the unweighted case, we have $\mathrm{rk}_H(C_{2n+1}) = 3$ and $\mathrm{rk}_H(\overline{C_{2n+1}}) = n + 1$. As an application, a graph G is perfect if and only if $\mathrm{rk}_H(G') \leq \omega(G')$ for every induced subgraph G' of G.

t-perfect graphs. A graph G is said to be *t-perfect* if its stable set polytope is completely described by the nonnegativity conditions $x_i \geq 0$ ($i \in V(G)$), the edge inequalities $x_i + x_j \leq 1$ ($ij \in E$), and the odd circuit inequalities $\sum_{i \in V(C)} x_i \leq (|C| - 1)/2$ for all odd circuits C of G.

Based on the fact that edges and odd cycles have Handelman at most 3 we can show that the Handelman rank of weighted t-perfect graphs is at most 3. Note however that this bound applies when defining the edge weights w_{ij} by $w_{ij} = w_i w_j$ (in (5)) and assuming $w_i \geq 1\ \ \forall i$.

3 Graph operations and links to other hierarchies

It is interesting to understand the behaviour of the Handelman rank under graph operations like node or edge deletion and clique sums. A first observation is that the rank is *not* monotone under node or edge deletion. For instance, the Handelman rank of $K_4 \setminus e$ is 2 (since $\alpha(K_4 \setminus e) = \rho_2(K_4 \setminus e)$), while the rank is 3 if we delete a node of degree 2 or an edge adjacent to it.

When deleting node u in G one can show the following: $\mathrm{rk}_H(G \setminus u) \leq \mathrm{rk}_H(G)$ if $\alpha(G) = \alpha(G \setminus u)$, $\mathrm{rk}_H(G \setminus u) \geq \mathrm{rk}_H(G)$ if $\alpha(G) = \alpha(G \setminus u) + 1$, with equality if u is an isolated node.

We do not formulate the exact results regarding clique sums which require case analysis. Based on this one can show that graphs with tree-width at most 2 have Handelman rank at most 3. (Alternatively, this follows from the fact that such graphs are t-perfect and the above bound 3 on their Handelman rank). An interesting open question is whether more generally it is true that the Handelman rank is at most the tree-width of the graph plus 1. We observe that this bound holds true for other hierarchies like the Sherali-Adams (SA) LP hierarchy and the Lasserre SDP hierarchy which both dominate the Handelman hierarchy. On the other hand, it is not known whether this holds for the Lovász-Schrijver (LS) hierarchy which seems most closely related to the Handelman hierarchy.

The LS procedure builds a hierarchy of polyhedra for the stable set polytope starting from the fractional stable set polytope (defined by nonnegativity and edge inequalities). Among others [3] shows that its rank is at most $|V(G)| - \alpha(G) - 1$, that the first relaxation is exact precisely for t-perfect graphs, and that the rank is equal to $\omega(G) - 2$ for perfect graphs. This suggests a shift of 2 between the Handelman and LS ranks, justified by the fact that edges are already taken into account at level 0 of the LS construction. The gap between the two ranks can be arbitrarily large: the clique t-sum of t copies of K_{t+1} has Handelman rank 2 and LS rank t.

References

[1] D. Handelman. Representing polynomials by positive linear functions on compact convex polyhedra. *Pacific Journal of Mathematics,* **132**:35–62, 1988.

[2] E. de Klerk and M. Laurent. Error bounds for some semidefinite programming approaches to polynomial optimization on the hyeprcube. *SIAM J. Opt.*, **20(6)**:3104–3120, 2010.

[3] L. Lovász and A. Schrijver. Cones of matrices and set-functions and $0-1$ optimization. *SIAM J. Disc. Mathematics*, **1(2)**: 166-190, 1991.

[4] M.-J. Park and S.-P. Hong. Handelman rank of zero-diagonal quadratic programs over a hypercube and its applications. *J. Global Optimization.* Published online: 17 April 2012.

A New Upper Bound for the Traveling Salesman Problem in Cubic Graphs

Maciej Liśkiewicz[1] and Martin R. Schuster[1]

[1]Institute of Theoretical Computer Science, University of Lübeck, Germany

1 Introduction

It is an outstanding open problem whether the traveling salesman problem (TSP) and the closely related Hamiltonian cycle problem can be solved in $\mathcal{O}(c^n)$ time for graphs on n vertices, for some constant $c < 2$. Recently Björklund et al. [3] have shown that the classical Bellman-Held-Karp exact algorithms [2, 9] for solving TSP can be modified to run in time $\mathcal{O}((2-\varepsilon)^n)$, where $\varepsilon > 0$ depends only on the maximum vertex degree. This provides the first upper bound on the time complexity of TSP that lies below 2^n for a broad class of graphs such as bounded degree graphs. Particularly, applying the result of [3] for graphs with maximum vertex degree three, also called cubic graphs, one gets that TSP can be solved in time $2^{3n/4}n^{\mathcal{O}(1)} = \mathcal{O}(1.682^n)$. On the other hand, the problem of testing whether a cubic graph has a Hamiltonian cycle and consequently the decision version of TSP remain NP-complete even if the graphs are restricted to be planar [7].

Exact algorithms for TSP for special classes of bounded degree graphs, in particular for cubic graphs, have been the subject of separate studies. The motivation for the study comes both from theoretical concerns and from practical applications, e. g. in computer graphics [1, 6]. The first exact algorithm for TSP in cubic graphs running faster than in time 2^n was proposed by Eppstein [5]. His algorithm solves the problem in $2^{n/3}n^{\mathcal{O}(1)} = \mathcal{O}(1.260^n)$ time and linear space and additionally it is easy to implement. Thus, although the technique by Björklund et al. improves the upper bound 2^n for any degree bounded ≥ 3, for specific bounds, like e. g. 3, better methods exist.

Eppstein's algorithm is a sophisticated recursive branch-and-bound search, which takes advantage of small vertex degrees in a graph. In [10] Iwama and Nakashima slightly modify Eppstein's algorithm and provide a new interesting method to bound the number of worst-case branches in any path of the branching tree corresponding to recursive subdivisions of the problem. As a consequence, Iwama and Nakashima claim $\mathcal{O}(1.251^n)$ to be an upper bound for the run-time of the algorithm. Unfortunately, their paper contains several serious mistakes that render the proof for the upper bound invalid (for details, see [11]). After reformulating the key lemma of [10] to be correct and then using the lemma to solve the recurrences derived in [10] in a proper way one could prove the upper bound $\mathcal{O}(1.257^n)$ that still beats the bound $\mathcal{O}(1.260^n)$ by Eppstein.

In this article we provide a new upper bound for TSP in cubic graphs. We show that Eppstein's algorithm with some minor modifications (see [11]), similar to those used in [10], yields the stated result:

Theorem 1.1. *The traveling salesman problem for an n-vertex cubic graph can be solved in $\mathcal{O}(1.2553^n)$ time and in linear space.*

Our proof techniques are based on ideas used by Eppstein [5] and Iwama and Nakashima [10] and a new, more careful study of worst-case branches in the tree of recursive subdivisions of the problem performed by the algorithm. Thus, our main contribution is more analytical than algorithmic. Nevertheless, we have implemented our algorithm and verified its easy implementability and good performance for graphs up to 114 vertices (for the experimental results see [12]).

Related work. Applying the result by Björklund et al. for an n-vertex graph with maximum degree four one gets that TSP can be solved in $\mathcal{O}(1.856^n)$ time and exponential space. Eppstein [5] showed that the problem can be solved in $\mathcal{O}(1.890^n)$ time but using only polynomial space. Next, Gebauer [8] proposed an algorithm that runs in time $\mathcal{O}(1.733^n)$. This algorithm can also list the Hamiltonian cycles but it uses exponential space. Very recently, Cygan et al. [4] have shown a Monte Carlo algorithm with constant one-sided error probability that solves the Hamiltonian cycle problem in $\mathcal{O}(1.201^n)$ time for cubic graphs and in $\mathcal{O}(1.588^n)$ time for graphs of maximum degree four. Though the technique presented in [4] works well for the Hamiltonian cycle problem, its modification to solve TSP seems to be highly nontrivial.

2 The Improved Analysis of Eppstein's Algorithm

Eppstein's TSP algorithm [5] searches *recursively* for a minimum weight Hamiltonian cycle H_{min} in a given cubic graph G. The algorithm constructs successively Hamiltonian cycles which are determined by a set of *forced edges* F. The goal is to find the set F which coincides with H_{min}. In each recursion step, an edge $e \in G \setminus F$ is chosen. Obviously, e either belongs to H_{min} or not. Thus, the algorithm makes two recursive calls: once e is added to F, assuming $e \in H_{min}$, and once e is removed from G, assuming $e \notin H_{min}$. The better solution to these two subproblems will then be returned. Splitting the problem into two subproblems determined by e will be called a *branch* (for an example see Fig. 1).

To analyse the time complexity of the algorithm, Eppstein derives a recurrence $T(s) = \max\{2T(s-3), T(s-2)+T(s-5)\}$, with a parameter s bounded by the number of vertices n. The solution $T(n) = \mathcal{O}(2^{n/3})$ of the recurrence provides a bound on the number of iterations of the algorithm.

Main Ideas of our Analysis. Our new analysis of Eppstein's algorithm largely relies on exploiting ideas due to Iwama and Nakashima [10] and on a new, more careful analysis of the branching tree. The worst-case component $2T(s-3)$ in the recurrence relation corresponds to the situations when the algorithm reduces the problem of size s to two subproblems each of size $s-3$. Eppstein proves that there are two cases leading to such situations. In their paper Iwama and Nakashima call the branches of the algorithm corresponding to these cases respectively A- and B-branches. Branches which are neither of type A nor B are called D-branches. Results of performing these branches are shown in Fig. 1.

Since A- and B-branches have the biggest impact on worst-case performance, the idea is to find an upper bound on the number of such branches. Iwama and Nakashima observed that the total number of A-branches along any path of the backtrack tree cannot exceed $n/4$ for any n-vertex input graph. Thus, an important challenge now is to bound the worst-case number of occurrences of B-branches and then to incorporate these information by introducing new variables into a recurrence equation.

(a) A-branch (b) B-branch

Figure 1: Branching on an edge $e = (y, z)$; bold black line: edge in F; bold gray line: edge selected by the current branch; dotted line: edge removed by the branch.

In our approach we will count, similarly as in [10], A-branches and B-branches together. We prove that if P is a single path of the backtrack tree and a, resp. b, denotes the number of A-, resp. B-branches, then $3a + 7b \leq n$ holds for any n-vertex input graph. This bound plays a crucial role in our analysis.

The Recurrence. For our analysis of the run-time of the algorithm, we define a multivariate recurrence equation in the variables n, s, x, y, and f. Variable n denotes the number of vertices of the input graph and will not be modified by the recurrence. Variable s is defined in a similar way as in [5]; we let $s = |V(G)| - |F| - 2|\mathcal{C}|$, where $\mathcal{C}$ is a set of 4-cycles of G that form connected components of $G \setminus F$. Next, let $x = n/4 - a \quad \text{and} \quad y = n/7 - b$, where a, resp. b, is the number of A-branches, resp. B-branches, made by the algorithm along the current backtrack path. Finally, f is the number of free vertices in G, meaning they are not adjacent to an edge in F. We can bound the worst-case number of leaves of the backtrack tree as the solution of the following recurrence defined for non-negative integers n, s, x, y, f as follows:

$$T(n,s,x,y,f) = \max \begin{cases} 2T(n,s-3,x-1,y,f-4) \\ 2T(n,s-3,x,y-1,f) \\ T(n,s-5,x,y,f-2) + T(n,s-2,x,y,f-2) \\ 2T(n,s-4,x,y,f) \\ T(n,s-4,x,y,f-2) + T(n,s-3,x,y,f-2). \end{cases} \tag{1}$$

The correctness follows from our main technical result bounding the number of A- and B-branches along any path of the backtrack tree:

Proposition 2.1. *Let P be a single path of the backtrack tree and suppose that there are b B-branches on P. Then along P the algorithm selects at least $4b$ edges which are neither selected by A- nor by B-branches.*

Since any A- and any B-branch selects three edges, the inequality $3a + 3b + 4b \leq n$ follows.

Corollary 2.2. *The invariant of the algorithm running on an input graph with n vertices is that $3a + 7b \leq n$ or equivalently that $3x + 7y \geq \frac{3}{4}n$.*

Corollary 2.3. *The worst-case number of leaves of the backtrack tree on an n-vertex graph is bounded by $T(n, n, n/4, n/7, n)$.*

To bound the solution for the recurrence (1) we use a function of the form $R(n, s, x, y, f) = 2^{\alpha s + \beta(x + \frac{7}{3}y - \frac{1}{4}n) + \gamma f}$. Note that the term $x + \frac{7}{3}y - \frac{1}{4}n$ incorporates the information provided by Proposition 2.1 and Corollary 2.2. Our aim is to find parameters α, β, and γ with the property that for all n, s, x, y, f it holds: $T(n, s, x, y, f) \leq R(n, s, x, y, f)$, and that the following upper bound is best possible: $T(n, n, n/4, n/7, n) \leq 2^{\alpha n + \beta(\frac{n}{4} + \frac{7}{3}\frac{n}{7} - \frac{1}{4}n) + \gamma n} = 2^{(\alpha + \frac{\beta}{3} + \gamma)n}$. Thus, we have to minimize $\alpha + \beta/3 + \gamma$. The minimization gives a rational approximation

$$\alpha + \tfrac{\beta}{3} + \gamma = \tfrac{1219}{3717} \quad \text{and} \quad T(n,n,n/4,n/7,n) \leq 2^{(\alpha+\frac{\beta}{3}+\gamma)n} = 2^{\frac{1219}{3717}n} \approx 1.25523^n.$$

This completes the proof of Theorem 1.1.

3 Conclusions

In this paper we have provided a new upper bound $\mathcal{O}(1.2553^n)$ for TSP in cubic graphs which consequently also applies for the Hamiltonian cycle problem. An interesting open problem is to further improve this bound. One could try e. g. to find a new algorithm and prove a better asymptotic time complexity than $\mathcal{O}(1.2553^n)$. On the other hand, we believe that the worst-case time complexity of Eppstein's algorithm is much smaller than the current upper bound.

Our experimental analysis (Fig. 2, for details see [12]) has confirmed that Eppstein's algorithm with our modification is easily to implement and that the algorithm has good performance. The experimental results show a gap between our upper bound on the tree size and actual sizes for graphs up to 112 vertices. This could indicate that the worst-case complexity of Eppstein's algorithm is much smaller than $\mathcal{O}(1.2553^n)$.

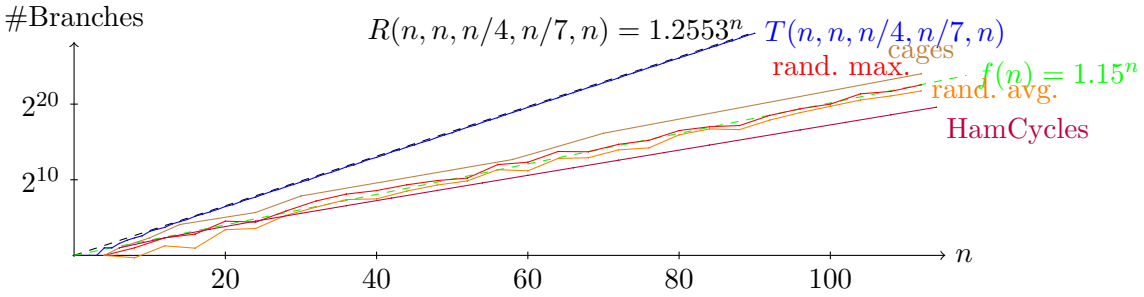


Figure 2: Total number of branches (in logarithmic scale) in backtrack trees for n-vertex random graphs, cages, and graphs of $2^n/3$ Hamiltonian cycles.

References

[1] Esther M. Arkin, Martin Held, Joseph S. B. Mitchell, and Steven S. Skiena. Hamiltonian triangulations for fast rendering. *The Visual Computer*, 12(9):429–444, 1996.

[2] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, 1962.

[3] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. The traveling salesman problem in bounded degree graphs. *ACM Trans. on Algorithms*, 8(2):18, 2012.

[4] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *FOCS*, pages 150–159, 2011.

[5] David Eppstein. The traveling salesman problem for cubic graphs. *J. Graph Algorithms Appl.*, 11(1):61–81, 2007.

[6] David Eppstein and M. Gopi. Single-strip triangulation of manifolds with arbitrary topology. *Eurographics Forum*, 23(3):371–379, 2004.

[7] Michael. R. Garey, David S. Johnson, and Robert Endre Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.

[8] Heidi Gebauer. Enumerating all hamilton cycles and bounding the number of hamilton cycles in 3-regular graphs. *Electr. J. Comb.*, 18(1), 2011.

[9] Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. *J. Soc. Ind. Appl. Math.*, 10:196–210, 1962.

[10] Kazuo Iwama and Takuya Nakashima. An improved exact algorithm for cubic graph tsp. In *COCOON*, pages 108–117, 2007.

[11] Maciej Liśkiewicz and Martin R. Schuster. A new upper bound for the traveling salesman problem in cubic graphs. *CoRR*, abs/1207.4694v2, 2012.

[12] Martin R. Schuster. *Exact Algorithms for Traveling Salesman Problem in Cubic Graphs*. Master Thesis, Universität zu Lübeck, 2012.

Manufacturing process flexibility with Robust Optimization using AIMMS

Ovidiu Listes[1]

[1]Paragon Decision Technology, Ovidiu.Listes@aimms.com

We address the design problem of manufacturing process flexibility and capacity expansion under uncertain demands using Robust Optimization. The model involves binary product-to-plant assignments and binary decisions for expanding capacity. Affine decision rules are used for capacity expansion and for the amounts of processed products. We illustrate how AIMMS is able to easily accommodate the formulation of the model under uncertainty and generate the robust counterpart automatically. We also show how the intuitive, effective modeling concepts in AIMMS allow for fast, flexible experiments and comparison of results based on various uncertainty sets.

Optimal Paths in Networks with Rated Transition Time Costs

Dmitrii Lozovanu[1] and Stefan Pickl[2]

[1]Institute of Mathematics and Computer Science, Academy of Sciences of Moldova, Academy str. 5, Chisinau, MD–2028, Moldova, e-mail: lozovanu@math.md
[2]Institute for Theoretical Computer Science, Mathematics and , Operations Research, Universität der Bundeswehr, München, 85577 Neubiberg-München, Germany, e-mail: stefan.pickl@unibw.de

Let $G = (X, E)$ be a directed graph with a vertex set X, $|X| = n$, and an edge set E where to each directed edge $e \in E$ a cost c_e is associated. For an arbitrary directed path $P(x, y) = \{x = x_0, e_0, x_1, e_1, x_2, e_2, \ldots, x_k = y\}$ with a given starting vertex x and a final vertex y we define the total rated cost $C(x, y) = \sum_{t=0}^{k-1} \lambda^t c_{e_t}$, where $\lambda > 0$ is a given positive value which we call the rate. We describe polynomial time algorithms for determining the paths with minimal total rated costs in networks with a fixed number of edges and with a free number of edges of the optimal path.

Keywords: Networks, Optimal paths, Time transition cost, Total rated cost, Polynomial time algorithm.
MSC: 90B10, 90B20.

1 Introduction and Problem Formulation

In this paper we consider an optimal paths problem on networks which generalizes the well-known minimum cost path problem in a weighted directed graph. The formulation of this problem is the following.

Let $G = (X, E)$ be a finite directed graph with a vertex set $X, |X| = n$ and an edge set E where to each directed edge $e = (u, v) \in E$ a cost c_e is associated. Assume that for two given vertices x, y there exists a directed path $P(x, y) = \{x = x_0, e_0, x_1, e_1, x_2, e_2, \ldots, x_k = y\}$ from x to y. For this directed path we define the total rated cost

$$C(x_0, x_k) = \sum_{t=0}^{k-1} \lambda^t c_{e_t},$$

where λ is a positive value. So, in this path the costs c_{e_t} of the directed edges e_t are rated by $\lambda^t C_{te}$. We consider the problem of determining a path from x to y with a minimal total rated cost for the case with a fixed number of edges and for the case with a free number of edges. If $\lambda = 1$ then the formulated problem becomes the well known problem of determining the shortest path from x to y. In general, the considered problem can be regarded as the problem of determining the optimal paths in a dynamic network determined by the graph

$G = (X, E)$ with the cost function $c_e(t) = \lambda^t c_e$ on the edges $e \in E$ that depend on time. Therefore if the number k of the edges for the optimal path is given then we can apply the dynamic programming algorithm or the time-expanded network method which determines the solution of the problem using $O(|x|^3 k)$ elementary operations.

Here we show that for the considered problem the linear programming approach can be applied and new algorithms for determining the optimal paths can be gained.

2 The Main Results

At first we show how a linear programming approach for the problem without a restriction on the number of edges of the optimal path can be applied.

Assume that the costs $c_e,\ e \in E$ are nonnegative and consider the following linear programming problem:
Minimize

$$\phi(\alpha) = \sum_{e \in E} c_e \alpha_e \tag{1}$$

subject to

$$\begin{cases} \sum\limits_{e \in E^-(u)} \alpha_e - \lambda \sum\limits_{e \in E^+(u)} \alpha_e = 1, & u = x, \\ \sum\limits_{e \in E^-(u)} \alpha_e - \lambda \sum\limits_{e \in E^+(u)} \alpha_e = 0, & \forall u \in X \setminus \{x, y\}, \\ \alpha_e \geq 0, & \forall e \in E, \end{cases} \tag{2}$$

where $E^-(u)$ is the set of directed edges that originates in the vertex $u \in X$ and $E^+(u)$ is the set of directed edges that enters in u.

We have proved the following results:

Theorem 1. *If $\lambda \geq 1$ and in G there exists at least a directed path $P(x, y)$ from a given starting vertex x to a given final vertex y then for nonnegative costs c_e of the edges $e \in E$ the linear programming problem (1),(2) has solutions. If α_e^* for $e \in E$ represents an optimal basic solution of this problem then the set of directed edges $E^* = \{e \in E | \alpha_e^* > 0\}$ determines an optimal directed path from x to y.*

Theorem 2. *If $G = (X, Y)$ has a structure of an acyclic directed graph with a sink vertex y then for an arbitrary $\lambda \geq 0$ and arbitrary costs $c_e, e \in E$ there exists the solution of the linear programming (1),(2). Moreover, if α_e^* for $e \in E$ represents an optimal basic solution of this problem then the set of directed edges $E^* = \{e \in E \mid \alpha_e^* > 0\}$ determines an optimal directed path from x to y.*

Note that for some special cases the linear programming (1),(2) determines the optimal path from x to y with $0 < \lambda < 1$ when graph $G = (X, E))$ contains directed cycles. However, there may be cases where the linear programming problem (1),(2) with $0 < \lambda < 1$ may not find the optimal path from x to y even for positive costs $c_e, \forall e \in E$ if $G = (X, E))$ contains directed cycles. The problem with $0 < \lambda < 1$ can be treated as the optimal control problem on networks with discounted costs and in the general case it can be studied using the linear programming

models from [2, 4] related to a stochastic control problem and discounted Markov decision processes with absorbing states.

In order to elaborate efficient algorithms for solving the problem in the cases mentioned above we have used the following dual model for the linear programming problem (1),(2):
Maximize

$$\psi(\beta) = \beta_x \tag{3}$$

subject to

$$\beta_u - \lambda\beta_v \le c_{u,v}, \forall (u,v) \in E^0, \tag{4}$$

where $E^0 = \{e = (u,v) \mid u \in X \setminus \{y\}, v \in X\}$.

An optimal basic solution of the dual problem (3),(4) (if it exists) can be obtained using a special recursive procedure for the calculation $\beta_u^*, u \in X$ starting with $\beta_v^* = 0$ for $v = y$ and $\beta_u = \infty$ for $u \in X \setminus \{y\}$. The optimal vales β_u^* for $u \in X \setminus \{y\}$ can be determined on the basis of the following recursive formula

$$\beta_u = \min_{u \in X(u)} \{\lambda\beta_v + c_{u,v}\}.$$

After $|X|$ iterations of this procedure we determine an optimal basic solution of the problem. An arbitrary directed edge (u,v) for which $\beta_u^* = \lambda\beta_v^* + c_{u,v}$ holds represents a directed edge of an optimal path from x to y. So, the algorithm based on this recursive procedure allows us to solve the considered problems in the general case with the same complexity as the problem with $\lambda = 1$.

If in G it is necessary to determine the optimal paths from every $x \in X \setminus \{y\}$ to y then the tree of these optimal paths can be found using the following linear programming problem: Minimize the objective function (1) subject to

$$\begin{cases} \sum\limits_{e \in E^-(u)} \alpha_e - \lambda \sum\limits_{e \in E^+(u)} \alpha_e = 1, \quad \forall u \in X \setminus \{y\}, \\ \alpha_e \ge 0, \quad \forall e \in E. \end{cases} \tag{5}$$

The dual model for this linear programming problem is the following:
Maximize

$$\psi(\beta) = \sum_{u \in X \setminus \{y\}} \beta_u \tag{6}$$

subject to (6). An optimal basic solution of the dual problem (5),(6) can be found in a similar way as for the previous dual problem.

Now let us consider the problem when the number of the edges in the optimal path is given and it is equal to k. We have shown that for this problem the following linear programming problem can be used:

Minimize

$$\phi(\alpha) = \sum_{e \in E} c_e \alpha_e \tag{7}$$

subject to

$$\begin{cases} \sum\limits_{e \in E^-(u)} \alpha_e - \lambda \sum\limits_{e \in E^+(u)} \alpha_e = 1, & u = x, \\ \sum\limits_{e \in E^-(u)} \alpha_e - \lambda \sum\limits_{e \in E^+(u)} \alpha_e = 0, & \forall u \in X \setminus \{x, y\}, \\ \sum\limits_{e \in E^-(u)} \alpha_e - \lambda \sum\limits_{e \in E^+(u)} \alpha_e = -\lambda^{k-1}, & u = y, \\ \alpha_e \geq 0, \ \forall e \in E. \end{cases} \tag{8}$$

The problem (7),(8) has solutions for arbitrary costs $c_e,\ e \in E$ if in G there exists a directed path that contains k edges. However the optimal solution of this problem may correspond to a directed path that contains directed cycles. Therefore in this case for determining the optimal solution it is a more useful to use a dynamic programming algorithm and a time-expanded network method from [1, 3] considering this problem on a dynamic network with costs $c_e(t) = \lambda^t c_e$ on the edges that depend on time.

3 Conclusion

The considered optimal paths problems in networks with rated transition time costs on the edges generalizes the shortest path problem in a weighted directed graph. For this problem new polynomial time algorithms based on dynamic programming have been elaborated and grounded. The proposed approach can be extended for minimum cost flow problems on networks with rated transition time costs on the edges.

References

[1] Lozovanu D., Pickl S. Optimization and Multiobjective Control of Time-Discrete Systems. Springer, (2009).

[2] Lozovanu D, Pickl S. *Determining Optimal Stationary Strategies for Discounted Stochastic Optimal Control Problems on Networks*, Extended abstracts of 9th CTW workshop on Graphs and Combinatorial Optimization, Cologne, Germany (2010), 115–118.

[3] Lozovanu D, Fonoberova M. *Optimal Dynamic Flows in Networks and Algorithms for Finding Them*, Chapter in the book "Analysis of Complex Networks", (ed. by Dehmer M. and Emmert-Streib F.), Wiley (2009), 377–401.

[4] Puterman, M., Markov Decision Processes: Stochastic Dynamic Programming., John Wiley, New Jersey (2005).

Linearization of ancestral multichromosomal genomes

Ján Maňuch[1,2], Murray Patterson[3], Roland Wittler[4], Cedric Chauve[1], and Eric Tannier[5,6]

[1]Department of Mathematics, Simon Fraser University, Burnaby BC, Canada
[2]Department of Computer Science, University of British Columbia, Vancouver BC, Canada
[3]Centrum Wiskunde & Informatica (CWI), Amsterdam, Netherlands
[4]Center for Biotechnology (CeBiTec), Bielefeld University, 33594 Bielefeld, Germany
[5]INRIA Rhône-Alpes, 655 avenue de l'Europe, F-38344 Montbonnot, France
[6]Lab. de Biométrie et Biologie Évolutive, CNRS and U. de Lyon 1, F-69622 Villeurbanne, France

1 Introduction

Genomes, meant as the linear organization of genes along chromosomes, have been successively modeled by several mathematical objects. Sturtevant and Tan [1] first introduced permutations to study the evolution of genome structure. Starting in the 1980's [2], a large body of work focused on the mathematical and algorithmic properties of such models, including linear and circular genomes [3]. In this framework, hardness results of algorithmic complexity were ubiquitous as soon as three genomes were compared [4].

In order to scale to the dozens of available genomes, another model was needed. Bergeron, Mixtacki and Stoye [5] proposed to use a graph matching between gene extremities to define a genome. Eukaryotes with organelles, or prokaryotes with several replicons, which have not yet been handled explicitly by a formal comparative genomics approach, arguably fit such a model. An unexpected consequence of this relaxation is that the comparison of three genomes with the breakpoint distance proved to be tractable, as an exact optimal median can be computed by solving a maximum weight perfect matching problem [6]. Moreover, the small parsimony problem can be solved for any number of genomes for the Single-Cut and Join (SCJ) distance by Fitch's parsimony algorithm on binary characters [7]. This opened the way to scalable methods at the level of large multispecies datasets.

An additional relaxation allows any graph, and not only a matching, to model genomes. Ancestral genome reconstruction methods often first compute sets of ancestral adjacencies (neighborhood relations between two genes) [8], or intervals (neighborhood relations between an arbitrary number of genes) [9], which result in non-linear structures. This, while unrealistic at a first glance, allows computational breakthroughs, like incorporating duplications and heterogeneous gene content in the framework [10, 11] with polynomial exact methods, and non-linear genomes may help to understand the amount of error in the data [11].

Nevertheless, biological applications in general require linear genomes. The *Linearization Problem* is, given a set of weighted intervals (the weight indicates a confidence value based on phylogenetic conservation of intervals), to find a maximum weight subset which is compatible with a linear structure.

According to the definition of a linear structure, this can be described by some form of problem on an edge-weighted hypergraph $H = (V, E)$, where each vertex $v \in V$ is a gene or gene extremity, and each (weighted) hyperedge $e \in E$ is an interval. If the target linear structure is a genome with several linear chromosomes, then in the case of adjacencies (intervals of size 2), H is just a graph and hence the Linearization Problem is equivalent to the Maximum Weight Vertex-Disjoint Path Cover Problem, so it is NP-complete [12]. For a genome with a single circular chromosome, the Linearization Problem in the case of adjacencies generalizes the Traveling Salesman Problem (TSP), so it is also NP-complete [12].

To the best of our knowledge, there is currently no tractability result known for the Linearization Problem. Currently all methods [8, 10] rely on heuristic or external Traveling Salesman Problem solvers, or branch and bound techniques [9, 13]. Moreover, none of the previously published methods is able to infer multichromosomal genomes, possibly with circular chromosomes, which is the natural model for bacterial genomes with plasmids.

In the present paper, we prove that the Linearization Problem for weighted adjacencies, when ancestral genomes can have several circular and linear chromosomes, is tractable. Note that this variant of the problem is a relaxation of the previous problems, so NP-hardness does not follow from them. We prove this in a more general case, where multiple copies $\mathbf{m}(v)$ of each vertex (gene) v is allowed. Here, instead of a collection of cycles and paths on the vertices, one asks for a collection of cyclic walks on the vertices, where there are at most $\mathbf{m}(v)$ occurrences of any vertex v in all such walks. In the context of genome reconstruction, this allows to model genes with multiple copies in an ancestral genome [14], or to include telomere markers [15].

We show that this corresponds to finding a maximum weight f-matching, which, in turn, is reducible to finding a maximum weight matching. Also, following the complexity pattern already observed with the model of [14], we further show that the Linearization Problem for intervals of size 2 and 3 is NP-complete, even if all intervals have the same weight and all vertices have multiplicity one. We discuss the possibilities that our tractability result opens for ancestral genome reconstruction.

2 Results

In the case of weighted adjacencies, let G be the corresponding (edge-weighted) graph. Let $\mathbf{m} : V(G) \to \mathbb{N}$ specify the maximum copy number (or multiplicity limit) for each vertex of G. We say that G is $\mathbf{m}$-*linearizeable* if there exists a collection of cyclic walks that satisfies the following two conditions: (i) G is a subgraph of the union of cyclic walks; and (ii) the total number of occurrences of each vertex v in all cyclic walks is at most $\mathbf{m}(v)$.

A $2\mathbf{m}$-*matching* of a graph G is a spanning subgraph of G such that the degree of each vertex $v \in V(G)$ is at most $2\mathbf{m}(v)$. The following lemma shows the correspondence between spanning subgraphs of G that are $\mathbf{m}$-linearizeable and $2\mathbf{m}$-matchings of G.

Lemma 2.1. *A spanning subgraph of a graph G is $\mathbf{m}$-linearizeable if and only if it is a $2\mathbf{m}$-matching of G.*

We give a proof sketch. For more details, we refer the reader to a similar proof in [14].

Proof. First, assume a spanning subgraph G' of G is $\mathbf{m}$-linearizeable. Then there is a collection of cyclic walks satisfying conditions (i) and (ii). Since each vertex v appears at most $\mathbf{m}(v)$ times in these cyclic walks and each occurrence has only two neighbors, the degree of v in G' is at most $2\mathbf{m}(v)$. Hence, G' is a $2\mathbf{m}$-matching of G.

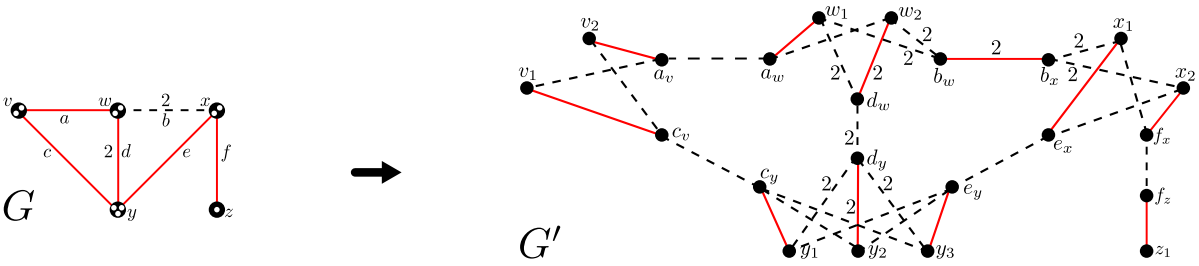


Figure 1: Reduction used to transform the maximum weight f-matching problem to the maximum weight matching problem. Edge weights are all one, unless otherwise indicated, and f is given by the white dots inside the nodes. The total edge weight in G is 8. The solid edges show a maximum weight f-matching in G ($w = 6$), and a corresponding maximum weight matching in G' (of weight $6 + 8 = 14$).

Conversely, assume G' is a $2\mathbf{m}$-matching of G. If $\deg_{G'}(v) < 2\mathbf{m}(v)$ for some $v \in V(G')$, then we add a new vertex v_0 and for each v such that $\deg_{G'}(v) < 2\mathbf{m}(v)$, we add a new edge $\{v_0, v\}$ with multiplicity $2\mathbf{m}(v) - \deg_{G'}(v)$ to G'. Since now every vertex of G' has even degree, each component C of G' is Eulerian, *i.e.,* there is a cyclic walk which contains all edges of C, and each $v \in V(C)$ appears exactly $\mathbf{m}(v)$ times in the walk. If C does not contain v_0 then this cyclic walk satisfies conditions (i) and (ii) for vertices in $V(C)$. If C contains v_0, then after omitting all occurrences of v_0 we obtain a cyclic walk satisfying conditions (i) and (ii) for vertices in $V(C)$. Hence, G' is $\mathbf{m}$-linearizeable. □

It follows that maximum weight $\mathbf{m}$-linearizeable subgraphs of G correspond to maximum weight $2\mathbf{m}$-matchings of G. Next, we give an algorithm for finding a maximum weight f-matching of G with running time $(O((|V(G)| + |E(G)|)^{3/2}))$, where $f : V(G) \to \mathbb{N}$. We will use a more general form of Tutte's reduction for reducing the maximum weight f-matching problem to the maximum weight matching problem similar to the ones presented in [16, 17].

Given edge weighted graph G and function f, construct G' as follows: For all x in $V(G)$, let $x_1, x_2, \ldots, x_{f(x)}$ be in $V(G')$; and for all $e = \{x, y\}$ in $E(G)$, let e_x and e_y be in $V(G')$. Now, for all $e = \{x, y\}$ in $E(G)$, let $\{x_1, e_x\}$, $\ldots$, $\{x_{f(x)}, e_x\}$, $\{e_x, e_y\}$, $\{y_1, e_y\}$, $\ldots$, $\{y_{f(y)}, e_y\}$ be edges of G', and all these edges have weight $w(e)$. This reduction is illustrated in Figure 1.

Property 2.2. *There is an f-matching in G with weight w if and only if there is a matching in G' with weight $w + W$, where $W = \sum_{e \in E(G)} w(e)$.*

An unweighted version of this property was shown in [17]. The weighted version can be shown in the same way, and hence, we omit the proof.

Since a maximum weight matching can be found in time $O(\sqrt{|V(G')|} \cdot |E(G')|)$ [18], we have time $O((|V(G)| + |E(G)|)^{3/2})$ algorithms for finding a maximum weight f-matching and for finding a maximum weight $\mathbf{m}$-linearizeable subgraph, in the case of adjacencies.

Finally, we show that if we generalize to intervals of size 2 and 3, this version of the Linearization Problem is NP-complete, even when all multiplicities and weights are equal to one. We do this by defining what it means for a hypergraph $H = (V, E)$ to be $\mathbf{m}$-linearizable, and then it suffices to prove the following theorem for unweighted hypergraphs.

Theorem 2.3. *Finding a 1-linearizable subgraph of hypergraph H with the maximum number*

of edges is NP-complete.[1]

3 Conclusions

There are exact optimization [8, 10, 11] or empirical [9] fast methods to construct ancestral adjacencies which do not necessarily form a linear signal. But to date, all linearization methods were heuristics or calls to TSP solvers [8, 10]. Moreover, no method is currently adapted to reconstruct bacterial ancestral genomes with plasmid(s), which is common in the living world.

It is not the first time that a slight change in the formulation of a problem dramatically changes its computational status [13]. Even if such a relaxation is less realistic in certain contexts, solving the relaxation can also help to approach efficiently the constrained problem, like for Double-Cut and Join (DCJ) and inversions/translocations, for example [19]. More generally, 2-factors (spanning subgraphs composed of collections of vertex-disjoint cycles) have been used to approximate Traveling Salesman solutions [16], so genomes composed of several circular chromosomes can be a way to approximate solutions for linear genomes.

References

[1] A.H. Sturtevant and C.C. Tan. The comparative genetics of drosophila pseudoobscura and drosophila melanogaster. *Journal of Genetics*, 34:415–432, 1937.

[2] G. A. Watterson, *et al.* The chromosome inversion problem. *Journal of Theoretical Biology*, 99:1–7, 1982.

[3] G. Fertin, *et al.* *Combinatorics of genome rearrangements.* MIT press, 2009.

[4] D. Bryant. The complexity of the breakpoint median problem. Technical Report CRM-2579, Centre de Recherches Mathématiques, Université de Montréal, 1998.

[5] A. Bergeron, J. Mixtacki, and J. Stoye. A unifying view of genome rearrangements. In *Algorithms in Bioinformatics, Proceedings of WABI'06*, volume 4175 of *LNBI*, pages 163–173, 2006.

[6] E. Tannier, C. Zheng, and D. Sankoff. Multichromosomal median and halving problems under different genomic distances. *BMC Bioinformatics*, 10:120, 2009.

[7] P. Feijao and J. Meidanis. SCJ: a breakpoint-like distance that simplifies several rearrangement problems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8:1318–1329, 2011.

[8] J. Ma, *et al.* Reconstructing contiguous regions of an ancestral genome. *Gen. Res.*, 16:1557–1565, 2006.

[9] C. Chauve and E. Tannier. A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genomes. *PLoS Cmpt. Biol.*, 4:e1000234, Nov 2008.

[1] proof omitted for space

[10] J. Ma, *et al.* Dupcar: reconstructing contiguous ancestral regions with duplications. *Journal of Computational Biology*, 15:1007–1027, October 2008.

[11] S. Bérard, C. Gallien, B. Boussau, G. Szöllősi, V. Daubin, and E. Tannier. Evolution of gene neighborhoods within reconciled phylogenies. *Bioinformatics*, 28(18):i382–i388, 2012.

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

[13] E. Tannier. Yeast ancestral genome reconstruction: the possibilities of computational methods. In *Comparative Genomics, Proceedings of RECOMB-CG'09*, volume 5817 of *LNCS*, pages 1–12, 2009.

[14] R. Wittler, J. Maňuch, M. Patterson, and J. Stoye. Consistency of sequence-based gene clusters. *Journal of Computational Biology*, 18(9):1023–1039, September 2011.

[15] C. Chauve, J. Maňuch, M. Patterson, and R. Wittler. Tractability results for the consecutive-ones property with multiplicity. In *Proceedings of CPM'11*, volume 6661 of *LNCS*, pages 90–103, 2011.

[16] L. Lovasz and M. D. Plummer. *Matching Theory*, volume 29 of *Annals of Discr. Math.* N. Holland, 1986.

[17] A. Dessmark, *et al.* On parallel complexity of maximum f-matching and the degree sequence problem. In *Proceedings of the 19th MFCS*, pages 316–325. Springer-Verlag, 1994.

[18] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of FOCS'80*, pages 17–27, 1980.

[19] I. Miklós and E. Tannier. Approximating the number of double cut-and-join scenarios. *Theoretical Computer Science*, 439:30–40, 2011.

A Branch-and-Cut algorithm for the Angular TSP

Isabel Méndez-Díaz[1], Federico Pousa[1], and Paula Zabala[1,2]

[1]Departamento de Computación - FCEN - Universidad de Buenos Aires
[2]Consejo Nacional de Investigaciones Científicas y Técnicas

1 Introduction

The *Angular Travelling Salesman Problem* (AngTSP) is a generalization of the classical Traveling Salesman Problem (TSP) where the objective is to minimize the total distance travelled and the total direction change of a tour visiting a set of points in the Euclidean space.

More formally, consider a complete weighted graph $G = (V; E)$, with $V = \{v_1, \ldots, v_n\}$ the set of nodes representing the set points, and E the set of arcs fully connecting the nodes in V. For each $(v_i v_j) \in E$, we define c_{ij} as the euclidean distance between points v_i and v_j. Furthermore, given two edges $(v_i v_j)$ and $(v_j v_k)$ incidents to a same vertex v_j, it is defined the angle between them, $\delta_{ijk} = arccos\left(< \frac{v_j - v_i}{\|v_j - v_i\|}, \frac{v_k - v_j}{\|v_k - v_j\|} >\right)$, as the absolute value of the direction change of the movement of travelling from v_i to v_k, going through v_j. The AngTSP involves finding a hamiltonian tour that minimize the sum of distances and angles.

There are many real world applications where the rotation angle is not negligible, producing a substantial increment in the total cost or when it is not possible to perform certain rotations due to mechanical restrictions. One of such cases is the trajectories design in mobile robotics which is introduced in [1]. The problem is presented in two versions. On the one hand, the pure angular version, in which the objective function only takes into account the sum of the angles turned in the circuit. On the other hand, it is proposed a version in which the objective function is a compromise between minimizing the distances and the angles.The authors prove that the AngTSP belongs to the NP-Hard class and that it can be approximated in a $\mathcal{O}(log\ n)$ ratio in polynomial time. Recently, in [2] the problem is studied from a practical point of view. The authors present a two phase algorithm for the Travelling Salesman Problem for Dubins Vehicles. The first phase, or *tour stage*, is to determine the order in which the points will be visited. The second phase, or *trajectory stage*, consists in determining the trajectory between every pair of points. The AngTSP is used in this work to solve the first stage.

In this paper we study the general version of the problem where the objective function minimizes a weighted combination between the total distance travelled and the total change of directions. The goal is to design an exact algorithm for the AngTSP based on Integer Linear Programming models.

This work was partially supported by UBACYT 20020100100666, PICT 2010-304 and 2011-817

2 Models

In [2], the authors present a model for the AngTSP based on the Dantzing, Fulkerson and Johnson model for the TSP. Following this idea, we study the performance of other models which are well known in the literature for the TSP (see, e.g., [3]) once adapted to the AngTSP. We consider the following models for the TSP: Dantzing, Fulkerson and Johson(DFJ), Sherali and Driscoll(SD), Sarin, Sherali and Bhootra(SSB), Sherali, Sarin y Tsai(SST), Gavish and Graves(GG) and Fox, Gavish and Graves(FGG).

Preliminary computational results, where we analyze different aspects in solving random instances with 10 to 50 nodes, showed that the DFJ model adapted for the AngTSP shows the best performance in every aspect, only comparable with the GG model.

The AngTSP-DFJ model consider the binary variables x_{ij} that take value 1 iff the tour goes along edge (v_iv_j) and, to account for angles, we introduce binary variables y_{ijk} which take value 1 if the tour goes along edges (v_iv_j) and (v_jv_k). The formulation is the following:

$$\min \alpha \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} c_{ij}x_{ij} + \beta \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{\substack{k=1 \\ k\neq i,j}}^{n} \delta_{ijk}y_{ijk}$$

$$\text{subject to: } \sum_{j=1,j\neq i}^{n} x_{ij} = 1, \quad i = 1,\ldots,n \tag{1}$$

$$\sum_{i=1,i\neq j}^{n} x_{ij} = 1, \quad j = 1,\ldots,n \tag{2}$$

$$\sum_{i,j\in S} x_{ij} \leq |S| - 1, \quad S \subseteq \{2,\ldots,n\}, 2 \leq |S| \leq n-1 \tag{3}$$

$$y_{ijk} \geq x_{ij} + x_{jk} - 1 \quad i,j,k = 1,\ldots,n \quad i \neq j \neq k \tag{4}$$

$$x_{ij}, y_{ijk} \quad i,j,k = 1,\ldots,n \;\; i \neq j \neq k \tag{5}$$

where $\alpha, \beta \geq 0$ $(\alpha + \beta = 1)$, are the relative weights of each objective depending on whether we want focus on angles, probably in detriment of the total distance, or on solutions where the most important aspect is the distance travelled, and the rotation angle are less important. Constraints (1),(2) and (3) are the well known degree and subtour elimination constraints. Inequalities (4) establish that if both x_{ij} and x_{jk} have a value of one, then the variable y_{ijk} must take a value one as well, meaning that an angle is considered whenever its two edges are used.

The previous results encourage us to strengthen the model in order to develop an exact algorithm based on this formulation.

3 Polyhedral analysis

The relationship between x and y variables tends to be weak in the sense that lets any y variable to have a value not depending on the values of its edges. Even though this situation is fixed by the objective function since it is a minimization problem with non-negative coefficients, we strengthen the model by adding the following equalities named *XYRel*:

$$\sum_{k=1,k\neq i,j}^{n} y_{ijk} = x_{ij}, \quad i,j = 1,\ldots,n \quad i \neq j \tag{6}$$

$$\sum_{k=1,k\neq i,j}^{n} y_{kij} = x_{ij}, \quad i,j = 1,\ldots,n \quad i \neq j \tag{7}$$

The first equality indicates that if the edge $(v_i v_j)$ is being used, then there must be exactly one angle that goes through v_i and v_j, and then goes to some other point v_k. The second equality is analogous to the first one but in this case v_i and v_j are the end points of the angle.

With these equalities we now prevent an angle variable to take a positive value if either of the edge variables related to it has zero value.

We have proved that, for $n \geq 7$, the minimal system is characterized by the TSP degree and *XYRel* equalities just removing one from (1), one from (6) and $n-1$ from (7).

By incorporating equations (6) and (7) to the model, inequalities (4) become redundant and therefore can be excluded. This produces a tighter formulation for the AngTSP and, in addition, reduces the overall number of constraints.

We now introduce new valid inequalities to strengthen the notion of angles along the tour.

The first inequality family states that for every 3 points, there must be at most one angle variable relating them.

$$y_{ijk} + y_{ikj} + y_{jik} + y_{jki} + y_{kij} + y_{kji} \leq 1 \quad \forall i,j,k = 1,\ldots,n \quad i \neq j \neq k \tag{8}$$

The next one states that for every pair of nodes there must be at most one variable indicating that they are one or two positions apart.

$$x_{ij} + \sum_{k=1,k\neq i,j}^{n} y_{ikj} + x_{ji} + \sum_{k=1,k\neq i,j}^{n} y_{jki} \leq 1 \quad \forall i,j = 1,\ldots,n \quad i \neq j \tag{9}$$

The following constraint represents all the possible positions that a fourth point can take related to some other three.

$$y_{ijk} \leq y_{lij} + y_{jkl} + \sum_{\substack{r,s=1 \\ r,s\neq i,j,k,l}}^{n} y_{rls} \quad \forall i,j,k,l =,\ldots,n \quad i \neq j \neq k \neq l \tag{10}$$

Finally, the next inequality forces that in case the tour uses edge $(v_i v_j)$, then node v_k could not be at right and left at the same time.

$$y_{ijk} + y_{kij} \leq x_{ij} \quad \forall i,j,k = 1,\ldots,n \quad i \neq j \neq k \tag{11}$$

4 Branch-and-Cut and computational results

In order to evaluate the strength of these inequalities, we test them in a B&C framework. Initially, we consider a LP relaxation with the minimal system. The subtour constraints are not include at once, they are treated as cutting planes during the process. Since the valid inequality families are polynomial sized, the separation phase is done by enumeration.

We also develop initial heuristics based on a *farthest insertion* scheme with a posterior 3-opt local search approach, that show to be very effective in finding primal bounds. Finally, we

overload the branching rule by specifying a priority criteria. We prioritize the branching on an edge variable over an angle one because fixing an edge is much less restrictive than fixing an angle, leading to more balanced trees.

The algorithm is coded using CPLEX 12.3 and the experiments are run on a workstation with an Intel(R) Core(TM) i7 CPU (3.40GHz) and 16 Gb of RAM. We consider a set of random instances varying the number of nodes which are grouped in four sets with 7 instances each. We set a time limit of 1200 seconds for each instance.

In Table 1 we report the average time when the algorithm finishes before the time limit, otherwise we report the average percentage final gap (100(UB - LB)/LB). A number between parenthesis represents the number of instances considered to compute the average.

Instances			Nodes			
			10-19	20-29	30-39	40-49
$\alpha = 1.0$	CPLEX	time	0.07(7)	0.7(7)	3.11(7)	12.32(7)
		gap	0%	0%	0%	0%
	B&C	time	0.04(7)	0.24(7)	1.3(7)	4.73(7)
		gap	0%	0%	0%	0%
$\alpha = 0.75$	CPLEX	time	111.61(7)	568.58(2)	****	****
		gap	0%	9.87%(2)	29.01%(7)	38.96%(7)
	B&C	time	0.08(7)	8.04(7)	206.63(7)	400.93(5)
		gap	0%	0%	0%	2.56%(2)
$\alpha = 0.5$	CPLEX	time	139.59(5)	****	****	****
		gap	27.82%(2)	43.70%(7)	58.36%(7)	65.83%(7)
	B&C	time	0.3(7)	45.77(7)	506.34(6)	1413.96(1)
		gap	0%	0%	2.89%(1)	8.38%(6)
$\alpha = 0.25$	CPLEX	time	572.01(3)	****	****	****
		gap	39.77%(4)	71.95%(7)	81.31%(7)	88.38%(7)
	B&C	time	0.84(7)	84.92(7)	425.19(4)	****
		gap	0%	0%	3.19%(3)	9.86%(7)
$\alpha = 0.0$	CPLEX	time	535.78(3)	****	****	****
		gap	61.71%(4)	95.46%(7)	100.0%(7)	100.0%(7)
	B&C	time	1.94(7)	55.26(7)	449.46(6)	1280.37(1)
		gap	0%	0%	4.49%(1)	9.83%(6)

The algorithm designed shows very good results, outperforming by far the CPLEX-default implementation. The polyhedral analysis is a key factor for the time required to solve each instance. However, from the results we can see that instances with a higher number of nodes remains as a challenge, at least in the fixed time limit. A deeper observation of the results shows that the difficulties tends to come from the dual bound, as the optimal value is usually obtained in few nodes by the heuristic. This lets us as future work the search of further valid inequalities to strengthen the formulation.

References

[1] Alok Aggarwal, Sanjeev Khanna, Rajeev Motwani, and Baruch Schieber. The angular-metric traveling salesman problem. *In Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 29:221–229, 1997.

[2] André César Medeiros and Sebastián Urrutia. Discrete optimization methods to determine trajectories for dubins vehicles. *Electronic Notes in Discrete Mathematics*, 36:17–24, 2010.

[3] Temel Oncan, Kuban Altinel, and Gilbert Laporte. Invited review: A comparative analysis of several asymmetric traveling salesman problem formulations. *Comput. Oper. Res.*, 36(3):637–654, March 2009.

Star-shaped mediation in influence games*

Xavier Molinero[1], Fabián Riquelme[2], and Maria Serna[2]

[1]Department of Applied Mathematics III
Universitat Politècnica de Catalunya, Manresa (Spain). E-mail:xavier.molinero@upc.edu
[2]Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona (Spain). E-mail:{farisori,mjserna}@lsi.upc.edu

We are interested in analyzing the properties of multi-agent systems [13] where a set of agents have to take a decision among two possible alternatives with the help of the social environment or network of the system itself. The ways in which people influence each other through their interactions in a social network and, in particular, the social rules that can be used for the spread of influence have been proposed in an alternative simple game model [11]. However not all individuals play the same role in the process of taking a decision. In this paper we are interested in formalizing and analyzing the simple game model that results in a *mediation system*. In this scenario we have a social network together with an external participant, *the mediator*. The mediator can interact, in different degrees, with the agents and thus help to reach a decision.

1 Preliminaries

The area of *simple games*, a subfamily of cooperative game theory, provides a formal model for analyzing decision systems [14]. A *simple game* is a set system providing the coalitions that can made an alternative pass. The ways in which people influence each other through their interactions in a social network have received a lot of attention in the last decade with important links to sociology, economics, epidemiology, computer science, and mathematics [1, 9, 6]. Agents face the choice of adopting a specific product or not, choose among competing programs from providers of mobile telephones, having the option to adopt more than one product at an extra cost, etc. Those decisions have to be taken by individuals participating in a social networks. A social network can be represented by a graph where each node is an agent, individual or player, and each edge represents the degree of influence of one agent over another one. Several "motivations" (ideas, trends, fashions, ambitions, rules, etc.) can be initiated by one or more agents and eventually be adopted by the system. The mechanism defining how these motivations are propagated within the network, from the influence of a small set of nodes initially *motivated*, is called the model for *influence spread*. In this subject, motivated by viral marketing and other applications, has been established the *influence maximization problem*

*This work is partially supported by 2009SGR1137 (ALBCOM). First author is also partially funded by Grant MTM2012–34426, the second one by Grant BecasChile of "National Commission for Scientific and Technological Research of Chile" (CONICYT), and the third one by Grant TIN2007–66523 (FORMALISM).

[5, 12], and the *linear threshold* and *independent cascade models* for influence spread [10] among other ones [2, 4, 1]. In such a setting there is also work done towards analyzing the problem from the point of view of non-cooperative game theory [8].

In a recent paper we proposed a simple game based on a model of spread of influence in a social network, where influence spreads according to the linear threshold model, the so-called *influence games* [11]. Now we extend this model in order to incorporate two different levels of influence spread, the one taking place at the social network and the other one exerted by a *mediator*, again analyzed as a simple game.

2 Definitions and Results

For simple games, we follow definitions and notation from [14]. As usual, given a finite set $N = \{1, \ldots, n\}$ of *players* or *voters*, $\mathcal{P}(N)$ denotes its *power set*. A family of subsets $\mathcal{W} \subseteq \mathcal{P}(N)$ is said *monotonic* when $\forall X \in \mathcal{W}$, if $X \subseteq Z$, then $Z \in \mathcal{W}$.

Definition 2.1 *A* simple game *is a tuple* $(N, \mathcal{W})$ *where* N *is called the* grand coalition*, and* $\mathcal{W}$*, the set of* winning coalitions*, is a monotonic family of subsets of* N*.*

In the context of simple games, subsets not appearing in $\mathcal{W}$ are called *losing coalitions*, and a *minimal winning (maximal losing) coalition* is a winning coalition such that by removing (adding) one player results in a losing (winning) coalition. We use $\mathcal{W}$, $\mathcal{L}$, $\mathcal{W}^m$ and $\mathcal{L}^M$ to denote the sets of winning, losing, minimal winning and maximal losing coalitions. Any of those sets determine uniquely the game and constitute the usual forms of representation for simple games [14]. Before introducing formally the family of influence games we need to define a family of labeled graphs based on the *linear threshold model*. We use standard graph [3] and computational complexity [7] notation. Note that *player* (element of N), *coalition* (subsets of N), *winning* and *losing* for simple games usually mean, respectively, *agent* (vertex of V), *team* (subsets of V), *successful* and *unsuccessful* for influence games.

Definition 2.2 *An* influence graph *is a tuple* $(G; f)$*, where* $G = (V, E)$ *is a labeled and directed graph (without loops) —with* V *its set of vertices and* E *its set of edges— and* $f : V \to \mathbb{N}$ *is a labeling function that quantify how influenceable each player or agent is.*

Given an influence graph $(G; f)$ and an *initial activation* set $X \subseteq V$, the *spread of influence* $F : V \to V$ is a function such that $F(X)$ is the set of agents activated by an iterative process. Initially the vertices in X are activated, i.e., $X \subseteq F(X)$. Let be $\mathtt{In}(u) = \{v \,|\, (v, u) \in E\}$, at each step any $u \notin F(X)$ such that $|\mathtt{In}(u) \cap F(X)| \geq f(u)$ is added to $F(X)$. The process stops when no additional activation occurs.

Definition 2.3 *An* influence game *is a tuple* $(G; f, q)$*, where* $(G; f)$ *is an influence graph and* $q \geq 0$ *is an integer, the quota. A coalition* $X \subseteq V$ *is winning iff* $|F(X)| \geq q$*.*

In this paper we incorporate another influence layer. On the bottom layer the influence is exerted among the agents and on another layer the relationship of influence between the agents and an external mediator is kept. The mediator can exert influence on some nodes and accept advice from others, thus introducing a modification on the way that influence spreads through the network. We model the society by a set of nodes V where the relation with the mediator can be expressed by three disjoint sets $A, B, C \subseteq V$, where A is formed by the agents

that can influence the mediator but are not influenced by him, B contains those agents that influence and can be influenced by the mediator and, finally, C is formed by the agents that can be influenced by the mediator but cannot exert influence. This kind of relationship can be understood by means of a star graph $(V \cup \{c\}, E)$ in which the mediator correspond to an additional vertex $c \notin V$ and where $E = \{(u, c) \mid u \in A \cup B\} \cup \{(c, v) \mid v \in B \cup C\}$.

In the following definition we assume that $A, B, C \subseteq V$ are disjoint.

Definition 2.4 *In a* star influence game $\Gamma(V, A, B, C, k, q)$*, where* $k, q \in \mathbb{N}$*, a coalition* $X \subseteq V$ *is winning iff either (1)* $|X| \geq q$ *or (2)* $|X \cap (A \cup B)| \geq k$ *and* $|X \cup B \cup C| \geq q$*.*

In a star mediation influence game $\Gamma(V, E, f, A, B, C, k, q)$*, where* $((V, E); f)$ *is an influence graph and* $k, q \in \mathbb{N}$*, a coalition* $X \subseteq V$ *is winning iff either (1)* X *is winning in the influence game* $((V, E); f, q)$ *or (2)* $|X \cap (A \cup B)| \geq k$ *and* $|X \cup B \cup C|$ *is winning in the influence game* $((V, E); f, q)$*.*

Theorem 2.5 *Given a star influence game, determine* $\mathcal{W}^m$ *or* $\mathcal{L}^M$ *can be done in incremental-polynomial time.*

Next we also analyze some properties of an influence game, conditions under which the system can reach an alternative, as decision problems from a computational point of view:

IsProper: Determine whether the complement of any winning coalition is losing.
IsStrong: Determine whether the complement of any losing coalition is winning.
IsDecisive: Determine whether a coalition is winning iff its complement is losing.

Moreover, other problems consider properties of an agent i with respect to an influence game:

IsDummy: In any winning coalition X including i, is $X \setminus \{i\}$ also winning?
IsPasser: Is any coalition including i winning?
IsVetoer: Is any coalition not including i losing?
IsDictator: Does the set of winning coalitions coincide with the set of coalitions including i?

Theorem 2.6 *Given a star influence game, the problems* IsProper*,* IsStrong*,* IsDecisive*,* IsDummy*,* IsPasser*,* IsVetoer *and* IsDictator *are in P.*

Theorem 2.7 *Given a star mediation influence game, the problems* IsProper*,* IsStrong*,* IsDecisive *and* IsDummy *are coNP-complete, while* IsPasser*,* IsVetoer *and* IsDictator *are in P.*

The above results are obtained by providing characterizations of the properties in terms of q, k, $|A|$, $|B|$ and $|C|$.

3 Future work

There remain many open problems, in particular to analyze whether different conditions on the influence relationship among the agents and the mediator can lead to characterizations or polynomial time algorithms for the problems considered in this paper or for other problems of interest, like some coming from social choice theory [15]. We have analyzed the results of the superposition of two influence networks in which one of them is restricted to be a star a future line of research is to analyze influence games resulting from the superposition of two or more complex social networks. Another area of interest is to consider weighted influence networks with edge weights or networks with several mediators related to different parts of the network with or without common agents.

References

[1] K. R. Apt and E. Markakis. Diffusion in social networks with competing products. *In SAGT 2011, LNCS*, 6982:212–223, 2011.

[2] S. Bharathi, D. Kempe, and M. Salek. Competitive infuence maximization in social networks. *In WINE 2007, LNCS*, 4858:306–311, 2007.

[3] B. Bollobás. *Modern graph theory*. Springer, 1998.

[4] A. Borodin, Y. Filmus, and J. Oren. Threshold models for competitive infuence in social networks. *In WINE 2010, LNCS*, 6484:539–550, 2010.

[5] P. Domingos and M. Richardson. Mining the network value of customers. *In ACM KDD 2001*, pages 57–66, 2001.

[6] D. Easley and J. Kleinberg. *Networks crowds and markets*. Cambridge University Press, 2010.

[7] M. R. Garey and D. S. Johnson. *Computers and intractability, a guide to the theory of NP-Completness*. W.H. Freeman and Company, New York, USA, 1999.

[8] M. T. Irfan and L. E. Ortiz. *A game-theoretic approach to influence in networks*. Association for the Advancement of Artificial Intelligence, 2011.

[9] M. Jackson. *Social and economic networks*. Princeton University Press, 2008.

[10] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. *In ACM KDD 2003*, pages 137–146, 2003.

[11] X. Molinero, F. Riquelme, and M. J. Serna. Social influence as a voting system: a complexity analysis of parameters and properties. *CoRR abs/1208.3751*.

[12] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. *In ACM KDD 2002*, pages 61–70, 2002.

[13] Y. Shoham and K. Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, Cambridge, UK, 2009.

[14] A. Taylor and W. Zwicker. *Simple games: Desirability relations, trading, pseudoweightings*. Princeton University Press, New Jersey, 1st edition, 1999.

[15] R. van den Brink, A. Rusinowska, and F. Steffen. Measuring power and satisfaction in societies with opinion leaders: Dictator and opinion leader properties. *Homo Oeconomicus*, 28:161–185, 2011.

Characterising subclasses of perfect graphs with respect to partial orders related to edge contraction

Haiko Müller[1] and Samuel Wilson[1]

[1]School of Computing, University of Leeds, UK

Characterising graph classes by forbidding a set of graphs has been a common feature in graph theory for many years. Up to date many classes have been characterised with respect to the induced subgraph relation and there are a plethora of results regarding graph classes closed under the minor relation. We consider the characterisation of graph classes closed under partial orders including edge contraction as an operation. More concretely we consider contraction minors and a newly defined partial order, under which many of the graph classes considered here can be characterised by a finite minimal forbidden set.

1 Introduction

The characterisation of graph classes by forbidding a set of graphs has been extensively studied providing characterisations for many graph classes, especially surrounding the class of perfect graphs. It is well known that any class closed under a partial order admits a characterisation by forbidding a set of graphs but finding the forbidden set is harder. With respect to the induced subgraph relation many of the classes surrounding perfect graphs have a description of their forbidden set but little effort has gone into considering a description with respect to different partial orders.

The study of partial orders on the set of all graphs has become increasingly popular thanks to the graph minor theorem by Robertson and Seymour. In the series of over 20 papers they have shown a number of results for classes closed under the minor relation, perhaps most interestingly that for any graph class closed under the minor relationship the minimal forbidden set is finite. This result is shown by proving that the relation forms a well-quasi ordering on the set of all graphs and therefore demonstrating that no infinite antichains exists. The graph minor theorem is a celebrated result but many of the subclasses of perfect graphs are not closed under the minor relation so the result does not apply. In addition there is no similar result for other partial orders under which the subclasses of perfect graphs are closed.

This motivates the search for alternative characterisations of graph classes. The contraction minor relation is an interesting partial order as the complexity for recognising if one graph is contained in another with respect to the contraction minor relation is an open problem,

The research presented in this paper is work towards the Ph.D of the second author, who is funded by EPSRC Doctoral Training Grant.

unlike for the minor relation where the graph containment problem is known to belong to the complexity class FPT. We characterise a number of well studied graph classes by forbidding a set of graphs with respect to the contraction minor relation, we then show an interesting connection between the minimal forbidden set with respect to the induced subgraph relation and the minimal forbidden set for a newly defined partial order ($\leqslant_{\mathrm{mw}}$), for which most classes considered here have a finite minimal forbidden set.

2 Definitions

Here we consider only finite simple undirected graphs. A graph G is a contraction minor of a graph H denoted $G \leqslant_{\mathrm{c}} H$ if there exists a set of edges $U \subseteq E(H)$ such that the contraction of the edges in U results in a graph isomorphic to G. For completeness we include the definition of the induced subgraph relation, a graph G is an induced subgraph of H denoted $G \leqslant_{\mathrm{i}} H$ if there exists a set of vertices $U \subseteq V(H)$ such $H[V(H) \setminus U]$ is isomorphic to G.

Every class $\mathcal{C}$ of graphs closed under a partial order $\leqslant$ is characterised by a set $\mathcal{F} = \{H \mid H \notin \mathcal{C} \wedge \forall G\,(G \leqslant H \Rightarrow G \simeq H \vee G \in \mathcal{C})\}$ of minimal forbidden graphs, where a partial order can not be determined from the context it is explicitly given as a subscript, e.g. $\mathcal{F}_{\mathrm{c}}$ denotes the minimal forbidden set with respect to contraction minors. For a set $\mathcal{F}$ of graphs, the class of $\mathcal{F}$-free graphs is $\{G \mid \forall H \in \mathcal{F}\,(H \not\leqslant G)\}$. This class is closed under $\leqslant$ and $\mathcal{F}$ contains all of the minimal forbidden graphs.

We define the following graphs classes adopted from [1];

$$\mathcal{C}_n = \{C_k \mid k \geq n\} \qquad \mathcal{D}_n = \{2K_1 \bowtie kK_1 \mid k \geq n\}$$
$$\mathcal{K}_n = \{kK_1 \mid k \geq n\} \qquad \mathcal{W}_n = \{C_4 \bowtie kK_1 \mid k \geq n\}$$
$$W_k = C_k \bowtie K_1$$

We define a new partial order denoted $\leqslant_{\mathrm{mw}}$ where $G \leqslant_{\mathrm{mw}} H$ if H can be transformed into a graph isomorphic to G by a series of edge contractions and inverse false twin operations where the inverse false twin operation is defined as follows; let G be a graph and let $u, v \in V(G)$ then u, v are false twins if the neighbourhood of u and v are equal and $uv \notin E(G)$. The inverse false twin operation allows for the removal of u.

3 Contribution

We show a number of results for characterising graph classes related to perfect graphs with respect to contraction minors and with respect to $\leqslant_{\mathrm{mw}}$.

Theorem 3.1. *Let G be a connected graph, we show the following conditions are equivalent;*

(i) G is a chordal graph.

(ii) G does not contain $\{C_n \mid n \geq 4\}$ with respect to $\leqslant_{\mathrm{i}}$ [6].

(iii) G does not contain $\mathcal{D}_2 \cup \mathcal{W}_0$ with respect to $\leqslant_{\mathrm{c}}$.

(iv) G does not contain $\{C_4, C_4 \bowtie K_1\}$ with respect to $\leqslant_{\mathrm{mw}}$.

It is well known that the class of split graphs is the intersection of the classes chordal and co-chordal. With respect to contraction minors and $\leqslant_{\mathrm{mw}}$ the set of minimal forbidden graphs for the class of co-chordal graphs is infinite since $\{\overline{C_n} \mid n \geq 6\}$ forms an antichain. Each element of this antichain is comparable to W_4.

Theorem 3.2. *Let G be a connected graph, we show the following conditions are equivalent;*

(i) G *is a split graph.*

(ii) G *does not contain* $\{2K_2, C_4, C_5\}$ *with respect to* $\leqslant_{\mathrm{i}}$ [2].

(iii) G *does not contain* $\mathcal{D}_2 \cup \mathcal{W}_0 \cup \{P_5, \overline{P}, 2K_2 \bowtie K_1\}$ *with respect to* $\leqslant_{\mathrm{c}}$.

(iv) G *does not contain* $\{C_4, W_4, P_5, \overline{P}, 2K_2 \bowtie K_1\}$ *with respect to* $\leqslant_{\mathrm{mw}}$ (Figure 1).

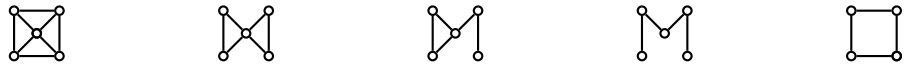


Figure 1: $W_4, 2K_2 \bowtie K_1, \overline{P}, P_5, C_4$: Contraction minimal non-split graphs.

Theorem 3.3. *Let G be a connected graph, we show the following conditions are equivalent;*

(i) G *is a cograph.*

(ii) G *does not contain* P_4 *with respect to* $\leqslant_{\mathrm{i}}$ [1, Theorem 11.3.3].

(iii) G *does not contain* $\{P_4, P_4 \bowtie K_1, \overline{P_5}, C_5, \overline{C_6}\}$ *with respect to* $\leqslant_{\mathrm{mw}}$ (Figure 2).

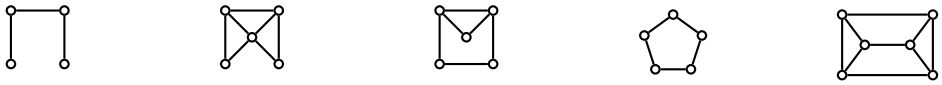


Figure 2: $P_4, P_4 \bowtie K_1, \overline{P_5}, C_5, \overline{C_6}$: Contraction minimal non-cographs

From the above results we distil a general relationship between $\mathcal{F}_{\mathrm{i}}$ and $\mathcal{F}_{\mathrm{mw}}$ for any class $\mathcal{C}$ closed under vertex deletion and edge contraction.

Theorem 3.4. *Any class $\mathcal{C}$ which has a finite forbidden set with respect to induced subgraphs and is closed under edge contraction has a finite forbidden set with respect to $\leqslant_{\mathrm{mw}}$ and the order of the largest forbidden graphs is bounded.*

Proof. Let $\mathcal{C}$ be a class closed under vertex deletion and edge contraction and let $\mathcal{F}_{\mathrm{i}}, \mathcal{F}_{\mathrm{mw}}$ denote the minimal forbidden sets. Let $H \notin \mathcal{C}$ which implies $\exists F \in \mathcal{F}_{\mathrm{i}} \mid F \leqslant_{\mathrm{i}} H$. Contracting the edges $\{uv \mid u, v \in \{V(H) \setminus V(F)\}\}$ leaves an independent set S and a copy of F with some edges between S and $V(F)$. As $\leqslant_{\mathrm{mw}}$ allows the removal of false twins the number of additional vertices is equal to the number of subsets of vertices in H. $\mathcal{F}_{\mathrm{i}}$ is finite therefore there is a maximum element, let $k = \max\{|F| \mid F \in \mathcal{F}_{\mathrm{i}}\}$. Then the maximum number of vertices of a graph in $\mathcal{F}_{\mathrm{mw}}$ is $2^k + k$. □

Lemma 3.5. *Let the graph classes $\mathcal{C}, \mathcal{C}'$ be $\mathcal{F}$-free, $\mathcal{F}'$-free respectively, then $\mathcal{C} \cap \mathcal{C}'$ is $(\mathcal{F} \cup \mathcal{F}')$-free.*

- Interval graphs = chordal ∩ co-comparability
- Trivially perfect = chordal ∩ cograph
- co-Trivially perfect = co-chordal ∩ cograph
- Threshold graphs = Trivially perfect ∩ co-Trivially perfect

From these characterisations we obtain the results in Table 1. It is noteworthy that for $\mathcal{C}_1 \cap \mathcal{C}_2$, $\mathcal{F}_1 \cup \mathcal{F}_2 \subseteq \mathcal{G} \setminus (\mathcal{C}_1 \cap \mathcal{C}_2)$ but $\mathcal{F}_1 \cup \mathcal{F}_2$ may contain graphs which are not minimal with respect to the partial order.

Class of ... graphs		Minimal forbidden graphs
co-chordal	$\leqslant_{\mathrm{i}}$	$\overline{C_{n+4}}^{*}$ for $n \geq 0$
	$\leqslant_{\mathrm{mw}}$	$\overline{C_{n+4}}$ + additional vertices*
trivially perfect	$\leqslant_{\mathrm{i}}$	C_4, P_4 [5]
	$\leqslant_{\mathrm{mw}}$	$P_4, P_4 \bowtie K_1, C_4, W_4$
co-trivially perfect	$\leqslant_{\mathrm{i}}$	$2K_2, P_4$ [1, Theorem 6.6.1]
	$\leqslant_{\mathrm{mw}}$	$2K_2 \bowtie K_1, P_4, P_4 \bowtie K_1, C_5, \overline{C_6}$
threshold	$\leqslant_{\mathrm{i}}$	$2K_2, C_4, P_4$ [1, Theorem 6.6.3]
	$\leqslant_{\mathrm{mw}}$	$P_4, P_4 \bowtie K_1, 2K_2 \bowtie K_1, C_4, W_4$
interval	$\leqslant_{\mathrm{i}}$	${C_{n+4}}^{*}, \mathrm{T}_2, \mathrm{X}_{31}, {\mathrm{XF}_2}^{n+1*}, {\mathrm{XF}_3}^{n*}$ for $n \geq 0$ [4]
	$\leqslant_{\mathrm{mw}}$	$\{C_4, W_4, \mathrm{T}_2, \mathrm{X}_{31}, {\mathrm{XF}_2}^{1}, S_3, \ldots$
co-comparability	$\leqslant_{\mathrm{i}}$	$C_{n+6}, \mathrm{T}_2, \mathrm{X}_2, \mathrm{X}_3, \mathrm{X}_{30}, \mathrm{X}_{31}, \mathrm{X}_{32}, \mathrm{X}_{33}, \mathrm{X}_{34}, \mathrm{X}_{35}, \mathrm{X}_{36}, {\mathrm{XF}_2}^{n+1*}$, ${\mathrm{XF}_3}^{n*}, {\mathrm{XF}_4}^{n*}, \text{co-}{\mathrm{XF}_1}^{2n+3*}, \text{co-}{\mathrm{XF}_5}^{2n+3*}, \text{co-}{\mathrm{XF}_6}^{2n+2*}, \overline{C_{2n+1}}$ for $n \geq 0$ [3] *
	$\leqslant_{\mathrm{mw}}$	$\{\mathrm{T}_2, \overline{S_3}, S_3, S_3 - e, C_6, D_0, D_1, D_2, \ldots$

Table 1: Forbidden graphs for a collection of graph classes: * denotes an infinite set.

Conclusions

We have presented a set of equivalent definitions for a number of graph classes related to perfect graphs, these graph classes had previously been characterised by forbidding a set of induced subgraphs but the characterisation with respect to other partial order had gone unexplored. This contribution shows that with respect to the contraction minor relation the minimal forbidden set is often infinite, this result is an effect of an infinite series of false twins. We have introduced a new partial order which allows a finite forbidden set characterisation and we have established an upper bound on the size of the minimal forbidden graphs with respect to $\leqslant_{\mathrm{mw}}$ if the class has a finite minimal forbidden set with respect to induced subgraphs.

References

[1] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph classes: a survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[2] S. Foldes and P.L. Hammer. Split graphs. *Congr. Numer*, 19:311–315, 1977.

[3] T. Gallai. Transitiv orientierbare Graphen. *Acta Mathematica Hungarica*, 18:25–66, 1967.

[4] P.C. Gilmore and A.J. Hoffman. A characterization of comparability graphs and of interval graphs. Technical report, DTIC Document, 1962.

[5] M.C. Golumbic. Trivially perfect graphs. *Discrete Mathematics*, 24(1):105 – 107, 1978.

[6] A. Hajnal and J. Surányi. Über die Auflösung von Graphen in vollständige Teilgraphen. *Univ Sci. Budapest, Eötvös Sect. Math. 1*, pages 113 – 121, 1958.

The VC-Dimension of Graphs with Respect to k-Connected Subgraphs

Andrea Munaro[1]

[1]Dept. of Computer Science and Hausdorff Center for Mathematics, University of Bonn,
aerdna.munaro@gmail.com

1 Introduction

The notion now called VC-dimension of a set system was introduced by Vapnik and Chervonenkis [5]. It represents a prominent measure of the "complexity" of the set system. Let $\mathcal{H}$ be a set system on a finite set X. A subset $Y \subseteq X$ is *shattered by* $\mathcal{H}$ if $\{E \cap Y : E \in \mathcal{H}\} = 2^Y$. The *VC-dimension of* $\mathcal{H}$ is defined as the maximum size of a set shattered by $\mathcal{H}$. We are interested in studying the VC-dimension of the set system on the vertex set of some graph which is induced by a certain family of its subgraphs. The definitions above can be adapted to our special setting as follows. Let $G = (V, E)$ be a finite simple graph and let $\mathcal{P}$ be a family of subgraphs of G. A subset $A \subseteq V$ is *$\mathcal{P}$-shattered* if every subset of A can be obtained as the intersection of $V(H)$, for $H \in \mathcal{P}$, with A. The *VC-dimension of G with respect to $\mathcal{P}$*, denoted by $\mathrm{VC}_{\mathcal{P}}(G)$, is defined as the maximum size of a $\mathcal{P}$-shattered subset. In this way we obtain several different notions of VC-dimension, each one related to a special family of subgraphs. This study was first initiated in a seminal paper by Haussler and Welzl [2]. They noticed that since a graph of order n has n closed neighbourhoods, then its VC-dimension with respect to closed neighbourhoods is at most $\lfloor \log_2 n \rfloor$. Kranakis et al. [3] investigated the VC-dimensions with respect to other families of subgraphs, such as the family of trees, connected graphs, paths, cycles and stars. In particular they proved that, denoting by $\mathrm{VC}_{\mathrm{con}}$ the VC-dimension with respect to connected subgraphs, the following holds.

Theorem 1 (Kranakis et al. [3])**.** *Let $\ell(G)$ denotes the number of leaves in a maximum leaf spanning tree of G. Then $\ell(G) \leq \mathrm{VC}_{\mathrm{con}}(G) \leq \ell(G) + 1$, for any graph G.*

From another perspective, a natural question is to investigate the computational complexity of computing $\mathrm{VC}_{\mathcal{P}}(G)$ for a given graph G and a family of its subgraphs $\mathcal{P}$.

Graph $\mathrm{VC}_{\mathcal{P}}$ Dimension	
Instance:	A graph G and a number $s \geq 1$.
Question:	Does $\mathrm{VC}_{\mathcal{P}}(G) \geq s$ hold?

Kranakis et al. [3] showed that Graph $\mathrm{VC}_{\mathrm{con}}$ Dimension is NP-complete. In the following sections we continue the study initiated in [3] by proving an analogue of Theorem 1 for k-connected subgraphs and by giving hardness results for Graph $\mathrm{VC}_{k-\mathrm{con}}$ Dimension.

2 Bounds on the VC-dimension

We extend Theorem 1 by considering families of k-connected subgraphs, for $k \geq 2$. Concerning the upper bound, the idea is to construct a spanning tree with at least $\mathrm{VC}_{k-\mathrm{con}}(G) + k - 1$ leaves. We fix a shattered set A of maximum cardinality and choose an arbitrary vertex $r \in A$ as the root. Then we consider some k neighbours of r, say $u_1, \ldots, u_k$, and we try to "attach" the remaining vertices in A to the graph $(\{r, u_1, \ldots, u_k\}, \{ru_1, \ldots, ru_k\})$ via appropriate paths.

Theorem 2. $\mathrm{VC}_{k-\mathrm{con}}(G) \leq \ell(G) - k + 1$, *for any graph* G *and* $k \geq 2$.

By considering the complete graph on $k+1$ vertices, it is easy to see that the bound above is tight. As for a lower bound, we note that having a sufficiently large complete subgraph is enough to guarantee shattering by k-connected subgraphs.

Theorem 3. *Let* G *be a graph of order* n*, size* m *and maximum degree* Δ*. For* $k \geq 2$*,*

$$\mathrm{VC}_{k-\mathrm{con}}(G) \geq \ell(G) - k + 1 - \left(n + 2 - \left\lceil \frac{n-2}{\Delta - 1} \right\rceil - \frac{n^2}{n^2 - 2m} \right).$$

3 The decision problem

We prove NP-completeness of Graph $\mathrm{VC}_{k-\mathrm{con}}$ Dimension by a reduction from the decision version of Set Multicover, a generalization of the well-known NP-complete problem Set Cover.

Theorem 4. Graph $\mathrm{VC}_{k-\mathrm{con}}$ Dimension *is* NP*-complete even for split strongly chordal graphs.*

Sketch of Proof. Membership in NP follows from the fact that if G and G' are two k-connected graphs such that $|V(G) \cap V(G')| \geq k$, then $G \cup G'$ is k-connected as well.

As for NP-hardness, recall that an instance of Set Multicover consists of a set $S = \{a_1, \ldots, a_n\}$, a collection of subsets $S_1, \ldots, S_m \subseteq S$ and integers k and t. The question is if there exists an index set $I \subseteq \{1, \ldots, m\}$ such that $\bigcup_{i \in I} S_i = S$, each a_i is covered by at least k distinct subsets and $|I| \leq t$.

Our reduction constructs a graph $G = (V, E)$ as follows. The set of vertices V is formed by four pairwise disjoint sets A, B, C and D. A is an independent set of $n \cdot (t + k + 1)$ vertices arranged in n columns of $t+k+1$ vertices each (for $1 \leq j \leq n$, each element in the j-th column corresponds to a copy of a_j). $B = \{v_1, \ldots, v_m\}$ is a clique, where v_i corresponds to the set S_i. C is a clique of size k. D is an independent set of $t + m + 1$ vertices. Each vertex in C is connected to all vertices in B (therefore, $B \cup C$ is a clique of size $m + k$) and D. Finally, $v_i \in B$ is connected to every copy of $a_j \in A$ if and only if $a_j \in S_i$.

Since $B \cup C$ is a clique and $A \cup D$ is an independent set, then G is split. Moreover, it is clearly chordal. Now consider a cycle $(w_1, w_2, \ldots, w_\ell)$ of even length $\ell \geq 6$. There exist two consecutive vertices w_i and w_{i+1} which belong to $B \cup C$. If $w_{i-2} \in B \cup C$, then $w_{i-2}w_{i+1}$ is an odd chord. Otherwise $w_{i-3} \in B \cup C$ and $w_{i-3}w_i$ is an odd chord. Therefore, G is strongly chordal.

Finally it is not difficult to prove that there is an index set $I \subseteq \{1, \ldots, m\}$ such that $\bigcup_{i \in I} S_i = S$, each a_i is covered by at least k distinct subsets and $|I| \leq t$, if and only if $\mathrm{VC}_{k-\mathrm{con}}(G) \geq |V| - (t + k)$. □

Now we consider the problem GRAPH VC_{2-con} DIMENSION and prove that it remains NP-complete for planar graphs. A natural way to prove NP-hardness of a planar problem is to reduce from another planar problem. Our proof is indeed based on a variant of PLANAR MONOTONE 3-SAT, which was shown to be NP-complete by de Berg and Khosravi [1]. Since their problem has already found many and diverse applications in NP-hardness proofs, we think that our variant may be useful as well.

Let $\mathcal{U} = \{x_1, \ldots, x_n\}$ be a set of n boolean variables and let $\mathcal{C} = C_1 \wedge \cdots \wedge C_m$ be a 3-CNF formula defined over $\mathcal{U}$, where each clause C_i is the disjunction of exactly three literals. 1-IN-3-SAT is the problem of deciding whether $\mathcal{C}$ is satisfiable in 1-in-3 sense. An instance $\mathcal{C}$ of 1-IN-3-SAT is *planar* if the associated variable-clause graph $G(\mathcal{C})$ is planar. A planar instance of 1-IN-3-SAT has a *rectilinear representation* if $G(\mathcal{C})$ can be drawn as follows: the vertices are drawn as rectangles, with all the rectangles representing variables on a horizontal line, and the edges are vertical segments. A clause with only positive literals is a *positive clause*, a clause with only negative literals is a *negative clause* and a clause with both positive and negative literals is a *mixed clause*. An instance of 1-IN-3-SAT is called *monotone* if it does not contain any mixed clause. Given a planar monotone instance $\mathcal{C}$ of 1-IN-3-SAT, a *monotone rectilinear representation* of $\mathcal{C}$ is a rectilinear representation where all positive clauses are drawn above the variables and all negative clauses are drawn below the variables. We can define the following decision problem.

PLANAR MONOTONE 1-IN-3-SAT	
Instance:	A planar monotone 3-CNF formula $\mathcal{C}$ defined over $\mathcal{U}$, together with a monotone rectilinear representation.
Question:	Is $\mathcal{C}$ satisfiable (in 1-in-3 sense)?

Theorem 5. PLANAR MONOTONE 1-IN-3-SAT *is* NP*-complete.*

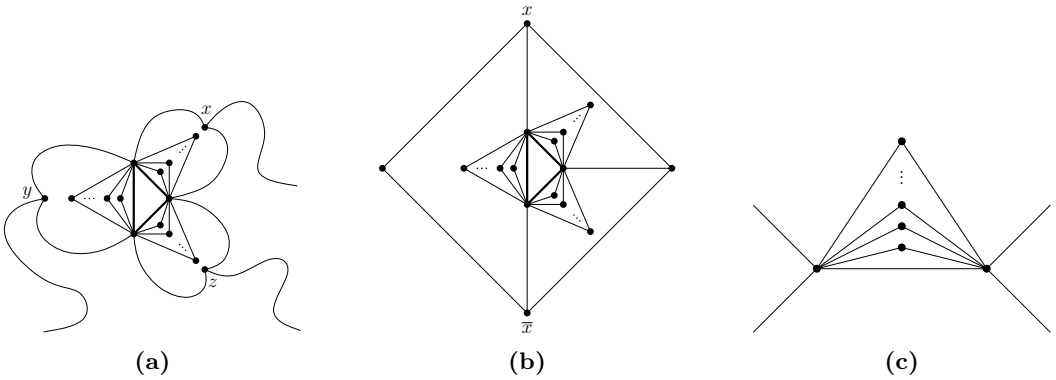
Our proof of Theorem 5 is based on a reduction from a variant of PLANAR 1-IN-3-SAT shown to be NP-complete by Mulzer and Rote [4].

Finally we show that GRAPH VC_{2-con} DIMENSION remains NP-complete for planar graphs by a reduction from PLANAR MONOTONE 1-IN-3-SAT. For an instance $\mathcal{C}$ of PLANAR MONOTONE 1-IN-3-SAT, our reduction constructs a planar graph $G_{\mathcal{C}}$ by adding several planar gadgets to the variable-clause graph $G(\mathcal{C})$.

Theorem 6. GRAPH VC_{2-con} DIMENSION *is* NP*-complete for planar graphs.*

Sketch of Proof. We use a reduction from PLANAR MONOTONE 1-IN-3-SAT. Let $\mathcal{C}$ be an instance of PLANAR MONOTONE 1-IN-3-SAT. We modify the associated graph $G(\mathcal{C})$ to get a planar graph $G_{\mathcal{C}}$ as follows. First we define a *shamrock* as the graph constructed successively from a triangle C, called the *centre* of the shamrock, by adding q ears of length 2 to every pair of vertices of C. These q ears constitute a *leaf* of the shamrock. For a fixed planar drawing of a shamrock, the exterior vertex in a leaf is called the *peak* of the leaf, while the other vertices are called the *veins* of the leaf. For every clause, we introduce a shamrock with p veins in every leaf (for a p to be chosen later) as depicted in Figure 1(a). Each peak of a leaf in the shamrock corresponds to a literal in the clause. For every variable x, we introduce a variable gadget as depicted in Figure 1(b). It consists of two parts. The first part is a 4-cycle with two specified opposite vertices corresponding to the literals x and $\overline{x}$, called the *literal* vertices, and with two *horizontal* vertices. The second part is a shamrock with p veins in every leaf and no peaks

which is connected to the 4-cycle via three edges with endpoints in the centre of the shamrock. We require that two of these edges join x and $\overline{x}$ to the centre of the shamrock. We connect every pair of consecutive variables in the monotone rectilinear representation of $\mathcal{C}$ through a variable connector gadget as depicted in Figure 1(c). It consists of $p+1$ independent paths, one of which is of length 1, while the remainings are all of length 2.



(a) **(b)** **(c)**

Figure 1: Gadgets for the reduction of Planar Monotone 1-In-3-Sat to Graph $\mathrm{VC}_{2-\mathrm{con}}$ Dimension. (a) Clause gadget corresponding to the clause $(x \vee y \vee z)$. (b) Variable gadget. (c) Variable connector gadget.

Finally, we connect clause and variable gadgets via edges joining every peak in a shamrock with the corresponding literal in the variable gadget. We do this according to the monotone rectilinear representation of $G(\mathcal{C})$ and we define $p := 5m + 6n + 1$. Clearly, the above construction can be done in polynomial time and the resulting graph $G_{\mathcal{C}} = (V, E)$ is planar. We can prove that $\mathrm{VC}_{2-\mathrm{con}}(G_{\mathcal{C}}) \geq |V| - (5m + 6n)$ if and only if $\mathcal{C}$ is satisfiable (in 1-in-3 sense). □

On the positive side, we have the following result.

Theorem 7. Graph $\mathrm{VC}_{k-\mathrm{con}}$ Dimension *is decidable in linear time for threshold graphs.*

References

[1] M. de Berg and A. Khosravi. Optimal Binary Space Partitions for Segments in the Plane. *Int. J. Comput. Geometry Appl.*, 22(3):187–206, 2012.

[2] D. Haussler and E. Welzl. ε-Nets and Simplex Range Queries. *Discrete & Computational Geometry*, 2:127–151, 1987.

[3] E. Kranakis, D. Krizanc, B. Ruf, J. Urrutia, and G. J. Woeginger. The VC-dimension of set systems defined by graphs. *Discrete Applied Mathematics*, 77(3):237–257, 1997.

[4] W. Mulzer and G. Rote. Minimum weight triangulation is NP-hard. In *Proceedings of the 22nd ACM Symposium on Computational Geometry, Sedona, Arizona, USA, June 5-7, 2006*, pages 1–10. ACM, 2006.

[5] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16(2):264–280, 1971.

Coloring of signed graphs

S. Pirzada[1], Muhammad Ali Khan[2], and E. Sampathkumar[3]

[1]Department of Mathematics, University of Kashmir, Srinagar, India
[2]King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
[3]Department of Studies in Mathematics, University of Mysore, India

A signed graph G is formed by assigning either a '+' or '−' sign to every edge of a graph $G_u = (V, E)$. The negative degree $d^-(v)$ of a vertex v in a signed graph is the number of negative edges incident at v. We say that an edge uv in a signed graph is a c-edge if both $d^-(u)$ and $d^-(v)$ are even or both are odd. A c-complete signed graph is a simple signed graph in which every two vertices form a c-edge. Over the years, different notions of vertex colorings have been defined for signed graphs. Here we define a new type of coloring that distinguishes between the end vertices of c-edges. By an s-coloring of a signed graph we mean a coloring of its vertices such that any two vertices joined by a c-edge receive different colors. The s-chromatic number $\chi_s(G)$ of a signed graph is the minimum number of colors required to s-color the vertices of G. In this paper, several bounds are obtained for $\chi_s(G)$ that are similar to those for the chromatic number of a graph. We show that the number of s-colorings of a signed graph G is a polynomial function of the number k of colors, which we call the s-chromatic polynomial $S(G, k)$ of G. We develop a deletion-contraction type recursive procedure to determine $S(G, k)$ for any signed graph. The deletion and contraction operations we use are modifications of the traditional operations as the traditional operations do not work for counting s-colorings. We introduce the notions of c-complete and c-full signed graphs, characterizing different classes of c-full signed graphs and determining the number of c-complete signed graphs on a given number of vertices. Furthermore, the relationship between s-coloring and the existing signed graph colorings is also investigated.

1 Introduction

A signed graph is a triple $G = (V, E, \sigma)$, where $G_u = (V, E)$ is a graph and $\sigma : E \to \{+, -\}$ is a sign function. The graph G_u is called the *underlying graph* of G. An edge e is said to be positive if $\sigma(e) = +$ and negative otherwise. For a vertex v in a signed graph G, let $d^+(v)$ and $d^-(v)$ respectively denote the number of positive and negative edges incident at v. There exist two types of coloring of signed graphs in the literature. Harary and Cartwright [1] define a coloring of signed graph in which no two end vertices of a negative edge are colored the same, whereas the end vertices of a positive edge are necessarily colored the same. This process of coloring is also referred to as signed graph *partitioning* or *clustering* and is widely used in network analysis. The motivation is to partition the vertices of a network in such a way that any two actors (vertices) that have a negative relationship (edge) are partitioned

into different color classes while the actors having a positive relationship are grouped together in the same color class. However, not all signed graphs can be colored using this type of coloring (for instance, consider the complete graph K_3 whose edges are alternately assigned $+$ and $-$ signs) . In the sequel we refer to this coloring as the *p-coloring* and say that a signed graph is *partitionable* or p-colorable if it can be colored in this way. The corresponding chromatic number, if it exists, will be called the *p-chromatic number* and denoted by $\chi_p(G)$. The other existing notion of coloring is due to Zaslavsky [2,3], who defines a coloring of a signed graph $G = (V, E, \sigma)$ as a function $f : V \to \{0, \pm 1, \ldots, \pm n\}$ such that if uv is an edge, $f(u) \neq f(v)$ and if uv is a negative edge with $f(u) = i$, then $f(v) \neq -i$. Zaslavsky obtained a chromatic polynomial for his coloring and showed that the traditional deletion-contraction recursion can be used to determine the chromatic polynomial of a signed graph. We refer to this coloring, the corresponding chromatic number and the related chromatic polynomial as *z-coloring*, *z-chromatic number* denoted by $\chi_z(G)$ and *z-chromatic polynomial* denoted by $P_z(G, k)$ respectively.

In this paper, we define a new type of vertex coloring for signed graphs. Given a signed graph G, two vertices u and v of G are said to be *c-adjacent* if uv is an edge and $d^-(u)$ and $d^-(v)$ are either both even or both odd. In this case, we call the edge uv a *c-edge*. For a vertex v, let $Sgn(v)$ denote the product of the signs of the edges incident at v. Clearly, two adjacent vertices u and v are c-adjacent if and only if $Sgn(u) = Sgn(v)$. We define an *s-coloring* of a signed graph G as a coloring of its vertices such that no two c-adjacent vertices receive the same color. The *s-chromatic number*, $\chi_s(G)$, is the minimum number of colors required for s-coloring G.

We begin by introducing some new terminology. A path P in a signed graph $G = (V, E, \sigma)$ is a *c-path* if all edges in P are c-edges. Clearly, a path is a c-path if and only if $d^-(v)$ is either even or odd for all vertices v in P. A signed graph G is *c-connected* if any two vertices in G are connected by a c-path. A set $S \subset V$ is *c-independent* if no two vertices in S are c-adjacent. The *c-independence number* $\beta_c(G)$ of G is the maximum cardinality of a c-independent set. Thus $\chi_s(G)$ is the minimum order of a partition of V into c-independent sets. Given a signed graph $G = (V, E, \sigma)$, the c-graph of G is an unsigned graph given by $G_c = (V, E_c)$, where E_c is the set of c-edges of G. Clearly, for any signed graph G, we have $\beta_c(G) = \beta(G_c)$, where $\beta(G_c)$ is the independence number of the c-graph G_c of G. We say that a signed graph G is *complete* is any two vertices in G are adjacent and *c-complete* if any two vertices in G are c-adjacent. Thus G is a c-complete signed graph of order n if and only if $\chi_s(G) = n$. A signed graph G is *c-full* if every edge in G is a c-edge. For simple signed graphs the notions of c-completeness and c-fullness coincide.

2 Results

Proposition 1. *For a signed graph G of order n, the following statements are equivalent. (i) G is c-complete, (ii) G is complete and c-full, (iii) $\chi_s(G) = n$, (iv) $\beta_c(G) = 1$.*

The following result characterizes c-full signed paths.

Theorem 2. *Let P_n denote the signed path of order n. (i) If n is even, then P_n is c-full if and only if all the edges in P_n are positive or the edges of P_n are alternately negative and positive, starting and ending with negative edges, (ii) If n is odd, then P_n is c-full if and only if all the edges in P_n are positive.*

From the above result, we have the following observations which determines the number of c-full signed paths.

Corollary 3. *(i) Among all signed paths of order $n \geq 2$ and n even, there are exactly two signed paths which are c-full. (ii) Among all signed paths of order $n \geq 2$ and n odd, there is exactly one signed path which is c-full.*

The following result characterizes c-full signed cycles.

Theorem 4. *Let C_n be a signed cycle of order n. (i) If n is even, then C_n is c-full if and only either all the edges in C_n are either positive or all edges in C_n are negative or (c) the edges in C_n are alternatively positive and negative, (ii) If n is odd, then C_n is c-full if and only if (a) all the edges in C_n are positive or (b) all the edges are in C_n are negative.*

In a signed graph G, if $d^-(v)$ is even for all v in G, then no two adjacent vertices receive the same color in an s-coloring of G. This is also true $d^-(v)$ is odd for all v in G. This implies the following.

Theorem 5. *For a connected signed graph G, the following statements are equivalent. (i) G is c-full, (ii) G is c-connected, (iii) The negative degree $d^-(v)$ is even (or odd) for all $v \in G$, (iv) For each edge uv in G, $Sgn(u) = Sgn(v)$, (v) $\chi_s(G) = \chi(G_u)$, where $\chi(G_u)$ is the chromatic number of the underlying graph G_u of G.*

We now determine the number of c-complete signed graphs on a given number of vertices. There are exactly two c-complete signed graphs on three vertices, namely the one in which all the edges are signed positive and its s-complement. There are six c-complete signed graphs on four vertices, the two c-complete signed graphs K_4 with all edges having the same positive sign and all edges having the same negative sign.

Theorem 6. *There are exactly $2n + 2$ c-complete signed graphs on $2n$ vertices, while the number of c-complete signed graphs on $2n + 1$ vertices is $n + 1$.*

In a signed graph G, the degree of a vertex v is $deg(v) = d^+(v) + d^-(v)$. For some integer $r \geq 0$. G is r-regular if $deg(v) = r$ for all vertices v in G. Further, G is r^+-regular (r^--regular) if $d^+(v) = r$ ($d^-(v) = r$), for all vertices v in G. The *s-complement* G^s of a signed graph G is obtained by interchanging the positive signs and negative signs on the edges of G.

Proposition 7. *Let G be a connected signed graph which is regular of degree r. Then (i) G is c-connected if and only if G^s is c-connected, (ii) G is c-complete if and only if G^s is c-complete, (iii) if G is r^--regular, then G is c-connected. The converse is not true.*

We note that a c-complete signed graph need not be r^+-regular or r^--regular for any $r \geq 0$. Also the s-complement G^s need not be c-connected. For example, the graph G in Figure 3, is r^--regular, where $r = 1$ and c-connected, but its s-complement is not c-connected.

Proposition 8. *Let G be a c-connected signed graph. Then its s-complement G^s is c-connected if and only if $d^+(v)$ is even or odd for all vertices v in G.*

Let P be a coloring of the vertices of a graph G. A set $S \subset V$ is a color class (with respect to P) if all the vertices of S receive the same color in some P coloring of G. For example, in the usual coloring of a graph any independent set of vertices is a color class. Motivated by this

definition, we define a color class in a s-coloring as follows. A set of vertices in a signed graph is an $s-color\ class$ if all the vertices in S receive the same color in some s-coloring of G. Note that an s-color class is a c-independent set. Thus the s-chromatic number $\chi_s(G)$ is the minimum order of a partition of the vertex set of G into s-color classes or c-independent sets. Also note that an independent set is c-independent. The c-independence number $\beta_{0c}(G)$ of a signed graph is the maximum cardinality of a c-independent set. Clearly, since any independent set is c-independent, we have for any signed graph $\beta_0(G) \leq \beta_{0c}$, where β_0 is the independence number of the underlying graph G_u of G. As in graphs we have the following bounds for $\chi_s(G)$.

Proposition 9. *For any signed graph of order n, $n\beta_{0c} \leq \chi_s(G) \leq n - \beta_{0c}(G) + 1$.*

A set S of vertices in a signed graph G is a c-clique if the subgraph $< S >$ induced by S is c-complete. The $c-graph\ G_c$ of a signed graph G is a graph having the same vertex set as G and two vertices u and v are adjacent in G_c if and only if uv is a c-edge in G. Clearly the degree of a vertex v in G_c is equal to its c-degree in G and $\chi_s(G) = \chi(G_c)$. Also $\Delta_c(G) = \Delta(G_c)$, $\delta_c(G) = \delta(G_c)$ and $\beta_{s0}(G) = \beta_0(G_c)$. It is well known that the determination of chromatic number of a graph is NP-hard. In view of the above remarks and the fact $\chi_s(G) = \chi(G_c)$, we deduce many bounds for $\chi_s(G)$ from the well known bounds of the chromatic number $\chi(G_c)$ of the graph G_c.

Proposition 10. *For a signed graph G, (i) $\chi_s(G) \leq 1 + max\ \delta_c(G')$, where the maximum is taken over all induced sub-signed graphs G' of G, (ii) $\chi_s(G) \leq 1 + \Delta_s(G)$, (iii) $\chi_s(G) \leq \chi(G_u)$ and equality holds if and only if G is c-full, (iv) The chromatic number of a signed tree is less or equal to two.*

Let $G = (V, E, \sigma)$ be a signed graph, where $G_u = (V, E)$ is a graph with possible multiple edges but no loops, such that for any multiple edges e and f, $\sigma(e) = \sigma(f)$. The aim of this section is to develop a chromatic polynomial for s-coloring of signed graphs on the same lines as the chromatic polynomial of ordinary unsigned graphs. Let $S(G, k)$ denote the number of distinct s-colorings of G as a function of the number of colors k. Let $P(H, k)$ denote the classical chromatic polynomial of a graph G. First we observe that the number of s-colorings of $G = (V, E, \sigma)$ using k colors is the same as the number of vertex colorings of the c-graph G_c of G. Thus $S(G, k) = P(G_c, k)$ and so $S(G, k)$ is a polynomial in k. Furthermore, the following result shows that several properties of chromatic polynomial are also satisfied by the s-chromatic polynomial.

Theorem 11. *Given a signed graph $G = (V, E, \sigma)$, the following holds for the polynomial $S(G, k)$. (i) The degree of $S(G, k)$ is $|V|$, (ii) The leading coefficient of $S(G, k)$ is 1, (iii) The coefficient of $k^{|V|-1}$ in $S(G, k)$ is equal to $-|E_c|$, (iv) The constant term of $S(G, k)$ equals 0, (v) The coefficient of k in $S(G, k)$ is nonzero if and only if G is c-connected, (vi) If the lowest nonzero coefficient of $S(G, k)$ is of k^p then the number of c-connected components of G is equal to p.*

Theorem 12.

1. *Given a signed graph $G = (V, E, \sigma)$, let $G^- = (V, E^c, \sigma^-)$ be the signed graph obtained by deleting all the non c-edges from G and assigning negative sign to all c-edges of G, then $\chi_s(G) = \chi_p(G^-)$.*
2. *Let $G = (V, E, \sigma)$ be a signed graph and let $G^+ = (V, E^c, \sigma^+)$ be the signed graph obtained by deleting all non c-edges from G and assigning positive sign to all c-edges of G. Then $\chi_s(G) = 2\chi_z(G^+) + 1$.*

References

[1] D. Cartwright, F. Harary, *On the coloring of signed graphs*, Elem. Math., **23** (1968) 85-89.

[2] T. Zaslavsky, *Signed graph coloring*, Disc. Math., **39** (1982) 215-228.

[3] T. Zaslavsky, *How colorfull is the signed graph*, Disc. Math., *4* (1970) 322-325.

Approximation results for the linear ordering problem on interval graphs

Alain Quilliot[1] and Djamal Rebaine[2]

[1]Université Blaise Pascal, LIMOS, UMR CNRS 6158, BP 10125 Campus des Cézeaux, 63173 Aubière (France), e.mail: alain.quilliot@isima.fr
[2](corresponding author) Université du Québec à Chicoutimi, Département d'Informatique et Mathématique, Saguenay (Canada), e.mail: djamal_rebaine@uqac.ca

We discuss in this study results on the linear ordering problem. We first present a new lower bound, and show that it can be achieved by a simple linear algorithm on unit interval graphs. Then we propose a heuristic algorithm on interval graphs and undertake its worst case analysis.
Keywords: Approximation algorithm, interval graphs, linear ordering, worst-case analysis.

1 Introduction

Let $G = (X, E)$ be a non-oriented graph where X and E denote the set of vertices and edges, respectively. The linear ordering problem for G consists of finding a one to one mapping σ from X to $\{1, \ldots, |X|\}$ such that $\sum_{(x,y)\in E} |\sigma(y) - \sigma(x)|$ is minimized.

The corresponding decision linear ordering problem was first shown to be $\mathcal{NP}$-complete for arbitrary graphs [4]. It was also shown later to remain $\mathcal{NP}$-complete even for some restricted classes of graphs such as interval graphs [2], and bipartite graphs [4]. Polynomial time algorithms were also developed for other restricted graph classes such as trees [1], and unit interval graphs [3].

In what follows, we propose through a reformulation of the problem a new lower bound. Next, we focus on interval graphs. We first solve in a very simple way the case of unit interval graphs, then we present approximation results for a polynomial time heuristic algorithm.

2 Preliminaries

2.1 Definitions and notation

We present in this section definitions and notation on graphs, sets, and linear ordering problems.

Definition 1. *Let $G = (X, E)$ be a simple graph. If $A \subset X$, then G_A is the subgraph induced by A from G. If $x \in X$, then $\Gamma_G(x) = \{y \in X \mid (x, y) \in E\}$. A triangle is a clique with three nodes, and an anti-edge is a pair (x, y), $x \neq y$, such that $(x, y) \notin E$. A Fork with root x is a triple $\{x, y, z\}$ such that (x, y) and (x, z) are in E and (y, z) is an anti-edge of G, and*

an Anti-Fork with root z is a triple $\{x, y, z\}$ such that (x, y) is in E, and (x, z) and (y, z) are anti-edges of G.

Definition 2. *A graph $G = (X, E)$ is an interval graph if it can be viewed as the intersection graph of a set of closed intervals of the real line. If G is such a graph, we may assume that X is a closed interval family, and, for any $x \in X$, denote by $o(x)$ and $d(x)$ the endpoints of interval $x = [o(x), d(x)]$. It is always possible to do in such a way that all values $o(x)$, $d(x)$, $x \in X$, are disjoint. Then, we set i) $x \subset y$ if $o(x) < o(y)$ and $d(y) < d(x)$, ii) $x << y$ if $d(x) < o(x)$, and x Ov y if $o(x) < o(y) < d(x) < d(y)$.*

G is a unit interval graph if the related closed interval subset X may be chosen in a way that no couple (x, y) of E exists such that $x \subset y$.

A fork $\{x, y, z\}$ is a strong fork if at least one interval t among $\{y, z\}$ is such that $t \subset x$.

Definition 3. *A linear ordering of a set X is a binary order relation σ (non reflexive and transitive) such that, for any (x, y) in X, $x \neq y$, either $(x\ \sigma\ y)$ or $(y\ \sigma\ x)$. In case X is an interval family with distinct endpoints, we denote by σ-can the canonical linear ordering defined as follows: x σ-can y if, and only if, $o(x) < o(y)$. Finally, a linear ordering σ is right if it is compatible with Ov and $<<$ orderings.*

2.2 Reformulation

We introduce here another way of formulating the linear ordering problem. Doing so makes it possible to use some counting arguments in the derivation of lower bounds and approximation results.

Given are a graph $G = (X, E)$ and a linear ordering σ of X. For any edge $e = (x, y)$ of E we set $BE(e, z, \sigma) = 1$ if $(x\ \sigma\ z\ \sigma\ y)$ or $(y\ \sigma\ z\ \sigma\ x)$, and $BE(e, z, \sigma) = 0$ otherwise. Value $BE(e, z, \sigma)$ is called the elementary break value of e by node z according to σ. The resolution of the Linear Ordering Problem is then equivalent to the problem of computing a linear ordering σ that minimizes $BG(G, \sigma) = \sum_{e=(x,y)\in E} \sum_{z\in X} BE(e, z, \sigma)$, the global break of graph G according to σ. We denote by $OPB(G) = Inf_{\sigma} B(e, \sigma)$ the optimal global break value.

3 A general lower bound

The following lower bound is derived from the idea that computing a good linear linear ordering σ of the vertices of a given graph $G(X, E)$ is equivalent to the problem of deciding for every vertex $x \in X$ which vertices of $\Gamma_g(x)$ are located before and after x according to σ. To handle this problem, we need to solve the following Bi-Partition problem:

Let $H = (Z, F)$ be a simple graph. For any partition $Z = A \cup^E B$ we denote by $V_H(A, B)$ the number of anti-edges of H included either in A or B. Solving the Bi-Partition problem corresponds to the computation of $N(H) = Inf_{(A,B),Z=A\cup^E B} V_H(A, B)$.

Now, if $Tr(G)$ denotes the number of triangles of $G = (X, E)$, and, for $x \in X$, $N^*_G(x)$ represents $N(\Gamma(x))$, then the following result holds.

Theorem 4. *For any graph $G = (X, E)$, we have that $OPB(G) \geq Tr(G) + \sum_x N^*_G(x)$.*

4 The case of interval graphs

In the case of unit interval graphs, we derive from Theorem 4 in a straightforward way the following result.

Theorem 5. *If $G = (X, E)$ is a unit interval graph then σ-can is an optimal solution for the linear ordering problem.*

For general interval graphs, it is easy to check that an optimal linear ordering may not be right. Still, experiments show that best linear orderings are most often right.

Lemma 6. *If the linear ordering σ is right, then $BG(G, \sigma) = Tr(G) + SFk(G, \sigma)$, where $SFk(G, \sigma)$ is the number of Strong Forks (x, y, z) such that ($x \ \sigma \ y$ and $x \ \sigma \ z$) or ($y \ \sigma \ x$ and $z \ \sigma \ x$).*

Lemma 6 makes it possible to design algorithms that construct a linear ordering σ by locally distributing the vertices of $\Gamma_G(x)$ before and after x according to σ, that is by performing a local resolution of the Bi-Partition problem. Ensuring compatibility of this process means extending Bi-Partition as follows. Given are a graph $H = (Z, F)$ and a partial ordering σ of Z. A subset $Y \subset Z$ is a

- Left-Section if for any x and y such that $x \in Y$, and $y \ \sigma \ x$, we also have $y \in Y$.
- Right-Section if for any x and y such that $y \in Y$, and $y \ \sigma \ x$, we also have $x \in Y$.

Given two disjoint subsets A and B of Z, such that A is a Left-Section and B is a Right-Section, the Extended Bi-Partition problem $BIPEXT(H, A, B, \sigma)$ aims to compute a partition $Z = A^* \cup^E B^*$ such that

- A^* contains A and is a Left-Section.
- B^* contains B and is a Right-Section.
- $V_H(A^*, B^*)$ is at its minimum.
- A^* is maximal for the inclusion order, with the above conditions being satisfied.

Let us note that if σ is linear, then this problem is clearly polynomially solvable.

In any case, from the above considerations, we may derive the following algorithm, which builds σ by iteratively distributing in an ad-hoc way the vertices of $\Gamma_G(x)$ before and after x according to σ, while processing vertices x according to the partial ordering $\subset$. In the following, we define $\mathrm{Dom}(x) = \{y \mid x \subset y\}$. Note that the result σ-bal of this algorithm is provided by σ.

Balancing Algorithm - For any $x \in X$, $\mathrm{Prof}(x)$ is the maximal length of chain $x = x_1 \subset \cdots \subset x_k$.

- Initialize the partial ordering σ by setting that $x \ \sigma \ y$ if $x \ Ov \ y$ or $x << y$;
- For $i = 1$ to (Prof-Max = $\max_{x \in X}$ Prof(x) - 1) do {
  - For $x \in X$ such that $\mathrm{Prof}(x) = i$ do {
    $Z = \{y \in X \mid (x, y) \in E, x \neq Dom(y)\}$;
    $A = \{y \in X \mid y \ \sigma \ x\}$; $B = \{y \in X \mid x \ \sigma \ y\}$;
    Solve BIPEX(Z, A, B, σ) and derive $A^* = A^*(x)$, $B^* = B^*(x)$; **(I1)**
  } // end of the inner For

- For any pair $(y, z) \in X$, not comparable for σ, set $y\ \sigma\ z$ if i), ii) or iii) below is satisfied:
  i. There exists x such that Prof(x)= i, $y \in A^*(x)$, $z \in B^*(x)$.
  ii. Prof(y)= i, $z \in B^*(y)$.
  iii. Prof(z)= i, $y \in A^*(z)$.

} // end of For

Theorem 7. *The Balancing Algorithm generates a relation σ-bal which is a right linear ordering with $BG(G, \sigma\text{-bal})$ not bigger than $BG(G, \sigma\text{-can})$.*

Processing the above (I1) generates a complication as there is no known efficient algorithm for it. Still, it is possible to solve it in a heuristic way through the polynomial descent process LS-BIPEXT. This process improves iteratively the $A^* \cup^E B^*$ by switching a well computed Right-Section C of A^* into B^* or a well computed Left-Section D of B^* into A^*. We call LS-Balancing the resulting version of the Balancing Algorithm.

Theorem 8. *The LS-Balancing Algorithm generates, in polynomial time, a right linear ordering σ-LS-bal such that $OPB(G) \leq BG(G, \sigma\text{-bal}) \leq BG(G, \sigma\text{-LS-bal}) \leq BG(G, \sigma\text{-can})$.*

If Prof-Max = $\max_{x \in X} \text{Prof}(x) - 1 = 2$, we set, for $x \in X$, $Strong(x)$ to be the number of Strong Forks with root x, $k(x)$ the maximal cardinality of a clique of $\Gamma_G(x)$, and $n(x)$ the cardinality of $\Gamma_G(x)|$.

Theorem 9. *Let $G = (X, E)$ be a graph such that Prof-Max = 2. If, for any $x \in X$, Prof(x) = 1, the optimal value of the related Bi-Partition and BIPEXT instances are the same, then $OPB(G) = BG(G, \sigma\text{-bal}) = BG(G, \sigma\text{-LS-bal})$. In any case, we have*

- $BG(G, \sigma\text{-bal}) - Tr(G) \leq Strong/2$.
- $BG(G, \sigma\text{-bal}) - OPB(G)$
  $\leq \sum_{x | \text{Prof}(x)=1} \left(Strong(x) - ((n(x) - 2k(x))^2 + (n(x) + 2k(x))/2) \right) /2$.

References

[1] Chung FRK. (1984): On optimal linear arrangements of trees, *Comp.& Maths. with appl.* vol. 10, pp. 43-60.

[2] Cohen J., Fomin F., Heggernes P., Kratsch D., Kucherov G. (2006): Optimal Linear Arrangement of Interval Graphs, *proceedings MFCS'06*, pp. 267-279, Springer-Verlag Berlin, Heidelberg.

[3] Corneil DG., Kim H., Natarajan S., Olariu S., Sprague AP. (1995): A simple linear time algorithm of unit interval graphs, *Inf. Proc. Letters*, vol. 55, pp. 99-104.

[4] Garey MR., Johnson DS (1979): *Computers and intractability: A guide to the theory of NP-Completeness*, Computer Press.

Hazmat transportation problem: instance size reduction through centrality erosion

Fabio Roda[1]

[1]LIX, École Polytechnique, F-91128 Palaiseau, France

1 Introduction

The Hazardous Material (Hazmat) Transportation Problem concerns the transportation of dangerous material from one or many production points to one or many garbage dumps, crossing different areas and it deals with the search of optimal cost and risk. This problem is well addressed in literature. We have considered, in a previous work [4], a particular type of constraint, namely *equity*. We have studied how a transportation system can ensure safe disposal of hazardous waste in such a way that the risk of potential catastrophic accidents is equitable over the population of the interested area. The search of equity is a goal that is not easy to formalize. The main contribution of our work was a formal formulation of two definitions of equity and their integration in a mathematical programming (MP) model. The first is a form of straightforward egalitarianism, as it asks for similar values of risk for all the interested areas (risk sharing). The second one, more sophisticated, admits an unequal distribution of available resources if this can help the most disadvantaged member of a group (Rawls's principle). We perform a series of tests to establish if our models can be utilized with realistic instances of our problem. We observe CPU time in function of the instance size. In particular, we use the AMPL modelling environment and the CPLEX 12.2 solver running with its default configuration on a single 2.4 GHz Intel Xeon CPU with 8GB RAM. Transportation systems, and in particular, road networks can be represented by means of graphs. We generate instances randomly, considering five parameters: the number of the vertices of the network (cardinality of vertex set), the probability that an arc exist (graph density), the maximum capacity on arcs, the number of commodities and the number of zones. Tables 1 and 2 show the result with increasing cardinality (till to medium size instances). Rows have to be considered by couples: one row shows the outcome considering equity constraints and the following one without equity constraints.

Tests show that our approach is practicable. Nevertheless, when the instance graphs sizes and densities grow, the solution of our MP formulation gets harder or even impossible. Thus, in this work, we focus on a method to reduce the size of the graphs preserving their fundamental features (in this context), in order to look for approximated solutions.

2 Graph reduction through centrality erosion

The basic idea is to establish a ranking among vertices. Once vertices are classified, a smaller graph can be obtained by removing the least relevant vertices.

Test	Card.	Gr.d.	Cap.	Comm.	Zones	Equity	cpu time	Total damage
1	10	0.5	10	3	4	yes	0.025996	866.045
	10	0.5	10	3	4	no	0.027995	677.997
2	20	0.5	10	3	4	yes	0.314952	383.153
	20	0.5	10	3	4	no	0.317951	313.265
3	30	0.5	10	3	4	yes	3.34749	400.658
	30	0.5	10	3	4	no	3.35249	245.669
4	40	0.5	10	3	4	yes	16.6485	289.919
	40	0.5	10	3	4	no	16.6565	246.897
5	50	0.5	10	3	4	yes	57.9702	1098.87
	50	0.5	10	3	4	no	57.9992	1062.44

Table 1: CPU time and total damage for equity as risk sharing, with increasing cardinality.

Test	Card.	Gr.d.	Cap.	Comm.	Zones	Equity	cpu time	Total damage
1	10	0.5	10	3	4	yes	0.004999	728.432
	10	0.5	10	3	4	no	0.005999	677.997
2	20	0.5	10	3	4	yes	0.004999	313.265
	20	0.5	10	3	4	no	0.005999	313.265
3	30	0.5	10	3	4	yes	0.004999	327.509
	30	0.5	10	3	4	no	0.008998	245.669
4	40	0.5	10	3	4	yes	0.011998	250.84
	40	0.5	10	3	4	no	0.016997	246.897
5	50	0.5	10	3	4	yes	0.033994	1381.43
	50	0.5	10	3	4	no	0.045993	1062.44

Table 2: CPU time and total damage for equity as Rawls' principle, with an increasing cardinality.

There are several metrics that can be used to calculate the importance of a vertex in a graph. In particular, sociology has studied this matter deeply, due to its interest for social networks, and has proposed many measures. The metric we use in this work is *betweenness centrality* that considers how often a vertex is along the shortest path between two other vertices. Anthonisse's work [1] and Freeman's work [3] are seminal. We refer to them as "traditional" approach. Nevertheless, we consider a more recent work from Brandes [2] (on which this section is based). We introduce below, more formally, *betweenness centrality* and some other common measures.

Classically, given a graph G, the *distance* between vertices s (source) and t (target) is the minimum length of any path connecting s and t, and we denote it with $d_G(s,t)$. We call σ_{st} the number of the shortest paths from vertex s to vertex t and $\sigma_{st}(v)$ the number of shortest paths from vertex s to vertex t that pass through v. We call $\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$ the *pair-dependency* of a pair $(s,t) \in V$ on an intermediary $v \in V$.

- Closeness centrality (Sabidussi, 1966)

$$C_C(v) = \frac{1}{\sum_{t \in V} d_G(v,t)} \tag{1}$$

- Graph centrality (Hage and Harary, 1995)

$$C_G(v) = \frac{1}{\max_{t \in V} d_G(v,t)} \tag{2}$$

- Stress centrality (Shimbel, 1953)

$$C_S(v) = \sum_{s \neq v \neq t \in V} \sigma_{st}(v) \tag{3}$$

- Betweenness centrality (Freeman, 1977; Anthonisse, 1971)

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{4}$$

In order to calculate centrality, we have to consider each combination of vertices in the graph and to find the shortest path between them. Each vertex that is along a shortest path gets a point. The resulting scores for each vertex in the ratio between the total number of shortest paths from node to node and the number of paths that pass through, and represents its *betweenness centrality*.

Typically, *betweenness centrality* is determined through a two steps procedure. First, we compute length and number of shortest paths between all pairs of vertices, then sum all pair-dependencies. From a computational time complexity point of view, traditional algorithms, which are based on the Floyd-Warshall algorithm, belong to $\Theta(n^3)$. Brandes introduces a faster algorithm for *betweenness centrality* computation that belongs to $O(nm + n^2 logn)$ [2]. We implemented it[1] in order to produce a ranking of the vertices of a given instance and applied it to instances that we can not solve directly with the method exposed in [4], due to their size. For example, we consider the instance that corresponds to the parameters vector $(70, 0.5, 10, 3, 4)$, where, in particular, 70 is the cardinality of the set of vertices. Thus, we calculate the *betweenness centrality* of its vertices. Figure 1 provides a graphical representation of the instance graph and of the centrality of the vertices (which are depicted in the figure with different color and size, depending on their centrality).

Consequently, we transform the graph corresponding to the instance $(70, 0.5, 10, 3, 4)$ into a new reduced one, removing the ten less central vertices and considering the graph induced by the remaining vertices. We then solve the problem for this reduced instance, as explained above. Table 3 and Table 4 show the outcome we obtain with the reduced instance respectively with equity as risk sharing and equity as Rawls's principle. The procedure enables the handling of instances that we could not solve directly by means of the solver and, in general, speed up the process.

[1] We exploit the public library GraphStream that is hosted by the University of Le Havre and has beens initiated and maintained by members of the RI_2C research team from the LITIS computer science lab, **http://graphstream-project.org/**.

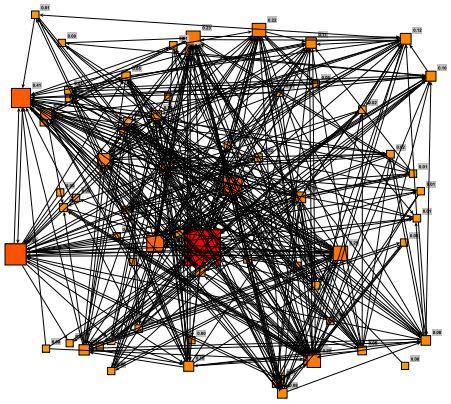


Figure 1: Graphical representation of the instance (70,0.5,10,3,4) showing the most central nodes. Big red squares have an high betweenness centrality.

Test	Card.	Gr.d.	Cap.	Comm.	Zones	Equity	cpu time	Total damage
original	70	0.5	10	3	4	yes	exit code	exit code
	70	0.5	10	3	4	no	exit code	exit code
reduced	70	0.5	10	3	4	yes	309.803	146.152
	70	0.5	10	3	4	no	309.847	145.569

Table 3: CPU time and Total damage for equity as risk sharing, for reduced instance.

Test	Card.	Gr.d.	Cap.	Comm.	Zones	Equity	cpu time	Total damage
original	70	0.5	10	3	4	yes	0.051992	145.569
	70	0.5	10	3	4	no	0.081987	145.569
reduced	70	0.5	10	3	4	yes	0.044993	145.569
	70	0.5	10	3	4	no	0.072988	145.569

Table 4: CPU time and Total damage for equity as Rawls' principle, for reduced instance.

References

[1] J. Anthonisse. The rush in a directed graph. Technical Report BN9/71, Stichting Mahtematisch Centrum, Amsterdam, 1971.

[2] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25:163–177, 2001.

[3] L. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, Mar. 1977.

[4] F. Roda, P. Hansen, and L. Liberti. The price of equity in the hazmat. In L. Adacher, M. Flamini, G. Leo, G. Nicosia, A. Pacifici, and V. Piccialli, editors, *CTW*, pages 235–238, 2011.

Parameterized And/Or Graph Solution

Uéverton dos Santos Souza[1], Fábio Protti[1], and Maise Dantas da Silva[1]

[1]Fluminense Federal University, Niterói, RJ, Brazil

Abstract. And/or graphs are data structures that have been used to model several problems in computer science. An and/or graph is an acyclic digraph containing a single source vertex, such that every vertex v has a label $f(v) \in \{\texttt{and},\texttt{or}\}$ and (weighted) edges represent dependency relations between vertices: a vertex labeled `and` depends on all of its out-neighbors, while a vertex labeled `or` depends on only one of its out-neighbors. The NP-hard optimization problem associated with an and/or graph G, denoted by Min-and/or, consists of finding a *minimum solution subgraph* H of G, where a solution subgraph must contain the source and obey the following rule: if an `and`-vertex (resp. `or`-vertex) is included in H then all (resp. one) of its out-edges must also be included in H. In this paper we deal with the natural question of parameterizing the cost of the solution subgraph. We prove that Min-and/or is fixed-parameter tractable when asked whether there is a solution subgraph H of cost at most k, where the cost of H is the sum of the weights of its edges.

Keywords: and/or graphs, fixed parameter tractability, NP-hardness

1 Introduction

In this paper we consider the complexity of a problem associated with important data structures, the so-called *and/or graphs*. An and/or graph is an acyclic digraph containing a single source vertex (that reaches all other vertices by directed paths), such that every vertex $v \in V(G)$ has a label $f(v) \in \{\texttt{and},\texttt{or}\}$. In such digraphs, edges represent dependency relations between vertices: a vertex labeled `and` depends on all of its out-neighbors (conjunctive dependency), while a vertex labeled `or` depends on only one of its out-neighbors (disjunctive dependency). Figure 1 shows an example of an and/or graph, where `and`-vertices have an arc around its out-edges.

And/or graphs were used for modeling problems originated in the 60's within the domain of Artificial Intelligence [8]. Since then, they have successfully been applied to other fields, such as Operations Research [7], Robotics [2], Software Engineering [6] among others. In addition, and/or graphs also generalize directed hypergraphs [5]. The main optimization problem associated with and/or graphs is formally defined below.

Min-and/or
Instance: An and/or graph $G = (V, E)$, containing a single source vertex s, and such that each edge e of G has an integer weight $\tau(e) > 0$.
Goal: Determine the minimum cost of a subdigraph $H = (V', E')$ of G (*solution subgraph*) satisfying the following properties:

- $s \in V'$;
- if a vertex v is in V' and $f(v)$=`and` then every out-edge of v belongs to E';
- if a non-sink vertex v is in V' and $f(v)$=`or` then exactly one out-edge of v belongs to E'.

In 1974, Sahni [9] showed that MIN-AND/OR is NP-hard. However, when restricted to trees, the problem can be solved in polynomial time. In [1], Adelson-Velsky showed that the problem MIN-AND/OR has interesting connections with real-word applications in scheduling.

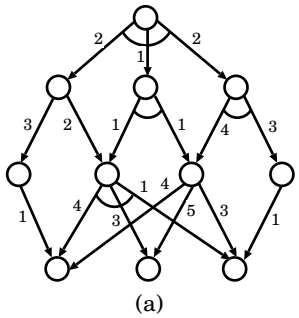


Figure 1: (a) A weighted and/or graph.

Motivated by the large applicability as well as the hardness of MIN-AND/OR, we study new complexity aspects of such problem, from a parameterized point of view. This approach is justified by the fact that many applications are concerned with satisfying a low cost limit. In Section 2 we prove that MIN-AND/OR is fixed-parameter tractable when asked whether there is a solution subgraph H of cost at most k, where the cost of H is the sum of the weights of its edges.

2 Fixed Parameter Tractability of MIN-AND/OR

The Parameterized Complexity Theory was proposed by Downey and Fellows [3] as a promising alternative to deal with NP-hard problems described by the following general form: given an object x and a nonnegative integer k, does x have some property that depends only on k (and not on the size of x)? In parameterized complexity theory, k is fixed as the *parameter*, considered to be *small* in comparison with the size $|x|$ of object x. It may be of high interest for some problems to ask whether they admit deterministic algorithms whose running times are exponential with respect to k but polynomial with respect to $|x|$.

Definition 1. [4] *A parameterized problem* Π *is fixed-parameter tractable, or* FPT*, if the question "*$(x,k) \in \Pi$*?" can be decided in running time* $f(|k|).|x|^{O(1)}$*, where* f *is an arbitrary function on nonnegative integers. The corresponding complexity class is called* FPT*.*

We denote by MIN-AND/OR(k) the parameterized version of MIN-AND/OR which asks whether there is a solution subgraph of cost at most k. In [10], some variants of MIN-AND/OR(k) are studied, such as: MIN-AND/OR(k,r), where every `or`-vertex has at most r out-edges with the same weight; MIN-AND/OR^0(k), whose domain includes and/or graphs with zero-weight edges; and MIN-X-Y(k), a natural generalization of MIN-AND/OR(k) where every vertex v_i has a label x_i-y_i to mean that v_i depends on x_i of its y_i out-neighbors. Table 1 illustrates results in the literature for MIN-AND/OR(k) and its variations. As we can observe, the question of classifying the parameterized problem MIN-AND/OR(k) was open up to now. The following theorem closes this question, using a reduction to a problem kernel.

	FPT	W[1]-hard	W[2]-hard
MIN-AND/OR(k, r)	X	–	–
MIN-AND/OR(k)	?	?	?
MIN-X-Y(k)	–	X	–
MIN-AND/OR$^0(k)$	–	–	X

Table 1: Results on variations of MIN-AND/OR(k)

Theorem 2.1. MIN-AND/OR(k) *is fixed-parameter tractable.*

Proof. The proof is based on the following correct reduction rules, that must be applied once in the order given below:

1. For every `or`-vertex, select an edge with minimum weight and remove all other out-edges.
2. Assign label `and` to every sink.
3. Color each edge e with a distinct color c_e and set the flag of c_e as the weight of e.
4. For each `and`-vertex v_i having one `or`-in-neighbor with more than k out-edges do:
   a) if v_i has more than one `and`-out-neighbor, where $A_{v_i} = \{u_1, u_2, \ldots, u_t\}$ is the set of `and`-out-neighbors of v_i, then: (i) for each vertex $u_j \in A_{v_i}$ create a new `and`-vertex w_j; (ii) create the edges (v_i, w_1) and (w_j, w_{j+1}) for all $j < t$; (iii) assign the same color and weight of (v_i, u_j) to the edge pointing to w_j; (iv) for every edge (u_j, v_r) where $u_j \in A_{v_i}$, create an edge (w_t, v_r) with the same color and weight as (u_j, v_r); (v) remove all `and`-out-edges of v_i distinct from (v_i, w_1).
   b) if v_i has only one `and`-out-neighbor v_j and some `or`-out-neighbors then: (i) create a new `and`-vertex w and an edge $e = (v_i, w)$ with the same color and weight as (v_i, v_j); (ii) for every out-neighbor u of v_j create an edge (w, u) with the same color and weight as (v_j, u); (iii) for every edge (v_i, r) where r is an `or`-vertex, create an edge (w, r) with the same color and weight as (v_i, r); (iv) remove the out-edges of v_i distinct from (v_i, w).
   c) if v_i has only `or`-out-neighbors then: (i) select an `or`-out-neighbor r with minimum out-degree; (ii) for each edge (r, v_j), create a new `and`-vertex w_j and edges (v_i, w_j), (w_j, v_j), where (w_j, v_j) has the same color and weight as (r, v_j), and (v_i, w_j) has the same color and weight as (v_i, r); (iii) for each edge (v_i, s) $(r \neq s)$, create an edge (w_j, s) (for all j) with the same color and weight as (v_i, s); (iv) assign label `or` to v_i; (v) remove all out-edges of v_i pointing to `or`-vertices.
5. Assign label `or` to every `and`-vertex with only one out-neighbor.
6. For each `or`-vertex v_i with more than k out-edges do: (i) create k `or`-vertices $w_1, \ldots, w_k$; (ii) for each edge (v_i, v_t) such that $\tau(v_i, v_t) = j$ and $j \leq k$, create an edge (v_i, w_j) with weight j and same color as (v_i, v_t); (iii) for each out-neighbor v_t of v_i such that $\tau(v_i, v_t) = j$, $j \leq k$, and $v_t \neq w_j$, do: for each edge (v_t, r) create an edge (w_j, r) with the same color and weight as (v_t, r); (iv) remove every out-edge of v_i not created in this step.
7. While there are `or`-vertices with more than k out-neighbors at distance at most k from s, do: repeat rules 2,3 and 4.
8. For each `and`-vertex v_i, if the sum of weights of its out-edges is greater than k remove it.
9. For each edge $e \in E(G)$, if $\tau(e) > k$ then remove it.
10. For every vertex v_i, if the weight of a shortest path from s to v_i is greater than k then remove it.

11. If some vertex has become unreachable from s then remove it.
12. Remove every vertex which has become a sink.
13. Remove each `and`-vertex such that some of its out-edges have been removed.
14. Repeat rules 12 and 13 while needed.

At this point, each vertex has at most k out-neighbors and is at distance at most k from s, therefore the graph has $O(k^{k+1})$ vertices. On the other hand, the graph may contain a large number of edges due to the existence of parallel edges created by previous rules. Say that two colors assigned to the same subset of edges (by disregarding parallelism of edges) are in the same *group* of colors. Since the graph has $O(k^{k+1})$ vertices, the number of distinct groups of colors is bounded by a function of k.

15. For each group of colors, select a color with minimum flag and remove all edges (v, w) colored with another color of this group such that v is an `or`-vertex
16. As parallel edges and edges with the same color have been created by rules 4,5 and 6, apply successively rules 4, 5, 6 *reversibly* until the graph has no parallel edges and only one edge per color.

After applying the rules, the final graph has size bounded by a function of k. Only vertices and edges redundant or not belonging to a solution subgraph of cost at most k have been removed, and a solution subgraph of cost k in this graph implies a solution subgraph of cost k in the original graph. Thus, the above reduction rules obtain a kernel to the problem, i.e., MIN-AND/OR(k) is fixed-parameter tractable. □

References

[1] Adelson-Velsky, G. M., Gelbukh, A. F., Levner, E., A fast scheduling algorithm in AND-OR graphs, *Topics in Applied and Theoretical Mathematics and Computer Science*, WSEAS Press, p. 170-175, 2001.

[2] Cao, T., Sanderson, A. C., And/or net representation for robotic task sequence planning, *IEEE Trans. Systems Man Cybernet, Part C: Applications and Reviews*, v. 28, p. 204-218, 1998.

[3] Downey, R., Fellows, M., *Parameterized Complexity*, Springer-Verlag, 1999.

[4] Flum, J., Grohe, M., *Parameterized Complexity Theory*, Springer, 2006.

[5] Gallo, G., Longo, G., Nguyen, S., Pallottino, S., Directed hypergraphs and applications, *Discrete Applied Mathematics*, v. 42, p. 177-201, 1993.

[6] Medeiros, R. P., Souza, U. S., Protti, F., Murta, L. G. P., Optimal Variability Selection in Product Line Engineering, *Proc. of the 24th International Conference on Software Engineering and Knowledge Engineering - SEKE 2012.*

[7] Morabito, R., Pureza, V., A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem, *Annals of Operations Research*, v. 179, p. 297-315, 2010.

[8] Nilsson, N. J., Problem-reduction representations, in Dojny, R. F. and Eakins, M., editors, *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, 1971.

[9] Sahni, S., Computationally related problems, *SIAM Journal on Computing*, v. 3, n. 4, p. 262-279, 1974.

[10] Souza, U. S., Protti, F., Dantas da Silva, M., Revisiting the Complexity of And/Or Graph Solution, arXiv:1203.3249v1 [cs.CC], March 2012.

Finding the Colors of the Secret in Mastermind

Thatchaphol Saranurak[1]

[1]Saarland University

In the Mastermind game with n positions and k colors, a secret $z \in [k]^n$ has to be found. We can only repeatedly ask queries $x \in [k]^n$ and learn $eq(x,z) = |\{i \in [n] \mid z_i = x_i\}|$ (number of black pegs) and $\pi(x,z) = \max_{\pi \in S_n} |\{i \in [n] \mid z_i = x_{\pi(i)}\}|$ (total number of black and white pegs). The aim is to use as few queries as possible.

When only black pegs are used, the minimum number of queries is $\Theta(\frac{n \log k}{\log n})$ for $k \leq n^{1-\epsilon}, \epsilon > 0$ constant. For $n/\log^{O(1)} n \leq k \leq n$, there is still a gap between $O(n \log\log n)$ and $\Omega(n)$ according to the recent result [2] from SODA 2013.

Let $C^* \subseteq [k]$ be the set of colors in z, and $c^* = |C^*|$. In this paper, we show that when the information from white pegs is also used, the parameter which captures the complexity of the problem is actually c^* rather than k. Namely, if $c^* \leq n^{1-\epsilon}$, we have a tight bound, and if $n/\log^{O(1)} n \leq c^* \leq n$ there is a same kind of gap given that $k \leq n^2$.

To show this, we devise a new algorithm for identifying C^* using information from $\pi(\cdot)$ only. The previous algorithm is randomized and asks $O(k/n + c^* \log\log c^* + n)$ queries. Our new algorithm is deterministic and asks $O(k/n + c^* \log\log c^* + c^* \frac{\log(n/c^*)}{\log c^*})$ queries, which as fast as the previous algorithm for all c^*.

1 Introduction

In Mastermind game with n positions and k colors, a secret $z \in [k]^n$ has to be found. We can only repeatedly ask queries $x \in [k]^n$ and learn $eq(x,z) = |\{i \in [n] \mid z_i = x_i\}|$ (number of black pegs) and $\pi(x,z) = \max_{\pi \in S_n} |\{i \in [n] \mid z_i = x_{\pi(i)}\}|$ (total number of black and white pegs). Let $b(n,k)$ be the minimum numbers of queries for identifying z when we only have access to $eq(\cdot)$, in other words, we use only information from black pegs. Let $bw(n,k)$ be the same quantity, but when we have access to both $eq(\cdot)$ and $\pi(\cdot)$, in other words, we also use the information from white pegs.

There is a sequence of papers proving the bounds for $b(n,k)$, which are discussed in [2]. There are tight bounds of $b(n,k) = \Theta(\frac{n \log k}{\log n})$ when $k \leq n^{1-\epsilon}, \epsilon > 0$ constant, and $b(n,k) = \Theta(k + b(n,n))$ when $k > n$. But if $n/\log^{O(1)} n \leq k \leq n$, there is still a gap between $O(n \log\log n)$ and $\Omega(n)$.

For the complexity of $bw(n,k)$, it is shown in [2] that

$$bw(n,k) = \begin{cases} \Theta(b(n,k)), & \text{if } k \leq n \\ \Theta(k/n + b(n,n)), & \text{if } k > n. \end{cases}$$

Let $C^* \subseteq [k]$ be the set of colors in z, and $c^* = |C^*|$. Note that $c^* \leq \min\{n, k\}$. In this paper, we prove the more elaborated bound.

Theorem 1.1.

$$bw(n,k) = \begin{cases} \Theta(b(n,c^*)), & \text{if } k \leq n \\ \Theta(k/n + b(n,c^*)), & \text{if } k > n. \end{cases}$$

This means that the parameter which captures the complexity of $bw(n,k)$ is actually c^* rather than k. Namely, if $c^* \leq n^{1-\epsilon}, \epsilon > 0$, then we have a tight bound because the bound for $b(n,c^*)$ is tight. If $n/\log^{O(1)} n \leq c^* \leq n$, there is still a gap between $O(n \log\log n)$ and $\Omega(n)$ given that $k \leq n^2$.

This bound tells us that the hard instances of the Mastermind game with black and white pegs are those with $c^* = \min\{n,k\}$, and therefore if $c^* \ll k$, then white pegs help us identify the secret z asymptotically faster than using only black pegs. For example when $c^* = 2$, $bw(n,n) = O(n/\log n)$, but $b(n,n) = \Omega(n)$.

1.1 Overview of the Technique

To show these bounds, we devise a new algorithm proving the new upper bound for $w(n,k)$ where $w(n,k)$ is defined to be the minimum numbers of queries for identifying the colors C^* when we have access to only $\pi(\cdot)$. The previous algorithm in [2] is randomized and reduces the problem to identify the secret itself using only the information from black pegs again. It gives us the bound $w(n,k) = O(k/n + c^* + b(c^*,n))$ in expectation. The current best bounds of $b(n,k)$ are $= O(n \log\log n + k)$ and $\Omega(n+k)$ when $k \geq n$. Hence, the algorithm asks $O(k/n + c^* \log\log c^* + n)$ and $\Omega(k/n + c^* + n)$ queries.

Our new algorithm for identifying C^* is deterministic, and gives the stronger bound of $w(n,k)$ for all c^*.

Theorem 1.2. *For Mastermind with n positions, k colors, and only access to $\pi(\cdot)$, given that the secret z contains the colors C^*, where $|C^*| = c^*$, then we can deterministically identify C^* using $O(c^* + c^* \frac{\log(n/c^*)}{\log c^*}))$ queries if $k \leq n$, and $O(k/n + c^* \log\log c^* + c^* \frac{\log(n/c^*)}{\log c^*}))$ queries if $k > n$. In other words,*

$$w(n,k) = \begin{cases} O(c^* + c^* \frac{\log(n/c^*)}{\log c^*}), & \text{if } k \leq n \\ O(k/n + c^* \log\log c^* + c^* \frac{\log(n/c^*)}{\log c^*}), & \text{if } k > n \end{cases}$$

Our approach for proving the upper bound for $w(n,k)$ is to reduce some sub-problem to a coin weighing problem similarly to the approach for proving the upper bound of $b(n,k)$ in [2].

We show our new algorithm in Section 2, and prove the bound of $bw(n,k)$ in Section 3.

2 The Algorithm for Finding the Colors of the Secret

In this section, we give an algorithm for identifying C^* using information from $\pi(\cdot)$ only. We need the following tools.

Coin Weighing Problem: We are given n coins with weight zero, d out of which are fake with weight one. We can weigh any subsets of coins and then learn the sum of the weight of the selected coins. The following two theorems will be used in our proofs.

Theorem 2.1. *[1] There is a deterministic polynomial time algorithm for identifying d fake coins using $O(\frac{d\log(n/d)}{\log d})$ weighing.*

The problem can be generalized to the case when there each coin may have different weight.

Theorem 2.2. *[1] Given that the sum of the weight of all coins is $O(n)$, there is a deterministic polynomial time algorithm for identifying the weight of each coin using $O(\frac{n}{\log n})$ weighing.*

Now we show our algorithm, which works in 4 phases. Here, by the "answer" of x we always mean $\pi(x,z)$. Our algorithm works in 4 phases.

1. Find a dummy color $c \in [k]\backslash C^*$. This can be done easily by asking monochromatic code queries x at most c^*+1 times until we find one with $\pi(x,z)=0$.

2. Learn c^* and find the first candidate set C. We ask roughly k/n queries containing all colors. This can be done by repeatedly asking queries containing distinct colors (the last query contains the remaining colors and dummy color). From this, we learn c^* as the sum of all answers. Let C be the set of all colors contained in all queries x with $\pi(x,z)>0$. It is clear that $C^* \subseteq C$ and $|C| \le c^* \cdot n$.

3. Reduce the size of the candidate set C until $|C| \le n$. This is the main phase. We will show in details how to achieve this using $O(c^* \log\log c^*)$ queries.

4. Find C^* from C where $|C| \le n$. We reduce the problem to coin weighing problem. We assign the "weight" one to any color $c \in C^*$, and zero to $c \notin C^*$. We can simulate weighing of any subset $D \subset C$ by asking x whose first $|D|$ positions contains the colors in D, and the remaining part contains the dummy color. We have that the weight of D is $\pi(x,z)$. By Theorem 2.1, we can identify C^* within $O(\frac{c^*\log(n/c^*)}{\log c^*})$ queries.

Thus, $w(n,k) = O(c^*+1+k/n+c^*\log\log c^* + \frac{c^*\log(n/c^*)}{\log c^*}) = O(k/n+c^*\log\log c^* + c^*\frac{\log(n/c^*)}{\log c^*}))$. Note that when $k \le n$, we even have $w(n,k) = O(c^* + c^*\frac{\log(n/c^*)}{\log c^*}))$. Because we can only do Phases 1 and 4, given that there is dummy color, in other words, $c^* < k$. If $c^* = k$, then we are done after the first c^* monochromatic queries. Therefore, this gives us Theorem 1.2.

Now we show how to do Phase 3.

Lemma 2.3. *Given a candidate set $C \supseteq C^*$ of size at most $k'n$, we can find the new candidate set C' of size at most $\frac{3k'}{4}n$ in $O(\frac{c^*}{\log(4c^*/k')})$ queries.*

Proof. First, we ask k' queries containing all colors in C like in Phase 2. Observe that there are at least $\frac{k'}{2}$ queries with the answer at most $\frac{2c^*}{k'}$, otherwise the sum of all answers is more than c^*, which is a contradiction. Let Q be the set of such queries.

For each $x \in Q$, we partition the positions of x into $\frac{4c^*}{k'}$ blocks of equal size. Since $\pi(x,z) \le \frac{2c^*}{k'}$, but there are $\frac{4c^*}{k'}$ blocks, at least half of all blocks contribute nothing to the answer. We call these blocks 0-blocks. We will identify the 0-blocks (which contain only unused colors) to reduce the size of the candidate set C. To do so, we again reduce the problem to coin weighing problem.

We view each block as a coin. The weight of the block is the contribution of the block to the answer $\pi(x, z)$. Hence, the weight of 0-blocks is 0. By Theorem 2.2, we can identify all 0-blocks of x in $O(\frac{c^*/k'}{\log(4c^*/k')})$ queries. So at least half of all colors in x are eliminated.

We do coin-weighing for each $x \in Q$. So there are $O(k' \cdot \frac{c^*/k'}{\log(4c^*/k')}) = O(\frac{c^*}{\log(4c^*/k')})$ queries in total. To sum, for at least half of all queries, we eliminate at least half of all colors in such queries. So at least $\frac{1}{4}$-fraction of the colors are eliminated. □

Lemma 2.4. *Given the candidate set C of size at most c^*n, we can find the new candidate set C' of size at most n in $O(c^* \log\log c^*)$ queries.*

Proof. By repeating the algorithm in Lemma 2.3 for constant number of times, we can actually half the size of C in the same query complexity.

Now we just apply Lemma 2.3 repeatedly. We have that the total query complexity is

$$O(\sum_{i=0}^{\log c^*} \frac{c^*}{\log(\frac{4c^*}{c^*/2^i})}) = O(c^* \sum_{i=0}^{\log c^*} \frac{1}{\log(2^{i+2})}) = O(c^* \sum_{i=0}^{\log c^*} \frac{1}{i+2}) = O(c^* \log\log c^*).$$

□

3 White Pegs Help when the Secret is not Colorful

In this section, we prove our bounds of $bw(n, k)$. The simple main idea is $bw(n, k) \le w(n, k) + b(n, c^*)$ because one approach for identify z is to first identify C^* using only $\pi(\cdot)$, and then identify z using only $eq(\cdot)$. With a simple argument, we have

Proposition 3.1. $b(n, k) = \Omega(k + \frac{n \log k}{\log n}) = \Omega(k + n/\log n)$ *for any* $k \ge 2$.

Proof of Theorem 1.1. When $k \le n$, by Theorem 1.2, $w(n, k) = O(c^* + c^* \frac{\log(n/c^*)}{\log c^*}) = O(c^* + n/\log n)$. Since $b(n, c^*) = \Omega(c^* + n/\log n)$ by Proposition 3.1, we have $bw(n, k) = O(b(n, c^*))$.

When $k > n$, there are two cases. If $c^* \le n/\log^2 n$, we have $w(n, k) = O(k/n + c^* \log\log c^* + c^* \frac{\log(n/c^*)}{\log c^*})$ by Theorem 1.2. Since both $c^* \log\log c^*$ and $c^* \frac{\log(n/c^*)}{\log c^*}$ are in $O(n/\log n)$, but $b(n, c^*) = \Omega(n/\log n)$ by Proposition 3.1, we are done.

If $n/\log^2 n < c^* \le n$, we use the bound from [2]: $w(n, k) = O(k/n + c^* + b(c^*, n))$. Note that $b(c^*, n) \le n + b(c^*, c^*)$ because we can trivially ask the monochromatic queries n times to identify the colors in the secret. So $w(n, k) = O(k/n + n + b(c^*, c^*))$. Since $b(n, c^*) = \Omega(n)$ by Proposition 3.1 and $b(n, c^*) \ge b(c^*, c^*)$, we are done. □

We omit the straight-forward lower bounds because of space.

References

[1] Nader H. Bshouty. Optimal algorithms for the coin weighing problem with a spring scale. In *COLT*, 2009.

[2] Benjamin Doerr, Reto Spöhel, Henning Thomas, and Carola Winzen. Playing mastermind with many colors. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms*, pages 695–704, 2013.

1-factors and circuits of cubic graphs

Eckhard Steffen[1]

[1]Paderborn University
Paderborn Institute for Advanced Studies in Computer Science and Engineering
Zukunftsmeile 1, 33102 Paderborn , (es@upb.de)

1 Introduction and Motivation

We consider cubic graphs G with vertex set $V(G)$ and edge set $E(G)$. If a cubic graph has two disjoint 1-factors, then it is 3-edge-colorable, and its edge set can be covered by three 1-factors. Hence, if the chromatic index of G is 4, then any two 1-factors of G intersect. For $k \geq 1$, let $m_k(G) = \max\{|\bigcup_{i=1}^{k} M_i| \colon M_1, \ldots, M_k \text{ are 1-factors of } G\}$, and $\mu_k(G) = |E(G)| - m_k(G)$. We consider 1-factor-covers of cubic graphs.

It is an open problem whether there exists an integer k such that $\mu_k(G) = 0$ for every bridgeless cubic graph. Berge (see [8]) conjectured that $\mu_5(G) = 0$ for every bridgeless cubic graph. In [5] it is shown that Berge's conjecture is equivalent to the following conjecture.

Conjecture 1 (Berge-Fulkerson Conjecture [3])**.** *Every bridgeless cubic graph G has six 1-factors such that every edge of G is contained in precisely two of them.*

The following conjecture of Fan and Raspaud is true if the Berge-Fulkerson Conjecture is true.

Conjecture 2 (Conjecture of Fan and Raspaud [2])**.** *Every bridgeless cubic graph has three 1-factors M_1, M_2, M_3 such that $M_1 \cap M_2 \cap M_3 = \emptyset$.*

Since the complement of a 1-factor of a cubic graph is a 2-factor, covers with 1-factors are closely related to cycle covers. A cycle cover of a graph G is a set $\mathcal{C}$ of cycles such that every edge of G is contained in at least one cycle. It is a double cycle cover if every edge is contained in precisely two cycles, and a k-(double) cycle cover ($k \geq 1$) if $\mathcal{C}$ consists of at most k cycles. The length of a cycle cover $\mathcal{C}$ is the number of edges of $\mathcal{C}$. Celmins and Preissmann independently formulated the 5-Cycle-Double-Cover-Conjecture which is a stronger version of the Cycle-Double-Conjecture of Seymour and Szekeres (see [8]).

Conjecture 3 (see [8])**.** *Every bridgeless graph has a 5-cycle double cover.*

This conjecture is equivalent to its restriction to cubic graphs. In this case a cycle is a 2-regular graph whose components are circuits. A 3-edge-colorable cubic graph G has a 2-cycle cover of length $\frac{4}{3}|E(G)|$. Alon and Tarsi stated the following conjecture.

Conjecture 4 ([1])**.** *Every bridgeless graph G has a cycle cover of length at most $\frac{7}{5}|E(G)|$.*

It is more or less obvious that all the aformentioned conjectures are true for cubic graphs which are 3-edge-colorable. Hence we concentrate our studies on bridgeless cubic graphs G with $\mu_2(G)$, $\mu_3(G) > 0$.

Another well known conjecture which is equivalent to its restriction to cubic graphs is Tuttes 5-flow conjecture.

Conjecture 5 (5-flow conjecture [7])**.** *Every bridgeless graph has a nowhere-zero 5-flow.*

2 Cores of cubic graphs

We introduce the notion of cores of cubic graphs and prove serveral partial results for the above mentioned conjectures.

If $X \subseteq E(G)$, then $G[X]$ denotes the graph whose vertex set consists of all vertices of edges of X and whose edge set is X.

Let G be a cubic graph that has three 1-factors M_1, M_2 and M_3. The set of the edges which are in more than one 1-factor is denoted by $\mathcal{M}$, and the set of edges which are not contained in any of the three 1-factors by $\mathcal{U}$; that is, $\mathcal{M} = \bigcup_{i \neq j}(M_i \cap M_j)$ and $\mathcal{U} = E(G) - \bigcup_{i=1}^{3} M_i$. Clearly, $\mathcal{M}$ and $\mathcal{U}$ are disjoint. We consider the graph $G[\mathcal{M} \cup \mathcal{U}]$ which is induced by the union of $\mathcal{M}$ and $\mathcal{U}$. If $M_1 = M_2 = M_3$, then $G = G[\mathcal{M} \cup \mathcal{U}]$. Hence we ask the three 1-factors to be pairwise different in the following definition.

Let M_1, M_2 and M_3 be three pairwise different 1-factors of a cubic graph G, $k \geq 0$, and $|E(G) - \bigcup_{i=1}^{3} M_i| = k$. The *k-core* of G with respect to M_1, M_2, and M_3 is the subgraph G_c of G which is induced by $\mathcal{M} \cup \mathcal{U}$; that is, $G_c = G[\mathcal{M} \cup \mathcal{U}]$. If the value of k is irrelevant, then we say that G_c is a core of G. We first prove some basic properties of cores.

Theorem 6. *Let G_c be a core of a cubic graph G with respect to three 1-factors M_1, M_2 and M_3, and let K_c be a component of G_c. Then $\mathcal{M}$ is a 1-factor of G_c, and*
1) K_c is either an even circuit or it is a subdivision of a cubic multigraph K, and
2) if K_c is a subdivision of a cubic multigraph K, then $E(K_c) \cap \bigcap_{i=1}^{3} M_i$ is a 1-factor of K.

Theorem 7. *Let $k \geq 0$, and G_c be a k-core of a cubic graph G with respect to three 1-factors M_1, M_2 and M_3. Then,*
1) if $k < 3$, then G is 3-edge-colorable.
2) $|V(G_c)| = 2k - 2|\bigcap_{i=1}^{3} M_i|$, and $|E(G_c)| = 2k - |\bigcap_{i=1}^{3} M_i|$.
3) $girth(G_c) \leq 2k$.
4) G_c has at most $2k/girth(G_c)$ components.

3 Main results

Conjecture of Fan and Raspaud

If a core G_c of a cubic graph is a cycle, then we say that G_c is a *cyclic* core. A cubic graph G has three 1-factors M_1, M_2, M_3 such that $M_1 \cap M_2 \cap M_3 = \emptyset$ if and only if G has a cyclic core. Hence, Conjecture 2 can be formulated as a conjecture on cores in bridgeless cubic graphs.

Conjecture 8 (Conjecture 2)**.** *Every bridgeless cubic graph has a cyclic core.*

Theorem 9. *Let $k \geq 0$, and G be a cubic graph with $\mu_3(G) = k$. If $girth(G) \geq k$ $(> k)$, then every k-core of G is bipartite (cyclic).*

Theorem 10. *Let G be a simple bridgeless cubic graph. If $\mu_3(G) \leq 6$, then G has a cyclic core. In particular, if G is trianglefree and $\mu_3(G) \leq 5$, then every $\mu_3(G)$-core is cyclic.*

Berge-Fulkerson Conjecture

Theorem 11. *Let G be a bridgeless cubic graph which has no non-trivial 3-edge-cut. If $\mu_3(G) \leq 4$, then G has a Fulkerson coloring.*

Short cycle covers and 5-cycle cover

Theorem 12. *Let $k \geq 0$, and G be a cubic graph. If G has a bridgeless k-core, then G has a cycle cover of length at most $\frac{4}{3}|E(G)| + 2k$.*

Theorem 13. *Let $k \geq 0$, and G be a cubic graph. If G has a bipartite k-core G_c, then G has an even 4-cycle cover of length at most $\frac{4}{3}|E(G)| + \frac{2}{3}k$. In particular, if G_c is cyclic, then G has an even 3-cycle cover of length at most $\frac{4}{3}|E(G)| + \frac{2}{3}k$.*

Theorem 14. *Let G be a cubic graph that has four 1-factors $M_1, \ldots, M_4$ with $|E(G) - \bigcup_{i=1}^{4} M_i| = k \geq 0$. If $\bigcap_{i=1}^{4} M_i = \emptyset$, then G has a 4-cycle cover of length $\frac{4}{3}|E(G)| + 4k$.*

Theorem 15. *Let G be a cubic graph. If $\mu_4(G) = 0$, then*
1) G has an even 4-cycle cover $\mathcal{C}$ of length $\frac{4}{3}|E(G)|$.
2) G has a 5-cycle double cover.

Following Zhang [8] we say that the Chinese postman problem is equivalent to the shortest cycle cover problem, if the shortest length of a closed trail that covers all edges of G is equal to the length of a shortest cycle cover. This is certainly true for cubic graphs G that have a cycle cover of length $\frac{4}{3}|E(G)|$. Zhang asked the following question (Problem 8.11.4 in [8]): Let $h \geq 5$ and G be a 3-edge-connected, cyclically h-edge-connected graph. If the Chinese Postman problem and the shortest cycle cover problem are equivalent for G, does G admit a nowhere-zero 4-flow? The answer to this question is negative since for $h \in \{5, 6\}$ there are cyclically h-edge connected snarks with $\mu_4(G) = 0$. It is known that $\mu_4(G) = 0$ if G is a flower snark or a Goldberg snark, see [4].

Nowhere-zero 5-flows

Let $k \geq 2$ be a positive integer. A graph H has a nowhere-zero k-flow if there is an orientation of its edges and a function ϕ from the edge set of H into the set of integers such that the Kirchhoff law is satisfied at every vertex of H and $0 < |\phi(e)| \leq k - 1$ for every edge e of H.

Let G be a bridgeless cubic graph. The oddness $\omega(G)$ of G is the minimum number of odd circuits of a 2-factor of G. In [6] it is shown that if G is a cyclically k-edge connected cubic graph and $k \geq \frac{5}{2}\omega(G) - 3$, then G has a nowhere-zero 5-flow.

Since every 1-factor of G has a non-empty intersection with every odd edge-cut of a G it follows that $\omega(G) \leq 2\mu_2(G)$.

Theorem 16. *Let G be a cyclically k-edge-connected cubic graph. If $k \geq 5\mu_2(G) - 3$, then G has a nowhere-zero 5-flow.*

For small values of $\mu_2(G)$ we prove:

Theorem 17. *Let G be a cyclically 6-edge-connected cubic graph. If $\mu_2(G) \leq 2$, then G has a nowhere-zero 5-flow.*

References

[1] N. Alon, M. Tarsi, Covering multigraphs by simple circuits, SIAM J. Algebraic Discrete Methods **6** (1985) 345 - 350

[2] G. Fan, A. Raspaud, Fulkerson's conjecture and circuit covers, J. Combin. Theory Ser. B, **61** (1994) 133-138

[3] D. R. Fulkerson, Blocking and antiblocking pairs of polyhedra, Math. Program. **1** (1971) 168-194

[4] J.L. Fouquet, J.M. Vanherpe, On the perfect matching index of bridgeless cubic graphs, arXiv:0904.1296v1 (2009)

[5] G. Mazzuoccolo, The equivalence of two conjectures of Berge and Fulkerson, J. Graph Theory **68** (2011), 125 - 128

[6] E. Steffen, Tutte's 5-flow conjecture for highly cyclically connected cubic graphs, Discrete Math. **310** (2010) 385–389

[7] W. T. Tutte, A contribution to the theory of chromatic polynomials, Canadian J. Math. **6** (1954) 80-91

[8] C.-Q. Zhang, Integer flows and cycle covers of graphs, Marcel Dekker, Inc. New York, Basel, Hong Kong (1997)

On the Generality of the Greedy Algorithm for Solving Matroid Problems

Lara Turner[1], Matthias Ehrgott[2], and Horst W. Hamacher[1]

[1]Department of Mathematics, University of Kaiserslautern, P. O. Box 3049, 67653 Kaiserslautern, Germany, {hamacher,turner}@mathematik.uni-kl.de
[2]The University of Auckland, Faculty of Engineering, Department of Engineering Science, New Zealand , m.ehrgott@auckland.ac.nz

Dedicated to Professor Francesco Maffioli.
We are very saddened by his sudden death.

We consider a matroid $M = (E, \mathcal{B})$ with ground set E, rank r, base set $\mathcal{B} \subseteq 2^E$ and costs $c(e) \in \mathbb{R}$ for all $e \in E$. The focus is on matroid problems with *universal objective function* defined as follows.

1. Given a matroid base $B \in \mathcal{B}$, the *sorted cost vector* (with respect to c and B) is defined as
$$c_{\geq}(B) := (c_{(1)}(B), \ldots, c_{(r)}(B)). \tag{1}$$
Here $c_{(i)}(B)$, $i = 1, \ldots, r$, is the i^{th}-largest cost coefficient.

2. Given r *universal weights* $\lambda_1, \ldots, \lambda_r \in \mathbb{R}$, the *universal minimum matroid base problem* (*Univ-MMBP*) is
$$\min_{B \in \mathcal{B}} f_\lambda(B) := \sum_{i=1}^{r} \lambda_i \cdot c_{(i)}(B). \tag{2}$$

Univ-MMBP is a rather powerful model, since it combines the numerous combinatorial special cases of matroids with a variety of objectives induced by specific choices of universal weights. This includes the well-known cases of sum-objective and bottleneck-objective by choosing in (2) $\lambda_i = 1$ for all $i = 1, \ldots, r$ and $\lambda_1 = 1,\ \lambda_i = 0$ for all $i = 2, \ldots, r$, respectively. Other objective functions with a lot of potential for applications are the balanced ($\lambda_1 = 1, \lambda_r = -1,\ \lambda_i = 0$ for all $i = 2, \ldots, r-1$), the k-sum ($\lambda_1 = \ldots = \lambda_k = 1,\ \lambda_i = 0$ for all $i = k+1, \ldots, r$), or the algebraic sum ($\lambda_1 = 2,\ \lambda_i = 1$ for all $i = 2, \ldots, r$), and many more.

The goal of this presentation is to show the potential and limits of using the well-known Greedy algorithm and to propose some extensions thereof to solve Univ-MMBP. Among others we show the validity and applications of the following results.

Theorem 1. *1. If all universal weight coefficients are non-negative or non-positive, then Univ-MMBP can be solved by the Greedy-Algorithm.*

2. *If λ has minus-plus form, i.e.*

$$\lambda = (-\alpha_1, \ldots, -\alpha_k, \beta_{k+1}, \ldots, \beta_r) \tag{3}$$

where $\alpha_1, \ldots, \alpha_k, \beta_{k+1}, \ldots, \beta_r \in \mathbb{R}_0^+$ with at least two universal weight coefficients $\alpha_i, \beta_j > 0$ then the resulting minus-plus Univ-MMBP is solvable by at most $min\{m^k, m^{r-k}\}$ applications of the Greedy Algorithm to matroids of the type $M \setminus B/\tilde{B}$

3. *If M is a uniform matroid, and λ has p fixed sign change universal vector form at positions $k_1, \ldots, k_p - 1, k_p$, i.e.*

$$\lambda = (\alpha_1, \ldots, \underbrace{\alpha_{k_1-1}, -\beta_{k_1}}_{(+,-)}, \ldots\ldots, \underbrace{\alpha_{k_p-1}, -\beta_{k_p}}_{(+,-)}, \ldots, -\beta_r) \tag{4}$$

with non-negative α_i and β_j, then an optimal solution B^* for the resulting p-fix Univ-MMBP is obtained by at most m^p many applications of the Greedy Algorithm.

The presentation is based on a paper with the same title and the same authors and the PhD dissertation "'Universal Combinatorial Optimization: Matroid Bases and Shortest Paths"' by Lara Turner.

Graph Products for Faster Separation of 1-Wheel Inequalities

Sven de Vries[1]

[1]Universität Trier

Using graph products we present an $O(|V|^2|E|+|V|^3 \log |V|)$ separation algorithm for the nonsimple 1-wheel inequalities (Cheng and Cunningham, 1997) of the stable set polytope, which is faster (on the sparse instances coming from the conflict graphs of integer programs) than the previously known $O(|V|^4)$ algorithm.

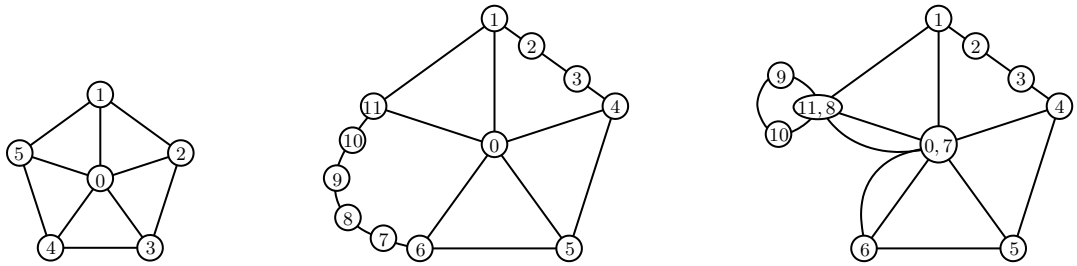
1 Introduction

Let $G = (V, E)$ be a simple connected graph with $|V| = n \geq 2$ and $|E| = m$. A subset of V is called *stable* if it does not contain adjacent vertices of G. The *incidence vector* of a set $N \subseteq V$ is $\chi^N \in \{0,1\}^V$ such that $\chi^N_v = 1$ if $v \in N$ and otherwise $\chi^N_v = 0$. The *stable set polytope* of G, denoted by STAB(G), is the convex hull of incidence vectors of stable sets of G. Some well-known valid inequalities for STAB(G) include the *trivial inequalities* ($x_v \geq 0$ for $v \in V$), the *odd cycle inequalities* ($\sum_{v \in C} x_v \leq k$ where C is the vertex-set of an odd cycle of length $2k+1$), and the *clique inequalities* ($\sum_{v \in K} x_v \leq 1$ where K induces a clique). A clique inequality is called *edge inequality* if the clique has just two vertices.

The *separation problem* for a class $\mathcal{C}$ of valid inequalities is: Given $x^* \in \mathbb{Q}^V$, does x^* violate one of the inequalities in $\mathcal{C}$? If *yes*, exhibit such an inequality. It is an important subroutine of every branch-and-cut-solver for IPs. The separation problem for $\mathcal{C} = \{$trivial, edge and odd-cycle inequalities$\}$ is solvable in $O(nm+n^2 \log n)$. Cheng and Cunningham (1997) describe a separation algorithm for 1-wheel inequalities in time $O(n^4)$, by solving shortest odd walk problems in a complete graph on $O(n)$ vertices. We improve this to $O(n^2m + n^3 \log n)$. As we will show, the separation problem boils down to check whether there exists a minimum wheel with weight below some threshold which in turn can be decided by finding a shortest odd walk in a product graph that does preserve sparsity as it has only $O(m)$ edges

2 1-Wheels

Cheng and Cunningham (1997) consider a wheel with $(2k+1)$ vertices and hub h (for an example of a wheel on 5 vertices and hub, see Fig. 1(a)) and its subdivisions 1(b). Let $1, \ldots, 2k'+1$ be the rim where the *spoke ends* are l_1 up to l_{2k+1}, ordered so that $1 = l_1 < l_2 < \cdots < l_{2k+1}$. Denote the *spoke paths* connecting h to some l_i by P_{l_i} and their subpaths that exclude both ends by $\mathring{P}_{l_i}$. With $|\mathring{P}_{l_i}|$ we denote the number of vertices in $\mathring{P}_{l_i}$. Let the interior of the spoke paths be disjoint. A 1-*wheel* has to fulfill additionally the condition that the cycles



(a) (b) (c)

Figure 1: From a wheel to a simple odd $(11, 5)$-wheel $W(0; 1, 4, 5, 6, 11)$ to a nonsimple odd wheel obtained by identifying vertices $0, 7$ and $8, 11$.

$h, \mathring{P}_{l_i}, l_i, l_i + 1, \ldots, l_{i+1}, \mathring{P}_{l_{i+1}}, h$ are odd for $i = 1, 2, \ldots, 2k + 1$; for a complete specification we denote it by $W(h; k'; l_1, l_2, \ldots, l_{2k+1}; P_{l_1}, P_{l_2}, \ldots, P_{l_{2k+1}})$.

Let $\mathcal{E}$ be the set of the l_i for which the paths P_{l_i} have an *even* number of edges, and let $\mathcal{O}$ be the set of remaining spoke ends. Cheng and Cunningham (1997) show that the inequalities

$$kx_h + \sum_{i=1}^{2k'+1} x_i + \sum_{i\in\mathcal{E}} x_i + \sum_{i=1}^{2k+1} x(\mathring{P}_{l_i}) \le k' + \frac{|\mathcal{E}| + \sum_{i=1}^{2k+1} |\mathring{P}_{l_i}|}{2} \qquad (I_{\mathcal{A}})$$

$$(k+1)x_h + \sum_{i=1}^{2k'+1} x_i + \sum_{i\in\mathcal{O}} x_i + \sum_{i=1}^{2k+1} x(\mathring{P}_{l_i}) \le k' + \frac{|\mathcal{O}| + 1 + \sum_{i=1}^{2k+1} |\mathring{P}_{l_i}|}{2} \qquad (I_{\mathcal{B}})$$

are valid and they give sufficent conditions for them to induce facets. (Here we use $x(\mathring{P})$ for a walk $P = v_0 - \cdots - v_{k+1}$ as a shorthand for $\sum_{i=1}^{k} x_{v_i}$.)

Proposition 2.1 (Cheng and Cunningham, 1997, Prop. 3.4)**.** *Let $\sum_{i=1}^{n} a_i x_i \le a_0$ be a valid inequality for STAB(G) and let v_1 and v_2 be two* nonadjacent *vertices of G. If H is obtained from G by identifying v_1 and v_2 to a single vertex $v_{1,2}$, then $(a_1 + a_2)x_{1,2} + \sum_{i=3}^{n} a_i x_i \le a_0$ is valid for STAB(H).*

Therefore, when speaking of *general* or *nonsimple* 1-wheels we will permit the identification of nonadjacent vertices where the sum of weights of the identified vertices is the new weight.

Lemma 2.2. *For 1-wheels and $I_{\mathcal{A}}$-inequalities assumption $\mathcal{E} = \emptyset$ is wlog. (and similar for $I_{\mathcal{B}}$).*

3 Separation of $I'_{\mathcal{A}}$

Now assume $\mathcal{E} = \emptyset$ for $(I_{\mathcal{A}})$ and call those simple 1-wheels $W(h; k', l_1, l_2, \ldots, l_{2k+1}; P_{l_1}, P_{l_2}, \ldots, P_{l_{2k+1}})$ *simple odd $(2k' + 1, 2k + 1)$-wheels* or shorter *simple odd wheels.* For them $l_{j+1} - l_j$ is odd for $j = 1, \ldots, 2k$ and all spoke paths are odd. The following form of $(I_{\mathcal{A}})$ results:

$$kx_h + \sum_{i=1}^{2k'+1} x_i + \sum_{i=1}^{2k+1} x(\mathring{P}_{l_i}) \le k' + \frac{\sum_{i=1}^{2k+1} |\mathring{P}_{l_i}|}{2}. \qquad (I'_{\mathcal{A}})$$

Because of Proposition 2.1, we focus on general (that is, possibly nonsimple) odd $(2k'+1, 2k+1)$-wheels. Let $\mathrm{ESTAB}(G) := \{x \in [0,1]^V : x_u + x_v \le 1 \ \forall uv \in E\}$ and $\mathrm{CSTAB}(G) := \{x \in$

$\mathrm{ESTAB}(G)$: x fulfills the odd cycle inequalities$\}$ and $\mathrm{W}_{\mathcal{A}}\mathrm{STAB}(G) := \{x \in \mathrm{CSTAB}(G)\colon x$ fulfills all (nonsimple) $I'_{\mathcal{A}}$-inequalities$\}$. Consider the following four-fold of $(I'_{\mathcal{A}})$

$$2 - 2x_h \le \sum_{j=1}^{2k+1} \Big(-2x_h + \big(|\mathring{P}_{l_j}| - 2x(\mathring{P}_{l_j})\big) + \big(|\mathring{P}_{l_{j+1}}| - 2x(\mathring{P}_{l_{j+1}})\big) + 2 \sum_{i=l_j}^{l_{j+1}-1} (1 - x_i - x_{i+1}) \Big). \quad (1)$$

Call the right hand side of (1) the *weight* of the odd wheel with respect to x. We can prove for given x^*, h, that in a *maximally violated inequality* all the spoke walks P_{l_j} are *shortest odd.*

Lemma 3.1. *For a given graph G and $x^* \in ESTAB(G)$ fix a hub h, an odd cycle $v_1, v_2, \ldots, v_{2k'+1}$, and spoke ends $1 = l_1 < l_2 < \cdots < l_{2k+1} \le 2k'+1$. Among all odd wheels with this hub and these spoke ends consider one (with some P_{l_j}'s) such that* (1) *is most violated, that is the right hand side is minimal. Then each P_{l_j} is a shortest* odd *walk from h to v_{l_j} with respect to the edge weights $(1 - x^*_u - x^*_v) \ge 0$ (because of $x^* \in CSTAB(G)$).*

For fix $x^* \in \mathrm{ESTAB}(G)$ let $P^1_{h,k}$ and $P^0_{h,k}$ be shortest odd respectively even of length ≥ 2 (having at least two edges) walks with respect to edge weights $(1 - x^*_u - x^*_v)$ for all $k \in V$; if no such $P^1_{h,k}$ or $P^0_{h,k}$ exists, we set $|\mathring{P}^1_{h,k}| - 2x(\mathring{P}^1_{h,k}) = +\infty$ or $|\mathring{P}^0_{h,k}| - 2x(\mathring{P}^0_{h,k}) = +\infty$ respectively.

Next we analyze the summand of equation (1) and let $\mathring{P}$ be $\mathring{P}^1$. If $l_{j+1} - l_j = 1$ then we have

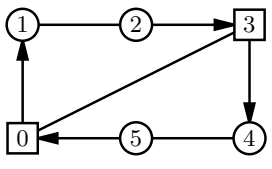
$$\begin{aligned} &- 2x_h + |\mathring{P}_{h,l_j}| - 2x(\mathring{P}_{h,l_j}) + |\mathring{P}_{h,l_{j+1}}| - 2x(\mathring{P}_{h,l_{j+1}}) + 2 \sum_{i=l_j}^{l_{j+1}-1} (1 - x_i - x_{i+1}) \\ &\qquad = 2(1 - x_{l_j} - x_{l_{j+1}}) - 2x_h + |\mathring{P}_{h,l_j}| - 2x(\mathring{P}_{h,l_j}) + |\mathring{P}_{h,l_{j+1}}| - 2x(\mathring{P}_{h,l_{j+1}}) \end{aligned}$$

and going from spoke end l_j to l_{j+1} costs $2(1 - x_{l_j} - x_{l_{j+1}}) - 2x_h + |\mathring{P}_{h,l_j}| - 2x(\mathring{P}_{h,l_j}) + |\mathring{P}_{h,l_{j+1}}| - 2x(\mathring{P}_{h,l_{j+1}})$. For $x \in \mathrm{CSTAB}(G)$ this weight is nonnegative.

Otherwise $l_{j+1} - l_j \ge 3$:

$$\begin{aligned} &- 2x_h + |\mathring{P}_{h,l_j}| - 2x(\mathring{P}_{h,l_j}) + |\mathring{P}_{h,l_{j+1}}| - 2x(\mathring{P}_{h,l_{j+1}}) + 2 \sum_{i=l_j}^{l_{j+1}-1} (1 - x_i - x_{i+1}) \\ =& 2(1 - x_{l_j} - x_{l_j+1} - x_h) + |\mathring{P}_{h,l_j}| - 2x(\mathring{P}_{h,l_j}) \\ &+ 2 \sum_{i=l_j+1}^{l_{j+1}-2} (1 - x_i - x_{i+1}) \qquad + 2(1 - x_{l_{j+1}-1} - x_{l_{j+1}}) - 0x_h + |\mathring{P}_{h,l_{j+1}}| - 2x(\mathring{P}_{h,l_{j+1}}). \end{aligned}$$

So here we would want an edge $\{l_j, l_j + 1\}$ leaving the spoke to contribute $(2 - 2x_h - 2x_{l_j} - 2x_{l_j+1}) + |\mathring{P}_{h,l_j}| - 2x(\mathring{P}_{h,l_j})$, the internal edges $\{i, i+1\}$ not incident with the spokes to contribute $2(1 - x_i - x_{i+1})$, and the final edge $\{l_{j+1} - 1, l_{j+1}\}$ to contribute $(2 - 2x_{l_{j+1}-1} - 2x_{l_{j+1}}) + |\mathring{P}_{h,l_{j+1}}| - 2x(\mathring{P}_{h,l_{j+1}})$. Notice that the second and third weight are positive, if x^* fulfills the edge inequalities, but the first weight *could be negative.*



The *categorical product* $G_1 \cdot D_2$ of a graph G_1 and a digraph D_2 is defined by $V(G_1 \cdot D_2) = V(G_1) \times V(D_2)$ and $A(G_1 \cdot D_2) = \{((u_1, u_2), (v_1, v_2))\colon (u_1, v_1) \in E(G_1)$ and $(u_2, v_2) \in A(D_2)\}$. For $G = (V, E)$ consider the digraph $D := G \cdot F$, with F depicted to the right. We want to embed the violated-odd/even-wheel-with-hub-h-finding-task into this graph. Interpret vertices of type $V \times \{0, 3\}$ as vertices that correspond

to spoke ends. Define for given $x^* \in \mathbb{Q}^V$ and vertex $h \in V$ the weighted digraph $D_h := D$ where the arc $e = ((u,i),(v,j))$ has weight:

$$w^1_e = \begin{cases} 2(1-x^*_u-x^*_v) - 2x^*_h + |\mathring{P}^1_{h,u}| - 2x^*(\mathring{P}^1_{h,u}) + |\mathring{P}^1_{h,v}| - 2x^*(\mathring{P}^1_{h,v}) & \text{if } \{i,j\} = \{0,3\} \\ 2(1-x^*_u-x^*_v) - 2x^*_h + |\mathring{P}^1_{h,u}| - 2x^*(\mathring{P}^1_{h,u}) & \text{if } (i,j) \in \{(3,4),(0,1)\} \\ 2(1-x^*_u-x^*_v) + |\mathring{P}^1_{h,v}| - 2x^*(\mathring{P}^1_{h,v}) & \text{if } (i,j) \in \{(2,3),(5,0)\} \\ 2(1-x^*_u-x^*_v). & \text{if } \{i,j\} \in \{\{1,2\},\{4,5\}\} \end{cases}$$

Lemma 3.2. *For any diwalk $U = (v_1,i_1)-(v_2,i_2)-\cdots-(v_q,i_q)$ in D_h with $\{i_1,i_q\} = \{0,3\} \not\ni i_2,\ldots,i_{q-1}$, with $q \geq 2$, and $x^* \in ESTAB(G)$ holds:*

(a) q is even.

(b) $w^1(U) = -2x^*_h + |\mathring{P}^1_{h,v_1}| - 2x^*(\mathring{P}^1_{h,v_1}) + 2\sum_{i=1}^{q-1}(1-x^*_{v_i}-x^*_{v_{i+1}}) + |\mathring{P}^1_{h,v_q}| - 2x^*(\mathring{P}^1_{h,v_q})$.

(c) If $x^* \in CSTAB(G)$ then $w^1(U) \geq 0$.

(d) If $x^* \in CSTAB(G)$ and $v_1 = v_q$, then $w^1(U) \geq 2 - 2x^*_h$.

Lemma 3.3. *Every (nonsimple) odd $(2k'+1,2k+1)$-wheel $W(h;1=l_1<l_2<\cdots<l_{2k+1} \leq 2k'+1; P^1_{l_1},P^1_{l_2},\ldots,P^1_{l_{2k'+1}})$ of G using shortest spoke paths corresponds to a $(v_1,0) \rightsquigarrow (v_1,3)$ diwalk U (containing at least 3 vertices with second component 0 or 3) in D_h of the same finite weight with respect to w and $x^* \in CSTAB(G)$ and vice-versa.*

Hence, to find a violated odd wheel with hub h and initial spoke end v_1 we have to find a shortest odd wheel with hub h and initial spoke end v_1; if its weight is less than $2(1-x^*_h)$ then it is violated. Otherwise there is no violated odd wheel with hub h and initial spoke end v_1. To do shortest path problem in D_h we need:

Lemma 3.4. *If $x^* \in CSTAB(G)$ and $h \in V(G)$ then D_h contains no w^1-negative dicycle.*

To find a shortest subdivided wheel with hub h and initial spoke end v_1 reduces to finding a shortest $(v_1,0) \rightsquigarrow (v_1,3)$ diwalk encountering at least two more vertices with second component in $\{0,3\}$ in D_h with respect to weights w^1 as defined for Lemma 3.2 and comparing that length to $2-2x^*_h$. If it has only two vertices with second component in $\{0,3\}$, then by Lemma 3.2(d) its weight is $\geq 2-2x^*_h$, hence no violated wheel with hub h and start v exists. This yields:

Corollary 3.5. *Given G and $x^* \in CSTAB(G)$ then there is a violated odd wheel-inequality with hub h starting in v iff D_h contains a $(v,0) \rightsquigarrow (v,3)$ diwalk shorter than $2-2x^*_h$ wrt. w^1.*

Towards the complexity of the separation one has for every hub $h \in V$ first to compute the odd $P^1_{l_j}$-walks having at least one arc each as candidates for spokes with one call to Dijkstra in $O(m+n\log n)$ for $G \cdot K_2$. By applying Johnson's (1977) all-pairs-shortest-path algorithm to D_h in time $O(nm+n^2\log n)$ we check whether there is a $(v,0) \rightsquigarrow (v,3)$ diwalk shorter than $2-2x^*_h$. For n hubs we the total running time is $O(n^2m+n^3\log n)$.

Theorem 3.6. *The separation problem given $x^* \in CSTAB(G)$ for $W_{\mathcal{A}}STAB(G)$ (and similar for $W_{\mathcal{B}}STAB(G)$) can be solved in time $O(n^2m+n^3\log n)$.*

References

Cheng, E. and W. H. Cunningham (1997). Wheel inequalities for stable set polytopes. *Mathematical Programming*, **77**(3):389–421.

Johnson, D. B. (1977). Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, **24**(1):1–13.

Computational complexity of the average covering tree value

Ayumi Igarashi[1] and Yoshitsugu Yamamoto[2]

[1]Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan, Email: igarashi80@sk.tsukuba.ac.jp
[2]Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan, Email: yamamoto@sk.tsukuba.ac.jp

Khmelnitskaya et al. introduced cooperative games with directed graph structure and proposed its single-valued solution concept, called the *average covering tree value*. In this paper we show that the problem for calculating the average covering tree value is #P-complete.

1 Introduction

This paper studies cooperative games with directed graph structure from the viewpoint of computational complexity. A cooperative game with a directed graph was introduced by Khmelnitskaya et al. [4]. The communication structure of this game is given by a directed graph. Khmelnitskaya et al. also proposed this single-valued solution concept, called the *average covering tree value*. To construct the average covering tree value, Khmelnitskaya et al. introduced a so-called covering tree of a directed graph, which preserves the dominance relation among players. The average covering tree value is defined as the average of marginal contribution vectors corresponding to covering trees. The solution coincides with the Shapley value [5] when the game has complete communication structure.

In this paper we discuss the computational complexity of the average covering tree value. We demonstrate that the problem for calculating the average covering tree value is #P-complete. The proof uses a reduction from counting the number of all linear extensions of an arbitrary partial order, which has been shown by Brightwell et al. to be a #P-complete counting problem.

2 Preliminaries

2.1 TU-games with directed graph structure

We consider a cooperative transferable utility game with restricted communication structure, called *digraph games*. A digraph game is represented by a triple (N, v, Γ), where N is a finite set of n players, $v : 2^N \to \mathbb{R}$ is a *characteristic function*, and $\Gamma \subseteq \{ (i,j) \mid i \neq j, i, j \in N \}$ is a collection of directed communication links between players. A subset $S \in 2^N$ is called a *coalition* and $v(S)$ stands for the *worth* of a coalition S. A *payoff vector* $\boldsymbol{x} \in \mathbb{R}^n$ is an n-dimensional vector giving payoff x_i to player $i \in N$.

2.2 Definitions for Digraph

The pair $G = (N, \Gamma)$ is called a *digraph* where N is a finite set of *nodes* and Γ is a collection of *directed links* between nodes. In this paper we only consider digraphs without self-loops. For a digraph $G = (N, \Gamma)$, a sequence of different nodes $(i_1, i_2, \ldots, i_k)$, $k \geq 2$, is a *path* in Γ if $\{(i_h, i_{h+1}), (i_{h+1}, i_h)\} \cap \Gamma \neq \emptyset$ for $h = 1, 2, \ldots, k-1$. A sequence $(i_1, i_2, \ldots, i_k)$, $k \geq 2$, is a *directed path* if $(i_h, i_{h+1}) \in \Gamma$ for all $h \in \{1, 2, \ldots, k-1\}$. A path $(i_1, i_2, \ldots, i_k)$ in Γ is a *cycle* in Γ if $\{(i_k, i_1), (i_1, i_k)\} \cap \Gamma \neq \emptyset$, and a directed path $(i_1, i_2, \ldots, i_k)$, $k \geq 2$, in Γ is a *directed cycle* in Γ if $(i_k, i_1) \in \Gamma$. A digraph $G = (N, \Gamma)$ is said to be *acyclic* if it has no directed cycles. A digraph $G = (N, \Gamma)$ is said to be *transitive* if for all $i, j, k \in N$, $(i, j) \in \Gamma$ and $(j, k) \in \Gamma$ implies $(i, k) \in \Gamma$. For a digraph $G = (N, \Gamma)$, the subset of Γ induced by $S \in 2^N$ is defined as

$$\Gamma|_S := \{\, (i, j) \in \Gamma \mid i, j \in S \,\}.$$

A subset $S \in 2^N$ is *connected* if for any two distinct nodes $i, j \in S$ there is a path in $\Gamma|_S$ between i and j. For $S \in 2^N$, a subset K of S is called a *connected component* of S if K is maximally connected, i.e., K is connected but the set $K \cup \{j\}$ is not connected for any $j \in S \setminus K$. Throughout the paper, it is assumed that N is always connected in the digraph (N, Γ). For a digraph $G = (N, \Gamma)$, for each node $i \in N$ we define its sets of *successors* and *descendants* as

$$\mathrm{suc}^{\Gamma}(i) = \{\, j \in N \mid (i, j) \in \Gamma \,\}$$

and

$$\mathrm{des}^{\Gamma}(i) = \{\, j \in N \mid i = j \text{ or there exists a directed path from } i \text{ to } j \text{ in } \Gamma \,\}.$$

A node $i \in N$ is said to be a *predecessor* of $j \in N$ in Γ if there exists a directed path from i to j in Γ. An acyclic connected digraph (N, T) is said to be a *tree* if it has a unique node without predecessors, the *root*, and for every other node in N there is a unique directed path in T from the root to that node. A node $i \in S$ is an *undominated* node of S if every predecessor j of i in $\Gamma|_S$ is a descendant of i in $\Gamma|_S$. A node $i \in S$ is a *nondominant* node of S if every descendant $j (\neq i)$ of i in $\Gamma|_S$ is a predecessor of i in $\Gamma|_S$. For a digraph (N, Γ) and a subset $S \in 2^N$, let $U_\Gamma(S)$ denote the set of undominated nodes of S and $D_\Gamma(S)$ denote the set of nondominant nodes of S. When a digraph is acyclic, the following lemmas hold.

Lemma 2.1. *Given an acyclic digraph $G = (N, \Gamma)$. Node i is in $U_\Gamma(S)$ if and only if there is no node $j \in S$ such that $(j, i) \in \Gamma|_S$.*

Lemma 2.2. *Given an acyclic digraph $G = (N, \Gamma)$. Node i is in $D_\Gamma(S)$ if and only if there is no node $j \in S$ such that $(i, j) \in \Gamma|_S$.*

A node $i \in N$ is called the *minimum* node of (N, Γ) if i is a descendant of every $j \in N \setminus \{i\}$ in Γ. If an acyclic digraph has the minimum node, it is uniquely determined.

2.3 Definitions for Poset

A *partially ordered set*, or for short *poset* is a pair $P = (N, \Gamma)$, where N is a finite set and Γ is a *partial order* on N, that is, an irreflexive, antisymmetric, and transitive binary relation. Two elements i and j are *comparable* in Γ if either $(i, j) \in \Gamma$ or $(j, i) \in \Gamma$. A *linear order* on N is a partial order Γ in which every pair of elements N is comparable. A *linear extension* of a partial order Γ on N is a linear order Γ' on N such that $(i, j) \in \Gamma'$ whenever $(i, j) \in \Gamma$. Equivalently a linear extension of a partial order Γ on N is a bijection π from N to $\{1, 2, \ldots, |N|\}$ such that for all $i, j \in N$, $(i, j) \in \Gamma$ implies $\pi(i) < \pi(j)$.

2.4 Digraphs and Posets

Every poset $P = (N, \Gamma)$ corresponds to a digraph considering N as the set of nodes and Γ as the set of directed links. This digraph is acyclic and transitive. Conversely, for every acyclic transitive digraph $G = (N, \Gamma)$, Γ is a partial order on N.

Lemma 2.3 (Bondy and Murty [1])**.** *A digraph G is a poset, if and only if G is acyclic and transitive.*

3 The average covering tree value

Khmelnitskaya et al.[4] define the average covering tree value by the average of marginal contribution vectors with respect to specific trees, called *covering trees* of $G = (N, \Gamma)$. In order to construct a covering tree of G, Khmelnitskaya et al.[4] apply the following algorithm.

Algorithm 1 Construct a covering tree of $G = (N, \Gamma)$

1: Set $T = \emptyset$ and $Q_j = \emptyset$ for all $j \in N$.
2: Choose any $i \in U_\Gamma(N)$ and set $Q_i = N \setminus \{i\}$
3: Let $\{K_1, K_2, \ldots, K_m\}$ be the set of connected components of Q_i. For every $k = 1, 2, \ldots, m$, choose $j_k \in U_\Gamma(K_k)$ and set $Q_{j_k} = K_k \setminus \{j_k\}$. Set $T = T \cup \{(i, j_1), (i, j_2), \ldots, (i, j_m)\}$ and $Q_i = \emptyset$.
4: If $Q_j = \emptyset$ for all $j \in N$, then stop. Otherwise, choose $i \in N$ such that $Q_i \neq \emptyset$ and return to Step 3.

We denote by $\mathcal{T}^\Gamma$ the set of all covering trees of a digraph G constructed by Algorithm 1.

Definition 3.1. *For a digraph game (N, v, Γ), the marginal contribution vector $\boldsymbol{m}^T$ corresponding to a covering tree $T \in \mathcal{T}^\Gamma$ is the vector of payoffs given by*

$$m_i^T = v(des^T(i)) - \sum_{j \in suc^T(i)} v(des^T(j)), \quad \text{for all } i \in N. \tag{3.1}$$

Definition 3.2 (ACT(N, v, Γ))**.** *For a digraph game (N, v, Γ), the average covering tree value is the average of the marginal contribution vectors $\boldsymbol{m}^T$ with respect to all covering trees of the digraph (N, Γ), i.e.,*

$$ACT(N, v, \Gamma) = \frac{1}{|\mathcal{T}^\Gamma|} \sum_{T \in \mathcal{T}^\Gamma} \boldsymbol{m}^T(N, v, \Gamma). \tag{3.2}$$

4 Computational complexity of the average covering tree value

To discuss the computational complexity of the average covering tree value, we give some properties of covering tree.

Lemma 4.1. *Given an acyclic transitive digraph $G = (N, \Gamma)$. Algorithm 1 yields a linear extension of Γ if and only if G has the minimum node.*

Lemma 4.2. *Let $G = (N, \Gamma)$ be an acyclic transitive digraph. If G has the minimum node, then Algorithm 1 potentially yields all linear extensions of Γ.*

For a poset $P = (N, \Gamma)$, let $\mathcal{R}(\Gamma)$ denote the set of all linear extensions of Γ, with the convention that $|\mathcal{R}(\emptyset)| = 1$. By Lemma 4.2, $\mathcal{T}^{\Gamma}$ corresponds to $\mathcal{R}(\Gamma)$ when the digraph (N, Γ) with the minimum node corresponds to a poset. In this case, we obtain another representation of the average covering tree value as follows.

Lemma 4.3. *Given a digraph game (N, v, Γ) such that the digraph $G = (N, \Gamma)$ is acyclic and transitive. Suppose that G has the minimum node. Then the average covering tree value of player $i \in N$ is rewritten as follows:*

$$\mathrm{ACT}_i(N, v, \Gamma) = \frac{1}{|\mathcal{R}(\Gamma)|} \sum_{\substack{S \subseteq N; \\ i \in U_{\Gamma}(S), \\ i \in D_{\Gamma}((N \setminus S) \cup \{i\})}} |\mathcal{R}(\Gamma|_{S \setminus \{i\}})| \cdot |\mathcal{R}(\Gamma|_{N \setminus S})| (v(S) - v(S \setminus \{i\})). \quad (4.1)$$

Then, we can prove the following proposition in a similar way to the proof of Proposition 3 in Faigle and Kern [3].

Proposition 4.4 (# P-completeness of the average covering tree value)**.** *Assume that there exists a polynomial-time algorithm to compute the average covering tree value for given digraph games. Then there exists a polynomial-time algorithm to compute the number of all linear extensions for any partial orders.*

Brightwell and Winkler [2] proved that the problem of counting the number of all linear extensions of an arbitrary partial order is #P-complete. Therefore, computing the average covering tree value is also #P-complete, implying that an efficient algorithm to calculate the average covering tree value is unlikely to exist.

References

[1] Bondy, J.A., Murty, U.S.R. *Graph Theory*, Springer (2008).

[2] Brightwell, G., Winkler, P.: Counting linear extensions is # P-complete. *Order* (1991) 8(3): 175–181.

[3] Faigle, U., Kern, W. : The Shapley value for cooperative games under precedence constraints. *International Journal of Game Theory* (1992) 21: 249–266.

[4] Khmelnitskaya, A.B., Selcuk, Ö., Talman, A.J.J.: The average covering tree value for directed graph games. *CentER Discussion Paper 2012-037, CentER, Tilburg University* (2011) 203–212.

[5] Shapley,L.S.: A value for n-person games. In H.W. Kuhn and A.W. Tucker (eds.): *Contributions to the Theory of Games II*, Princeton University Press, Princeton, NJ (1953) 307–317.