



HAL
open science

Proceedings of the 8th Cologne-Twente Workshop on Graphs and Combinatorial Optimization

Sonia Cafieri, Antonio Mucherino, Giacomo Nannicini, Fabien Tarissan, Leo
Liberti

► **To cite this version:**

Sonia Cafieri, Antonio Mucherino, Giacomo Nannicini, Fabien Tarissan, Leo Liberti (Dir.). Proceedings of the 8th Cologne-Twente Workshop on Graphs and Combinatorial Optimization. 2009. hal-01217897

HAL Id: hal-01217897

<https://hal.science/hal-01217897v1>

Submitted on 20 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



8th Cologne-Twente Workshop on
Graphs and Combinatorial
Optimization

CTW09

École Polytechnique and CNAM

Paris, France, June 2-4, 2009

Proceedings of the Conference

Sonia Cafieri, Antonio Mucherino, Giacomo Nannicini,
Fabien Tarissan, Leo Liberti
(Eds.)

8th Cologne-Twente Workshop on
Graphs and Combinatorial Optimization (CTW09)
École Polytechnique and CNAM
Paris, France, June 2-4, 2009

The Cologne-Twente Workshop (CTW) on Graphs and Combinatorial Optimization started off as a series of workshops organized bi-annually by either Köln University or Twente University. As its importance grew over time, it re-centered its geographical focus by including northern Italy (CTW04 in Menaggio, on the lake Como and CTW08 in Gargnano, on the Garda lake). This year, CTW (in its eighth edition) will be staged in France for the first time: more precisely in the heart of Paris, at the Conservatoire National d'Arts et Métiers (CNAM), between 2nd and 4th June 2009, by a mixed organizing committee with members from LIX, École Polytechnique and CEDRIC, CNAM.

As tradition warrants, a special issue of Discrete Applied Mathematics (DAM) will be devoted to CTW09, containing full-length versions of selected presentations given at the workshop and possibly other contributions related to the workshop topics. The deadline for submission to this issue will be posted in due time on the CTW09 website <http://www.lix.polytechnique.fr/ctw09>.

The Proceedings Editors wish to thank the members of the Program Committee: U. Faigle (Universität zu Köln), J. Hurink (Universiteit Twente), L. Liberti (École Polytechnique), F. Maffioli (Politecnico di Milano), G. Righini (Università degli Studi di Milano), R. Schrader (Universität zu Köln), R. Schultz (Universität Duisburg-Essen), for setting up such an attractive program. Special thanks also go to the anonymous referees. This conference was partially sponsored by: the Digiteo (www.digiteo.fr) foundation, CNRS, and the Thales Chair at LIX.

Editors:
SONIA CAFIERI
ANTONIO MUCHERINO
GIACOMO NANNICINI
FABIEN TARISSAN
LEO LIBERTI
LIX, École Polytechnique
Paris, May 2009

Organization

The CTW09 conference is co-organized by the Laboratoire d'Informatique (LIX) at École Polytechnique and by the Centre d'Etude et Recherche en Informatique du CNAM (CEDRIC) at the Conservatoire National d'Arts et Mtiers (CNAM).

Scientific Committee

- U. Faigle (Universität zu Koln)
- J.L. Hurink (Universiteit Twente)
- L. Liberti (École Polytechnique, Paris)
- F. Maffioli (Politecnico di Milano)
- G. Righini (Università degli Studi di Milano)
- R. Schrader (Universität zu Koln)
- R. Schultz (Universität Duisburg-Essen)

Organizing Committee

- S. Cafieri (LIX, École Polytechnique)
- M.-C. Costa (CEDRIC, CNAM)
- C. Dürr (LIX, École Polytechnique)
- L. Liberti (Chair – LIX, École Polytechnique)
- A. Mucherino (LIX, École Polytechnique)
- G. Nannicini (LIX, École Polytechnique)
- C. Picouleau (CEDRIC, CNAM)
- M.-C. Plateau (GdF)
- J. Printz (CMSL, CNAM)
- E. Rayssac (LIX, École Polytechnique)
- F. Tarissan (LIX, École Polytechnique)

Table of Contents

Traveling Salesman Problem

Lecture Hall A, Tue 2, 08:45–10:15

C. Dong, C. Ernst, G. Jaëger, D. Richter, P. Molitor

Effective Heuristics for Large Euclidean TSP Instances Based on Pseudo Backbones 3

M. Casazza, A. Ceselli, M. Nunkesser

Efficient Algorithms for the Double Traveling Salesman Problem with Multiple Stacks 7

M. Bruglieri, A. Colorni, A. Lue

The Parking Warden Tour Problem 11

Graph Theory I

Lecture Hall B, Tue 2, 08:45–10:15

I. Sau, D. Thilikos

On Self-Duality of Branchwidth in Graphs of Bounded Genus 19

S. Nikolopoulos, C. Papadopoulos

A Simple Linear-Time Recognition Algorithm for Weakly Quasi-Threshold Graphs 23

H. Gropp

From Sainte-Laguë to Claude Berge — French Graph Theory in the Twentieth Century 28

Combinatorial Optimization

Lecture Hall A, Tue 2, 10:30–12:30

J. Maßberg, T. Nieberg

Colored Independent Sets 35

V. Lozin

A Note on the Parameterized Complexity of the Maximum Independent Set Problem 40

G. Nicosia, A. Pacifici, U. Pferschy

On Multi-Agent Knapsack Problems 44

L. Simonetti, Y. Frota, C. Souza

An Exact Method for the Minimum Caterpillar Spanning Problem 48

Coloring I

Lecture Hall B, Tue 2, 10:30–12:30

R. Machado, C. de Figueiredo

NP-Completeness of Determining the Total Chromatic Number of Graphs that do not Contain a Cycle with a Unique Chord 55

R. Kang, T. Müller

Acyclic and Frugal Colourings of Graphs 60

P. Petrosyan, H. Sargsyan

On Resistance of Graphs 64

E. Bampas, A. Pagourtzis, G. Pierrakos, V. Syrgkanis

Colored Resource Allocation Games 68

Cutting and Packing

Lecture Hall A, Tue 2, 14:00–15:30

C. Arbib, F. Marinelli, C. Scoppola

A Lower Bound for the Cutting Stock Problem with a Limited Number of Open Stacks 75

H. Fernau, D. Raible

Packing Paths: Recycling Saves Time 79

J. Schneider, J. Maßberg

Rectangle Packing with Additional Restrictions 84

Paths

Lecture Hall B, Tue 2, 14:00–15:30

F. Usberti, P. França, A. França

The Open Capacitated Arc Routing Problem 89

M. Maischberger

Optimising Node Coordinates for the Shortest Path Problem 93

R. Bhandari

The Sliding Shortest Path Algorithms 97

Quadratic Programming

Lecture Hall A, Tue 2, 15:45–16:45

W. Ben-Ameur, J. Neto

A Polynomial-Time Recursive Algorithm for some Unconstrained Quadratic Optimization Problems 105

<i>I. Schuele, H. Ewe, K.-H. Kuefer</i> Finding Tight RLT Formulations for Quadratic Semi-Assignment Problems	109
Trees <i>Lecture Hall B, Tue 2, 15:45–16:45</i>	
<i>V. Borozan, R. Muthu, Y. Manoussakis, C. Martinhon, A. Abouelaoualim, R. Saad</i> Colored Trees in Edge-Colored Graphs	115
<i>K. Cameron, J. Fawcett</i> Intermediate Trees	120
Plenary Session I <i>Lecture Hall A, Tue 2, 16:45–17:30</i>	
<i>A. Lodi</i> Bilevel Programming and Maximally Violated Valid Inequalities	125
Integer Programming <i>Lecture Hall A, Wed 3, 08:45–10:15</i>	
<i>R. Schultz</i> Decomposition Methods for Stochastic Integer Programs with Dominance Constraints	137
<i>S. Kosuch, A. Lisser</i> On a Two-Stage Stochastic Knapsack Problem with Probabilistic Constraint	140
<i>G. Cornuéjols, L. Liberti, G. Nannicini</i> Improved Strategies for Branching on General Disjunctions	144
Graph Theory II <i>Lecture Hall B, Wed 3, 08:45–10:15</i>	
<i>G. Katona, I. Horváth</i> Extremal Stable Graphs	149
<i>V.A. Leoni, M.P. Dobson, Gr. Nasini</i> Recognizing Edge-Perfect Graphs: some Polynomial Instances	153
<i>M. Freitas, N. Abreu, R. Del-Vecchio</i> Some Infinite Families of Q -Integral Graphs	157
Applications <i>Lecture Hall A, Wed 3, 10:30–12:30</i>	
<i>A. Ceselli, R. Cordone, M. Cremonini</i> Balanced Clustering for Efficient Detection of Scientific Plagiarism	163

<i>V. Cacchiani, A. Caprara, M. Fischetti</i> Robustness in Train Timetabling	171
<i>F. Roda, L. Liberti, F. Raimondi</i> Combinatorial Optimization Based Recommender Systems	175
<i>G. Righini, R. Cordone, F. Ficarelli</i> Bounds and Solutions for Strategic, Tactical and Operational Ambulance Location	180
Coloring II <i>Lecture Hall B, Wed 3, 10:30–12:30</i>	
<i>E. Hoshino, C. de Souza, Y. Frota</i> A Branch-and-Price Approach for the Partition Coloring Problem	187
<i>M. Soto, A. Rossi, M. Sevaux</i> Two Upper Bounds on the Chromatic Number	191
<i>F. Bonomo, G.A. Durán, J. Marenco, M. Valencia-Pabon</i> Minimum Sum Set Coloring on some Subclasses of Block Graphs	195
<i>A. Lyons</i> Acyclic and Star Colorings of Joins of Graphs and an Algorithm for Cographs	199
Exact Algorithms <i>Lecture Hall A, Wed 3, 14:00–15:30</i>	
<i>H. Fernau, S. Gaspers, D. Kratsch, M. Liedloff, D. Raible</i> Exact Exponential-Time Algorithms for Finding Bicliques in a Graph	205
<i>L. Bianco, M. Caramia</i> An Exact Algorithm to Minimize the Makespan in Project Scheduling with Scarce Resources and Feeding Precedence Relations	210
<i>R. Macedo, C. Alves, V. de Carvalho</i> Exact Algorithms for Vehicle Routing Problems with Different Service Constraints	215
Networks I <i>Lecture Hall B, Wed 3, 14:00–15:30</i>	
<i>D. Lozovanu, S. Pickl</i> Algorithmic Solutions of Discrete Control Problems on Stochastic Networks	221
<i>L. Belgacem, I. Charon, O. Hudry</i> Routing and Wavelength Assignment in Optical Networks by Independent Sets in Conflict Graphs	225
<i>A. Guedes, L. Markenzon, L. Faria</i> Recognition of Reducible Flow Hypergraphs	229

Complexity

Lecture Hall A, Wed 3, 15:45–16:45

A. Scozzari, F. Tardella

On the Complexity of Graph-Based Bounds for the Probability Bounding Problem 235

B. Engels, S. Krumke, R. Schrader, C. Zeck

Integer Flow with Multipliers: The Special Case of Multipliers 1 and 2 239

Polyhedra

Lecture Hall B, Wed 3, 15:45–16:45

R. Stephan

Classification of 0/1-Facets of the Hop Constrained Path Polytope
Defined on an Acyclic Digraph 247

A. Galluccio, C. Gentile, M. Macina, P. Ventura

The k -Gear Composition and the Stable Set Polytope 251

Plenary Session II

Lecture Hall A, Wed 3, 16:45–17:30

M. Habib

Diameter and Center Computations in Networks 257

Polynomial-time Algorithms

Lecture Hall A, Thu 4, 08:45–10:15

M. Bodirsky, G. Nordh, T. von Oertzen

Integer Programming with 2-Variable Equations and 1-Variable Inequalities 261

D. Müller

Faster Min-Max Resource Sharing and Applications 265

A. Bettinelli, L. Liberti, F. Raimondi, D. Savourey

The Anonymous Subgraph Problem 269

Graph Theory III

Lecture Hall B, Thu 4, 08:45–10:15

A. Rafael, F.-T. Desamparados, J.A. Vilches

The Number of Excellent Discrete Morse Functions on Graphs 277

F. Bonomo, G.A. Durán, L.N. Grippo, M.D. Safe

Partial Characterizations of Circle Graphs 281

T. Shigezumi, Y. Uno, O. Watanabe

A Replacement Model for a Scale-Free Property of Cliques 285

Graph Theory IV

Lecture Hall A, Thu 4, 10:30–12:30

- A. Darmann, U. Pferschy, J. Schauer, G. Woeginger*
Combinatorial Optimization Problems with Conflict Graphs 293
- J.M. Sigarreta, I. González Yero, S. Bermudo, J.A. Rodríguez-Velázquez*
On the Decomposition of Graphs into Offensive k -Alliances 297
- M. Brinkmeier*
Increasing the Edge Connectivity by One in $\mathcal{O}(\lambda_G n^2 \log^{(*)} n)$ Expected Time 301
- V. Giakoumakis, O. El Mounir*
Enumerating all Finite Sets of Minimal Prime Extensions of Graphs 305

Networks II

Lecture Hall B, Thu 4, 10:30–12:30

- C. Bentz*
On Planar and Directed Multicuts with few Source-Sink Pairs 313
- F. Usberti, J. González, C.L. Filho, C. Cavellucci*
Maintenance Resources Allocation on Power Distribution Networks with a Multi-Objective Framework 317
- E. Grande, P.B. Mirchandani, A. Pacifici*
Column Generation for the Multicommodity Min-cost Flow Over Time Problem 321
- F. Tarissan, C. La Rota*
Inferring Update Sequences in Boolean Gene Regulatory Networks 325

Bioinformatics

Lecture Hall A, Thu 4, 14:00–15:30

- N. Van Cleemput, G. Brinkmann*
NANOCONES – A Classification Result in Chemistry 333
- A. Mucherino, C. Lavor, N. Maculan*
The Molecular Distance Geometry Problem Applied to Protein Conformations 337
- G. Collet, R. Andonov, N. Yanev, J.-F. Gibrat*
Protein Threading 341

Clustering

Lecture Hall B, Thu 4, 14:00–15:30

- J. Correa, N. Megow, R. Raman, K. Suchan*
Cardinality Constrained Graph Partitioning into Cliques with Submodular Costs 347

<i>F. Liers, G. Pardella</i>	
A Simple MAX-CUT Algorithm for Planar Graphs	351
<i>E. Amaldi, S. Coniglio, K. Dhyani</i>	
k -Hyperplane Clustering Problem: Column Generation and a Metaheuristic	355
Games	
<i>Lecture Hall A, Thu 4, 15:45–16:15</i>	
<i>U. Faigle, J. Voss</i>	
A System-Theoretic Model for Cooperation and Allocation Mechanisms	361
Matrices	
<i>Lecture Hall B, Thu 4, 15:45–16:15</i>	
<i>L.A. Vinh</i>	
Distribution of Permanent of Matrices with Restricted Entries over Finite Fields	367
Plenary Session III	
<i>Lecture Hall A, Thu 4, 16:45–17:30</i>	
<i>J. Lee</i>	
On the Boundary of Tractability for Nonlinear Discrete Optimization	373
<i>List of Authors</i>	385
<i>List of Sessions</i>	388
<i>List of Timeslots</i>	389
<i>Keywords</i>	390

Traveling Salesman Problem

Lecture Hall A

Tue 2, 08:45–10:15

Effective Heuristics for Large Euclidean TSP Instances Based on Pseudo Backbones¹

C. Dong, C. Ernst, G. Jäger, D. Richter, P. Molitor

Computer Science Institute, University Halle, D-06120 Halle, Germany

{dong, ernstc, jaegerg, richterd, molitor}@informatik.uni-halle.de

Abstract

We present two approaches for the *Euclidean TSP* which compute high quality tours for large instances. Both approaches are based on pseudo backbones consisting of all common edges of good tours. The first approach starts with some pre-computed good tours. Using this approach we found record tours for seven VLSI instances. The second approach is window based and constructs from scratch very good tours of huge TSP instances, e. g., the World TSP.

Key words: Euclidean Traveling Salesman Problem, Pseudo Backbone, Problem Contraction, Iterative Approach, Window Based Approach

1. The overall approach

Given a set of cities and the distances between each pair of them, the Traveling Salesman Problem (TSP) is the \mathcal{NP} -hard problem of finding a shortest cycle visiting each city exactly once. In this paper we consider *Euclidean TSP* whose cities are embedded either in the Euclidean plane using the Euclidean distance or a ball using the spherical grid of latitude and longitude. The *backbone* of a TSP instance consists of all edges, which are contained in *each* optimum tour of the instance, and is an important criterion for the hardness of a TSP instance. The larger the backbone of an instance, the simpler is the remaining sub-instance. Unfortunately it is usually hard to compute the backbone of an instance. An interesting observation is that tours of an instance with good quality are likely to share many edges. We can presume that these edges are also contained in optimum tours and call them *pseudo backbone edges*. This basic observation is elaborated in detail in our approach. Assume that for a given TSP instance a set of pseudo backbone edges is computed.

¹ This work is supported by German Research Foundation (DFG) with grant number MO 645/7-3.

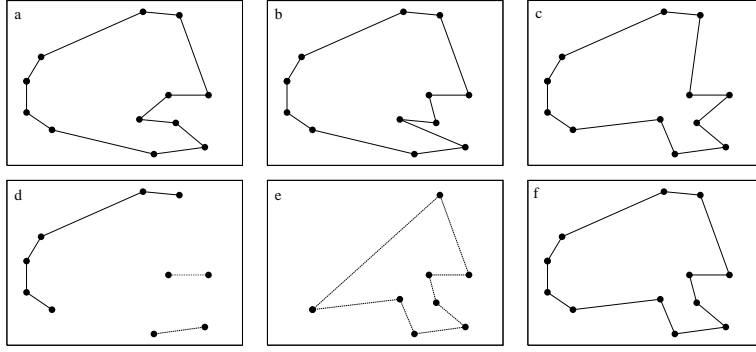


Fig. 1. Illustration of the first approach. The instance has 12 points in the Euclidean plane. By the three starting tours given in (a), (b), and (c), we receive the pseudo backbone edges (d). From the maximal paths consisting only of pseudo backbone edges, only one has a length greater than 1. Only this path contributes to the size reduction. After contracting, we receive a new instance with 8 points which contains 3 p -edges (e). The three p -edges are fixed while searching tours for the new instance. In (e) an optimal tour t' for the new instance is shown. After re-contracting the p -edges by the corresponding paths, we receive a tour t for the original instance (f). For this instance, the final tour is optimal.

Our idea is to contract maximal paths of pseudo backbone edges to single edges which are kept fixed during the following process. By the contraction step, a new TSP instance with smaller size is created which can be attacked more effectively.

2. Using good starting tours for pseudo backbone computation

Let a TSP instance be given as a complete graph $G = (V, E)$ with $E = V \times V$. Our first approach undergoes the following five steps (see Fig. 1). The first step is to find a set Ω of good tours for G which are called *starting tours*. The second step is to collect the pseudo backbone edges, i. e., compute the set $B := \{e \in E; e \in \cap_{T \in \Omega} T\}$ of edges which are contained in each tour of Ω . Let V_B be the set of vertices which are endpoints of at least one edge of B . The third step is to construct all maximal paths consisting only of edges in B and contract each of these maximal paths to an edge, the endpoints of which are that of the path. We denote them by p -edges (path edges) and the set of all end points of the p -edges by V_p . The contraction step results in a new TSP instance $H = (W, F)$ with $W = (V \setminus V_B) \cup V_p$, $F = W \times W$, where the weight of the p -edges can be chosen arbitrarily. The fourth step is to find a good tour t' for the new TSP instance H subject to the condition that all p -edges must be in the tour. Finally, the fifth step is to obtain a tour t for the original TSP instance G by re-contracting the p -edges by the corresponding paths in the computed tour t' . The experimental results strongly demonstrate the effectivity of the approach: for seven VLSI instances with sizes 13584, 17845, 19402, 21215, 28924, 47608 and 52057 we could find better tours than the best tours known so far (see TSP homepage: <http://www.tsp.gatech.edu/>). The success of this approach

strongly depends on having good starting tours generated by different methods – for the above mentioned results we used starting tours which had been constructed by different tolerance based algorithms presented in [3] (see [1] for more information on tolerances).

3. Iterative window based pseudo backbone computation

Our second approach computes tours of large Euclidean TSP instances from scratch, i. e., it does not require starting tours. In fact, computing multiple different good starting tours for the `World TSP` with 1,904,711 cities is hardly realizable in reasonable time. The basic idea of our window based approach consists of splitting the bounding box of the vertices of the TSP instance in non-disjoint windows by moving a window frame across the bounding box of the vertices of the TSP instance with a step size of half the width (height) of the window frame (see Fig. 2). Thus each vertex is contained in up to four windows. Each window defines a sub-instance for which a good tour is computed, e. g., by Helsgaun’s LKH [2], independently of the neighboring sub-instances. Now, the approach is based on the assumption that an edge (u, w) , which is contained in the same four windows and in each of the four tours, has high probability to appear in an optimal tour of the original TSP instance – in some sense the four windows together reflect the surrounding area of (u, w) with respect to the four directions. Such edges are declared as pseudo backbone edges (see Fig. 3(a)-3(c)). After the contraction of the maximum paths of pseudo backbone edges, the approach is iterated with monotonically increasing window frame. We applied the described algorithm to the `World TSP` and required about 4.75 days for computing from scratch a tour of length 7,569,766,108 which is only at most 0.7661% greater than the length of an optimum tour. Currently, our approach is still dominated in some sense by LKH. By assigning the right values to the parameters, LKH computes a tour for the `World TSP` in less than two days which is at most 0.1174% greater than the length of an optimum tour [4].[†] However, note that till now we have used only default parameters for LKH without any parameter tuning. By detailed parameter tuning – as done by Helsgaun – the window frames of our approach can be chosen much larger which should lead to an improvement of the computed tours and running times.

References

- [1] B. Goldengorin, G. Jäger, and P. Molitor. Tolerances applied in combinatorial optimization. *J. Comput. Sci.* 2(9), 716-734, Science Publications, 2006.

[†] Note that the computation times, Helsgaun states in [2], do not include the computation times of the starting tours [4].

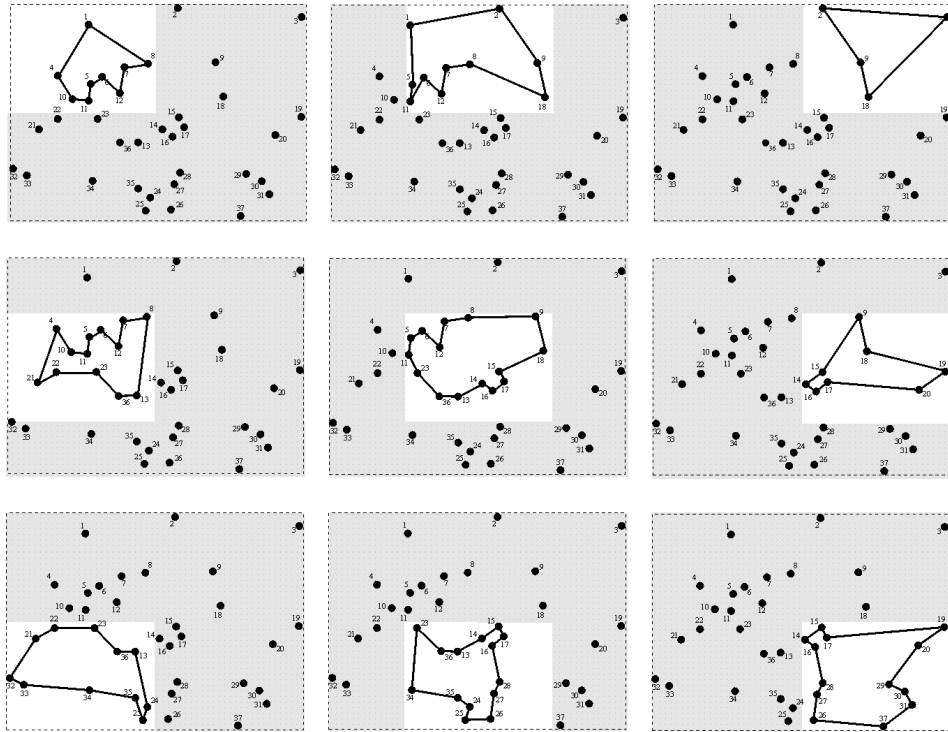


Fig. 2. Illustration of the window based technique of splitting large TSP instances into sub-instances.

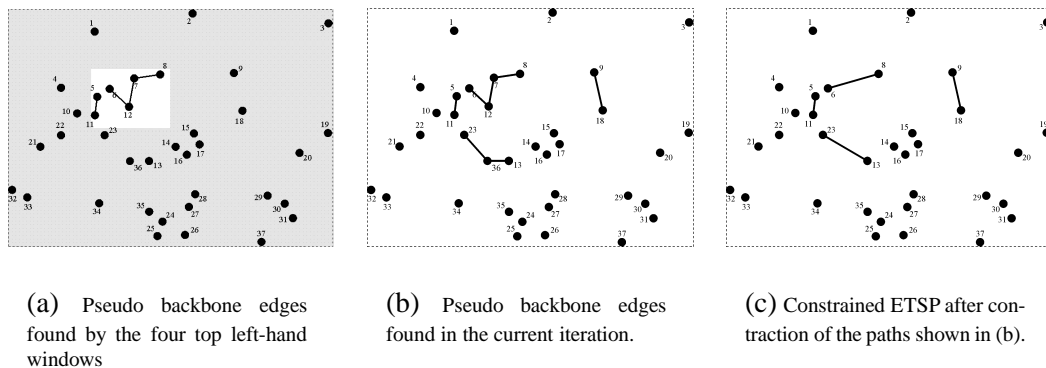


Fig. 3. Window based pseudo backbone computation and contraction.

- [2] K. Helsgaun. An Effective Implementation of K-opt Moves for the Lin-Kernighan TSP heuristic. *Writings on Computer Science* 109, Roskilde University, 2007.
- [3] D. Richter, B. Goldengorin, G. Jäger, and P. Molitor. Improving the Efficiency of Helsgaun’s Lin-Kernighan Heuristic for the Symmetric TSP. *Proc. of the 4th Workshop on Combinatorial and Algorithmic Aspects of Networking (CAAN)*, Lecture Notes in Comput. Sci. 4852, 99-111, 2007.
- [4] K. Helsgaun. Private Communication, March 2009.

Efficient algorithms for the Double Traveling Salesman Problem with Multiple Stacks

Marco Casazza,^a Alberto Ceselli,^a Marc Nunkesser^b

^a*Dept. of Information Technologies - University of Milan*
{marco.casazza, alberto.ceselli}@unimi.it

^b*Inst. of Theoretical Computer Science - ETH Zürich*
mnunkess@inf.ethz.ch

Key words: Traveling Salesman Problem, LIFO constraints, efficient algorithms

1. Introduction

Routing is a key issue in logistics, and has been deeply studied in the literature; however, several practical applications require the loading of vehicles to be explicitly considered. The Double Traveling Salesman Problem with Multiple Stacks (DTSPMS) is one of the simplest examples of integrated routing and loading problem: two cities are given, in which N customers are placed. Items have to be collected from the customers through a tour in the first city, and then delivered through a tour in the second city. During the pickup tour, the items have to be organized in *stacks* on the back of the vehicle; the delivery operations can start only from the top of the stacks. The DTSPMS is NP-Hard, as it includes the TSP as a special case. Both heuristics [1] [2] and exact methods [3] have been proposed to solve it.

The main aim of this paper is to investigate on theoretical properties of the DTSPMS; we also propose and test an efficient heuristic algorithm which exploits such properties.

2. Formulation and properties

The DTSPMS can be modeled as the following graph optimization problem. We are given a set of customers numbered $1 \dots N$ and two (di-)graphs $\mathcal{G}^+(\mathbb{N}^+, \mathcal{A}^+)$ and $\mathcal{G}^-(\mathbb{N}^-, \mathcal{A}^-)$ with weights c^+ and c^- respectively on the arcs. The former is the *pickup graph* and latter is the *delivery graph*. Both sets \mathbb{N}^+ and \mathbb{N}^- consist of one vertex n_i^+ and n_i^- for each customer i , and an additional vertex 0 which represents a depot. Hence the number of vertices is the same in the two graphs. Each customer i requires the pickup of an item in vertex n_i^+ and the delivery of the same item in vertex n_i^- .

We indicate as *pickup tour* (resp. *delivery tour*) any permutation of vertices of the

pickup graph (resp. delivery graph). Each tour starts from and ends at the depot. Each tour has a cost, which is the cost for traveling from one vertex to the next, according to the order indicated by the permutation. Given two customers i and j , we say that i precedes j on the pickup tour if n_i^+ appears to the left of n_j^+ in the corresponding permutation. In a similar way, i precedes j on the delivery tour if n_i^- appears to the left of n_j^- in the corresponding permutation.

The vehicle has a given number S of *stacks* available for transportation. A *loading plan* is a mapping l from each customer i to a pair (s, p) , representing the arrangement of the items in the stacks of the vehicle. In particular, $l(i) = (s, p)$ if the item of customer i occupies position p on stack s , with $(s, 1)$ representing the bottom of stack s .

Each stack actually represents a Last-In-First-Out structure: a loading plan is *feasible* with respect to a pickup tour (and vice versa) if, given any pair of customers i and j such that i precedes j in the pickup tour, either $l(i) = (s, p)$ and $l(j) = (t, q)$ with $s \neq t$, or $p < q$. That is, if item i is picked up before item j , i cannot be placed on top of j in the same stack. A similar definition holds for the delivery tour. If i precedes j in both the pickup and the delivery tour, it must be $l(i) = (s, p)$ and $l(j) = (t, q)$ with $s \neq t$, and we say that customers i and j are *incompatible*. Hence, a solution of the DTSPMS is composed by two ingredients: a pair of pickup and delivery tours and a loading plan; such solution is feasible if the loading plan is feasible with respect to both tours.

In the following we show that, given one of the two ingredients of a feasible solution, the remaining one can be found in polynomial time. This holds in particular for an optimal solution. We present only a sketch of the proofs.

Problem (1): Given a pickup tour and a delivery tour, find a feasible loading plan using the minimum number of stacks.

Proposition 1. Problem (1) can be solved in polynomial time.

We define a *conflict graph* C having one vertex for each customer, and one edge for each pair of incompatible customers. Problem (1) can be re-stated as the problem of coloring graph C with the minimum number of colors: different colors represents different stacks; since no adjacent vertices can take the same color in a feasible coloring, no incompatible customers can be assigned to the same stack. The order of the items inside each stack can be chosen according to their order in one of the tours. We show that C is a *permutation graph*, which is a special case of perfect graph. In these graphs coloring problems can be solved in polynomial time by means of flow computations [4]. As far as efficiency is concerned, we show that Problem (1) can be solved in $O(N \cdot \log N)$ time by an adaptation of the algorithm presented in [4].

Problem (2): Given a loading plan, find a delivery tour which is feasible with respect to the loading plan and has minimum cost.

Problem (3): Given a loading plan, find a pickup tour which is feasible with respect to the loading plan and has minimum cost.

Proposition 2. Problem (2) and Problem (3) can be solved in polynomial time.

In fact, once a feasible loading plan is given, suppose to incrementally build partial delivery tours by choosing items on the top of the stacks. Let $f(s_1, \dots, s_S, p)$ be the minimum cost of a partial tour in which s_1 items are left in stack 1, s_2 items are left in stack 2 and so on, and in which the item on the top of stack p is the next to be delivered. Let i be the customer corresponding to the item on top of stack p ; if i is the first customer to be visited, then $f(s_1, \dots, s_S, p) = c_{0,i}^-$; otherwise, consider any stack q , which has on top item j : $f(s_1, \dots, s_S, p) = \min_{q=1..S} \{f(s_1, \dots, s_q + 1, \dots, s_S, q) + c_{j,i}^-\}$. An optimal solution can be found in $O(|N|^{S+1})$ time by computing all the values for $f()$ using dynamic programming recursion. Informally, the computation can be repeated to solve Problem (3) by considering the items of each stack in reverse order.

However, given two random tours, it might not be possible to find a loading plan using at most S stacks. Therefore we define as *partial loading plan* a loading plan in which the items of a subset of customers do not appear, and we consider the following:

Problem (5): Given a pickup and a delivery tour, find a feasible partial loading plan using at most S stacks, including the maximum number of items.

Proposition 3. Problem (5) can be solved in polynomial time.

We build a graph having a vertex for each customer and two vertices for the depot (start and end), an arc between the vertex of each customer and the vertices of its compatible customers, between the start depot vertex and each customer vertex, and between each customer vertex and the end depot vertex. The start and end depot vertices are respectively a source and a sink of S units of flow. We assign capacity 1 and cost 0 to each arc, and cost -1 to each customer vertex. Problem (5) can be restated as the problem of finding a minimum cost flow on a suitable modification of this graph. Informally, the S units of flow define sequences of customers included in the same stack. Every time a vertex receives flow, the corresponding customer is inserted in a stack, and a value -1 is collected; therefore in an optimal solution the maximum number of customers is included.

3. Algorithms

We elaborated on the previous results to obtain a heuristic algorithm for the DTSPMS. The algorithm works in five steps: (a) find a pickup tour and a delivery tour (b) solve Problem (5), creating a feasible partial loading plan including the highest number of customers (c) solve Problem (2) and Problem (3) considering only customers in the partial loading plan, creating optimal partial pickup and delivery tours (d) create a feasible DTSPMS solution by including the remaining customers in the stacks using a best insertion policy (e) create a candidate solution for the next iteration of the algorithm by including the remaining customers in the partial tours using a best insertion policy (f) repeat steps (b) – (f).

First we note that the number of customers which are inserted in the partial loading plan, which is found in step (b), is always non decreasing from one iteration to the next. In fact, customers whose items are included in a partial loading plan during

iteration k appear in the tours according to the order given by the partial loading plan; our insertion algorithm do not change that order; hence, during iteration $k + 1$ it is always possible to rebuild the partial loading plan of iteration k . Therefore, we stop the algorithm whenever no additional customer is inserted in the partial loading plan during step (c). In order to obtain a feasible solution in step (d), we consider the items which are not included in the loading plan in a random order. We try to place every item in each possible position in the stacks, and in each compatible insertion point in the tours. Then, we place each item in the position of the loading plan giving minimum insertion cost. Instead, in order to obtain a candidate solution in step (e), we consider in a random order each customer whose item is not in the partial loading plan, and we perform a best insertion operation in both the pickup and delivery tours. We keep the best solution found in step (d) during the iterations of the main algorithm as final solution. In the literature, it is common to further constrain the problem by imposing a limit on the number of items which can be placed in the same stack. When such a constraint is imposed, during step (d) we remove from the partial loading plan each item exceeding the limit, and we care not to insert additional items in full stacks.

We implemented our heuristic algorithm in C, using MCF library for the flow sub-problems and CONCORDE to obtain tours in step (a). We considered the testbed of 10 instances involving 33 customers proposed in [1] and [3]. We run experiments on a 1.83GHz notebook *. As a benchmark, we considered the results of the HVNS metaheuristic [1], when let run for ten seconds. Our method provides in a fraction of a second solutions whose quality is about 8% worse than those given by HVNS. This highlights as a promising research direction to combine our algorithm with local search methods.

References

- [1] A. Felipe, M.T. Ortuno and G. Tirado (2008) Neighborhood structures to solve the double TSP with multiple stacks using local search. FLINS Proceedings, Madrid, September 21–24 2008
- [2] H. Petersen and O. Madsen. (2008) The double travelling salesman problem with multiple stacks - formulation and heuristic solution approaches. European Journal of Operational Research, forthcoming
- [3] H. Petersen, C. Archetti and M.G. Speranza (2008) Exact Solutions to the double TSP with multiple stacks. Tech. rep, University of Brescia
- [4] A. Brandstädt (1992) On Improved Time Bounds for Permutation Graph Problems. Lecture Notes in Computer Science 657

* A full table is available at: <http://www.dti.unimi.it/~ceselli/DTSPMS.shtml>

The Parking Warden Tour Problem

Maurizio Bruglieri,^a Alberto Colorni,^a Alessandro Lué^b

^a*INDACO, Politecnico di Milano, via Durando 38/a, 20158 Milano, Italy*
{maurizio.bruglieri, alberto.colorni}@polimi.it

^b*POLIEDRA, Politecnico di Milano, via Garofalo 39, 20133 Milano, Italy*
lue@poliedra.polimi.it

Key words: irregular parking, parking warden tour, arc routing problem.

1. Introduction

Irregular parking is a scourge for most of the Italian cities and in most of the cases enforcement ([6]) is not effective. Unfortunately, municipalities have limited resources to hire a sufficient number of parking wardens, and the tours and schedule of the existing wardens are not planned using quantitative models. For the municipality of Como (Italy), we are studying how to re-configure the parking system, considering both pricing and organizational aspects. Within this study, we developed a model to improve the level of efficiency of the parking enforcement optimizing the parking warden tours. The problem is the following. The team of parking wardens, the city road network, the link travel time, and the estimated profit deriving from the sanctions applied to the cars irregularly parked are given. We want to determine the tour of each warden (that is a cycle where both vertices and edges may be repeated and having total duration less than the warden service time), with the aim of maximizing the total profit collected. In the literature, we do not find mathematical models that face this problem. This problem differs from the Profitable Arc Tour problem [1] since in the latter both the arc profits and costs are fixed whereas in our problem they depend on the moment of the day (the average number of irregular parked cars can vary during the day) and on the time passed from the previous inspection of a warden (the profit on a link slumps to zero if a warden has just visited this link). This peculiarity occurs in other different routing problems. For instance, in the snowplough vehicles routing problem the profit collected is the amount of snow removed, which depends on the time passed from the previous transit of a snowplough, supposing that it is snowing during the operations. At the best of our knowledge ([1], [4] [5]), this is the first study of an arc routing problem where the arc profit depends also on the solution itself. For

this new arc routing problem, we present a MILP formulation from which we also develop a simple but effective heuristic approach.

2. Graph representation of the problem

Since the wardens inspect the road network by foot, the problem can be modelled by way of an undirected graph where each edge represents a road link that can be travelled in both the directions. In each road link the cars can be parked from one to four sides according to the width of the road and the presence or not of a traffic island. Each side is visited by the wardens in different moments, except the two sides of the traffic islands. Therefore, we have to duplicate the ending vertices of the original road links if they have two parking sides or triplicate the ending vertices if in addition there is a traffic island, to avoid parallel edges. The edges linking the copies of the same vertex represent the action of crossing the road to change the side and no profit is associated to them.

3. Mixed Integer Linear Programming formulation

Beside the undirected graph $G = (V, E)$ described in the previous section, we suppose also given the following data:

- W = parking warden set
- T = parking warden service time
- c_e = travel time by foot of edge e
- q = time needed to sanction one car
- p = profit for one irregularly parked car
- s_e = estimation of irregularly parked cars on edge e
- R_e = estimation of turn-over time on edge e

We assume that a team of wardens is available at a single depot, represented by vertex 0, and they have to come back to depot at the end of the service. Moreover we assume that the profit of an edge e slumps to zero when such edge is visited by a warden. Afterwards, the profit increases linearly from 0 to ps_e until the turn-over time R_e is reached, after which it remains constant until the next visit. Under these assumptions, we state that the Parking Warden Tour Problem (PWTP) can be modeled by way of the following Mixed Integer Linear Program (MILP), where K is an upper bound on the number of edges that the wardens can visit along their tour (for instance $K = \frac{T}{\min_{e \in E} c_e}$) and $\delta(0)$ denotes the edges incident to the depot.

$$\max \sum_{w \in W} \sum_{k=1}^K \pi_{kw} \quad (3.1)$$

$$\sum_{k=1}^K \sum_{e \in E} (c_e x_{ekw} + q s_e z_{ekw}) \leq T \quad \forall w \in W \quad (3.2)$$

$$\sum_{e \in \delta(0)} x_{e1w} = 1 \quad \forall w \in W \quad (3.3)$$

$$\sum_{e \in E} x_{ekw} \leq 1 \quad \forall k=2, \dots, K, \forall w \in W \quad (3.4)$$

$$v_{01w} = 1 \quad \forall w \in W \quad (3.5)$$

$$\sum_{k=2}^K v_{0kw} = 1 \quad \forall w \in W \quad (3.6)$$

$$\sum_{i \in V} v_{ikw} \leq 1 \quad \forall k=2, \dots, K+1, \forall w \in W \quad (3.7)$$

$$\sum_{i: \{i,j\} \in E} v_{ik+1w} \geq v_{jkw} \quad \forall j \in V, \forall k=1, \dots, K, \forall w \in W \quad (3.8)$$

$$\sum_{i \in N} \sum_{k'=k+1}^{K+1} v_{ik'w} \leq (K-k+1) (1-v_{0kw}) \quad \forall k=2, \dots, K, \forall w \in W \quad (3.9)$$

$$x_{ekw} \geq v_{ikw} + v_{jk+1w} - 1 \quad \forall e = \{i,j\} \in E, \forall k=1, \dots, K, \forall w \in W \quad (3.10)$$

$$x_{ekw} \leq v_{ikw} \quad \forall e = \{i,j\} \in E, \forall k=1, \dots, K, \forall w \in W \quad (3.11)$$

$$x_{ekw} \leq v_{jk+1w} \quad \forall e = \{i,j\} \in E, \forall k=1, \dots, K, \forall w \in W \quad (3.12)$$

$$z_{ekw} \leq x_{ekw} \quad \forall e = \{i,j\} \in E, \forall k=1, \dots, K, \forall w \in W \quad (3.13)$$

$$t_{1w} = 0 \quad \forall w \in W \quad (3.14)$$

$$t_{kw} \geq t_{k-1w} + \sum_{e \in E} (c_e x_{ek-1w} + q s_e z_{ek-1w}) \quad \forall k=2, \dots, K+1, \forall w \in W \quad (3.15)$$

$$\pi_{kw} \leq p \sum_{e \in E} s_e z_{ekw} \quad \forall k=1, \dots, K, \forall w \in W \quad (3.16)$$

$$\pi_{k''w} \leq p s_e \frac{t_{k''w} - t_{k'w}}{R_e} + pS(2 - z_{ek''w} - z_{ek'w} + \sum_{k=k'+1}^{k''-1} z_{ekw}) \quad \forall e \in E, \forall k', k''=1, \dots, K: k'' > k', \forall w \in W \quad (3.17)$$

$$\pi_{k''w''} \leq p s_e \frac{t_{k''w''} - t_{k'w'}}{R_e} + pS(1 + \frac{T}{R_e})(3 - z_{ek''w''} - z_{ek'w'} - y_{k'k''w'w''}) \quad \forall e \in E, \forall k', k''=1, \dots, K, \forall w', w'' \in W: w' < w'' \quad (3.18)$$

$$\pi_{k'w'} \leq p s_e \frac{t_{k'w'} - t_{k''w''}}{R_e} + pS(1 + \frac{T}{R_e})(2 - z_{ek''w''} - z_{ek'w'} - y_{k'k''w'w''}) \quad \forall e \in E, \forall k', k''=1, \dots, K, \forall w', w'' \in W: w' < w'' \quad (3.19)$$

$$y_{k'k''w'w''} \leq 3 + \frac{t_{k''w''} - t_{k'w'}}{T} - z_{ek'w'} - z_{ek''w''} \quad \forall e \in E, \forall k', k''=1, \dots, K, \forall w', w'' \in W: w' < w'' \quad (3.20)$$

$$y_{k'k''w'w''} \geq -2 + \frac{t_{k''w''} - t_{k'w'}}{T} + z_{ek'w'} + z_{ek''w''} \quad \forall e \in E, \forall k', k''=1, \dots, K, \forall w', w'' \in W: w' < w'' \quad (3.21)$$

$$x_{ekw} \geq 0 \quad \forall e \in E, \forall k=1, \dots, K, \forall w \in W \quad (3.22)$$

$$y_{k'k''w'w''} \in \{0, 1\} \quad \forall k', k''=1, \dots, K, \forall w', w'' \in W: w' < w'' \quad (3.23)$$

$$v_{ikw} \in \{0, 1\} \quad \forall i \in V, \forall k=1, \dots, K+1, \forall w \in W \quad (3.24)$$

$$z_{ekw} \in \{0, 1\} \quad \forall e \in E, \forall k=1, \dots, K, \forall w \in W \quad (3.25)$$

$$\pi_{kw} \geq 0 \quad \forall k=1, \dots, K, \forall w \in W \quad (3.26)$$

$$0 \leq t_{kw} \leq T \quad \forall k=1, \dots, K+1, \forall w \in W \quad (3.27)$$

Variables π_{kw} model the profit collected by warden w when visiting the k -th edge of his/her tour: these variables are settled by constraints (3.16) and by *big-M* constraints (3.17), (3.18) and (3.19), where $S = \max_{e \in E} s_e$. Therefore the objective function (3.1) models the maximization of the total collected profit.

Variables v_{ikw} are equal to 1 if vertex i is the k -th vertex visited by warden w , 0

otherwise. Thanks to constraints (3.10), (3.11) and (3.12), variables x_{ekw} are binary although not directly constrained to be so: in particular they are equal to 1 if edge e is the k -th edge visited by warden w in his/her tour (independently on the fact that its profit is collected or not), are equal to 0 otherwise. Indeed (3.10), (3.11) and (3.12) can be seen as McCormick linearization constraints ([3]) imposing that variables x_{ekw} have the same behaviour of bilinear terms $v_{ikw}v_{jk+1w}$ where e is the edge linking vertices i and j .

Variables z_{ekw} are equal to 1 if edge e is the k -th edge visited by warden w in his/her tour and its profit is collected, are equal to 0 otherwise.

Variables t_{kw} model the time instant when the k -th edge is visited by warden w ; variables $y_{k'k''w'w''}$ model the precedence relationships in the visit of the same edge by different wardens: in particular when the k' -th edge travelled by warden w' and the k'' -th edge travelled by warden w'' coincide, such variables are equal to 1 if w'' precedes w' , are equal to 0 otherwise.

4. Some computational results

We notice that MILP (1)-(27) involves $O(|E|K|W| + K^2|W|^2)$ binary variables and $O(|E|K^2|W|^2)$ linear constraints, therefore in practice we cannot think of applying this model directly to the whole team of wardens, unless to consider just small instances. Anyway from the MILP model we can build a simple but effective heuristic approach. It consists in iteratively solving, with the MILP (1)-(27), $|W|$ instances of PWTP with one warden, where, at each iteration, the profit collection of the edges already visited with profit in the previous iterations, is forbidden. We have implemented the MILP (1)-(27) in AMPL [2] and considered random instances with number of vertices between 10 and 50, number of edges between 30 and 150 and up to 4 wardens. The preliminary computational results obtained with CPLEX11.0 solver show that the heuristic is able to find a solution always in few seconds, whereas the MILP can require hundreds of seconds up to 2 wardens and also several hours for 4 wardens. Concerning the solution quality, we have found an average percentage gap between the heuristic and the optimal solution of about 5.8%.

References

- [1] D. Feillet, P.Dejax, M.Gendreau. The Profitable Arc Tour Problem: Solution with a Branch-and-Price Algorithm. *Transportation Science*, 39(4):539–552, 2005.
- [2] R. Fourer and D. Gay The AMPL Book. Duxbury Press, Pacific Grove, 2002.
- [3] G.P. McCormick. Computability of global solutions to factorable nonconvex

- programs: part I-convex underestimating problems. *Mathematical Programming*, 10:146–75, 1976.
- [4] N. Perrier, A. Langevin, J.F. Campbell. A survey of models and algorithms for winter road maintenance. Part IV: Vehicle routing and fleet sizing for plowing and snow disposal. *Computers & Operations Research*, 34:258–294, 2007.
 - [5] N. Perrier, A. Langevin, A. Amaya. Vehicle routing for urban snow plowing operations. *Les Cahiers du GERAD*, G-2006-33, 2006.
 - [6] R. Petiot Parking enforcement and travel demand management. *Transport Policy*, 11:399–411, 2004.

Graph Theory I

Lecture Hall B

Tue 2, 08:45–10:15

On Self-Duality of Branchwidth in Graphs of Bounded Genus[★]

Ignasi Sau,^a Dimitrios M. Thilikos^b

^a*Mascotte project INRIA/CNRS/UNSA, Sophia-Antipolis, France; and Graph Theory and Comb. Group, Applied Maths. IV Dept. of UPC, Barcelona, Spain.*

^b*Department of Mathematics, National Kapodistrian University of Athens, Greece.*

Abstract

A graph parameter is *self-dual* in some class of graphs embeddable in some surface if its value does not change in the dual graph more than a constant factor. Self-duality has been examined for several width-parameters, such as branchwidth, pathwidth, and treewidth. In this paper, we give a direct proof of the self-duality of branchwidth in graphs embedded in some surface. In this direction, we prove that $\mathbf{bw}(G^*) \leq 6 \cdot \mathbf{bw}(G) + 2g - 4$ for any graph G embedded in a surface of Euler genus g .

Key words: graphs on surfaces, branchwidth, duality, polyhedral embedding.

1. Preliminaries

A *surface* is a connected compact 2-manifold without boundaries. A surface Σ can be obtained, up to homeomorphism, by adding $\mathbf{eg}(\Sigma)$ *crosscaps* to the sphere. $\mathbf{eg}(\Sigma)$ is called the *Euler genus* of Σ . We denote by (G, Σ) a graph G embedded in a surface Σ . A subset of Σ meeting the drawing only at vertices of G is called *G -normal*. If an O -arc is G -normal, then we call it a *noose*. The *length* of a noose is the number of its vertices. *Representativity*, or *face-width*, is a parameter that quantifies local planarity and density of embeddings. The representativity $\mathbf{rep}(G, \Sigma)$ of a graph embedding (G, Σ) is the smallest length of a non-contractible noose in Σ . We call an embedding (G, Σ) *polyhedral* if G is 3-connected and $\mathbf{rep}(G, \Sigma) \geq 3$. See [7] for more details. For a given embedding (G, Σ) , we denote by (G^*, Σ) its dual embedding. Thus G^* is the geometric dual of G . Each vertex v (resp. face r) in (G, Σ) corresponds to some face v^* (resp. vertex r^*) in (G^*, Σ) . Also, given a set $X \subseteq E(G)$, we denote as X^* the set of the duals of the edges in X .

[★] This work has been supported by IST FET AEOLUS, COST 295-DYNAMO, and by the Project “Kapodistrias” (AΠ 02839/28.07.2008) of the National and Kapodistrian University of Athens (project code: 70/4/8757).

Given a graph G and a set $X \subseteq E(G)$, we define $\partial X = (\bigcup_{e \in X} e) \cap (\bigcup_{e \in E(G) \setminus X} e)$ (notice that $\partial X = \partial(E(G) \setminus X)$). A *branch decomposition* (T, μ) of a graph G consists of an unrooted ternary tree T (i.e., all internal vertices are of degree three) and a bijection $\mu : L \rightarrow E(G)$ from the set L of leaves of T to the edge set of G . For every edge $f = \{t_1, t_2\}$ of T we define the *middle set* $\mathbf{mid}(e) \subseteq V(G)$ as follows: Let L_1 be the leaves of the connected component of $T \setminus \{e\}$ that contain t_1 . Then $\mathbf{mid}(e) = \partial\mu(L_1)$. The *width* of (T, μ) is defined as $\max\{|\mathbf{mid}(e)| : e \in T\}$. An optimal branch decomposition of G is defined by a tree T and a bijection μ which give the minimum width, called the *branchwidth* of G , and denoted by $\mathbf{bw}(G)$.

Suppose G_1 and G_2 are graphs with disjoint vertex-sets and $k \geq 0$ is an integer. For $i = 1, 2$, let $W_i \subseteq V(G_i)$ form a clique of size k and let G'_i ($i = 1, 2$) be obtained from G_i by deleting some (possibly none) of the edges from $G_i[W_i]$ with both endpoints in W_i . Consider a bijection $h : W_1 \rightarrow W_2$. We define a *clique-sum* $G_1 \oplus G_2$ of G_1 and G_2 to be the graph obtained from the union of G'_1 and G'_2 by identifying w with $h(w)$ for all $w \in W_1$.

Let \mathcal{G} be a class of graphs embeddable in a surface Σ . We say that a graph parameter *CRRAP* is (c, d) -*self-dual* on \mathcal{G} if for every graph $G \in \mathcal{G}$ and for its geometric dual G^* , $\text{CRRAP}(G^*) \leq c \cdot \text{CRRAP}(G) + d$. Results concerning self-duality of pathwidth can be found in [4; 1]. Branchwidth is $(1, 0)$ -self-dual in planar graphs that are not forests [9], while analogous results have been proven for other parameters such as pathwidth [3; 1] and treewidth [5; 2; 6]. In this note, we give a proof that branchwidth is $(6, 2g - 4)$ -self-dual in graphs of Euler genus at most g . We also believe that our result can be considerably improved. In particular, we conjecture that branchwidth is $(1, g)$ -self-dual.

2. Self-duality of branchwidth

If (G, Σ) is a polyhedral embedding, then the following proposition follows by an easy modification of the proof of [4, Theorem 1].

Proposition 2.1. Let (G, Σ) and (G^*, Σ) be dual polyhedral embeddings in a surface of Euler genus g . Then $\mathbf{bw}(G^*) \leq 6 \cdot \mathbf{bw}(G) + 2g - 4$.

In the sequel, we focus on generalizing Proposition 2.1 to arbitrary embeddings. For this we first need some technical lemmata, whose proofs are easy or well known, and omitted in this extended abstract. Note that the removal of a vertex in G corresponds to the contraction of a face in G^* , and viceversa.

Lemma 2.2. The removal of a vertex or the contraction of a face from an embedded graph decreases its branchwidth by at most 1.

Lemma 2.3. (Fomin and Thilikos [3]) Let G_1 and G_2 be graphs with one edge or one vertex in common. Then $\mathbf{bw}(G_1 \cup G_2) \leq \max\{\mathbf{bw}(G_1), \mathbf{bw}(G_2), 2\}$.

Theorem 2.4. Let (G, Σ) be an embedding with $g = \text{eg}(\Sigma)$. Then $\mathbf{bw}(G^*) \leq$

$$6 \cdot \text{bw}(G) + 2g - 4.$$

Proof. The proof uses the following procedure that applies a series of cutting operations to decompose G into polyhedral pieces plus a set of vertices whose size is linearly bounded by $\text{eg}(\Sigma)$. The input is the graph G and its dual G^* embedded in Σ .

1. Set $\mathcal{B} = \{G\}$, and $\mathcal{B}^* = \{G^*\}$ (we call the members of \mathcal{B} and \mathcal{B}^* *blocks*).
2. If (G, Σ) has a minimal separator S with $|S| \leq 2$, let C_1, \dots, C_ρ be the connected components of $G[V(G) \setminus S]$ and, for $i = 1, \dots, \rho$, let G_i be the graph obtained by $G[V(C_i) \cup S]$ by adding an edge with both endpoints in S in the case where $|S| = 2$ and such an edge does not already exist (we refer to this operation as *cutting* G along the separator S). Notice that a (non-empty) separator S of size at most 2 corresponds to a non-empty separator S^* of G^* , and let $G_i^*, i = 1, \dots, \rho$ be the graphs obtained by cutting G^* along S^* . We say that each G_i (resp G_i^*) is a *block* of G (resp. G^*) and notice that each G and G^* is the clique sum of its blocks. Therefore, from Lemma 2.3, $\text{bw}(G^*) \leq \max\{2, \max\{\text{bw}(G_i^*) \mid i = 1, \dots, \rho\}\}$ **(1)**. Observe that we may assume that for each $i = 1, \dots, \rho$, G_i and G_i^* are embedded in a surface Σ_i such that G_i is the dual of G_i^* and $\text{eg}(\Sigma) = \sum_{i=1, \dots, \rho} \text{eg}(\Sigma_i)$. Notice that $\text{bw}(G_i) \leq \text{bw}(G), i = 1, \dots, \rho$ **(2)**, as the possible edge addition does not increase the branchwidth, since each block of G is a minor of G . We set $\mathcal{B} \leftarrow \mathcal{B} \setminus \{G\} \cup \{G_1, \dots, G_\rho\}$ and $\mathcal{B}^* \leftarrow \mathcal{B}^* \setminus \{G^*\} \cup \{G_1^*, \dots, G_\rho^*\}$.
3. If (G, Σ) has a non-contractible and non-surface-separating noose meeting a set S with $|S| \leq 2$, let $G' = G[V(G) \setminus S]$ and let F be the set of faces in G^* corresponding to the vertices in S . Observe that the obtained graph G' has an embedding to some surface Σ' of Euler genus *strictly* smaller than Σ that, in turn, has some dual G'^* in Σ' . Therefore $\text{eg}(\Sigma') < \text{eg}(\Sigma)$. Moreover, G'^* is the result of the contraction in G^* of the $|S|$ faces in F . From Lemma 2.2, $\text{bw}(G^*) \leq \text{bw}(G'^*) + |S|$ **(3)**. Set $\mathcal{B} \leftarrow \mathcal{B} \setminus \{G\} \cup \{G'\}$ and $\mathcal{B}^* \leftarrow \mathcal{B}^* \setminus \{G^*\} \cup \{G'^*\}$.
4. Apply (recursively) Steps 2–4 for each block $G \in \mathcal{B}$ and its dual.

We now claim that before each recursive call of Steps 2–4, it holds that $\text{bw}(G^*) \leq 6 \cdot \text{bw}(G) + 2\text{eg}(\Sigma) - 4$. The proof uses descending induction on the distance from the root of the recursion tree of the above procedure. Notice that all embeddings of graphs in the collections \mathcal{B} and \mathcal{B}^* constructed by the above algorithm are polyhedral, except from the trivial case that they are just cliques of size 2. Then the theorem follows directly from Proposition 2.1.

Suppose that G (resp. G^*) is the clique sum of its blocks G_1, \dots, G_ρ (resp. G_1^*, \dots, G_ρ^*) embedded in the surfaces $\Sigma_1, \dots, \Sigma_\rho$ (Step 2). By induction, we have that $\text{bw}(G_i^*) \leq 6 \cdot \text{bw}(G_i) + 2\text{eg}(\Sigma_i) - 4, i = 1, \dots, \rho$ and the claim follows from Relations **(1)** and **(2)** and the fact that $\text{eg}(\Sigma) = \sum_{i=1, \dots, \rho} \text{eg}(\Sigma_i)$.

Suppose now (Step 3) that G (resp. G^*) occurs from some graph G' (resp. G'^*) embedded in a surface Σ' where $\text{eg}(\Sigma') < \text{eg}(\Sigma)$ after adding the vertices in S (resp. S^*). From the induction hypothesis, $\text{bw}(G'^*) \leq 6 \cdot \text{bw}(G') + 2\text{eg}(\Sigma') - 4 \leq 6 \cdot \text{bw}(G') + 2\text{eg}(\Sigma) - 2 - 4$ and the claim follows easily from Relation (3) as $|S| \leq 2$ and $\text{bw}(G') \leq \text{bw}(G)$.

3. Recent results and a conjecture

Very recently Mazoit [6] proved that treewidth is a $(1, g + 1)$ -self-dual parameter in graphs embeddable in surfaces of Euler genus g . Using that the branchwidth and the treewidth of a graph G , with $|E(G)| \geq 3$, satisfy $\text{bw}(G) \leq \text{tw}(G) + 1 \leq \frac{3}{2}\text{bw}(G)$ [8], this implies that $\text{bw}(G^*) \leq \frac{3}{2}\text{bw}(G) + g + 2$, improving the constants of Theorem 2.4. We believe that an even tighter self-duality relation holds for branchwidth and hope that the approach of this paper will be helpful to settle the following conjecture.

Conjecture 1. If G is a graph embedded in some surface Σ , then $\text{bw}(G^*) \leq \text{bw}(G) + \text{eg}(\Sigma)$.

References

- [1] O. Amini, F. Huc, and S. Pérennes. On the Pathwidth of Planar Graphs. *SIAM Journal on Discrete Mathematics*, 2009. To appear.
- [2] V. Bouchitté, F. Mazoit, and I. Todinca. Chordal embeddings of planar graphs. *Discrete Mathematics*, 273(1-3):85–102, 2003. EuroComb'01 (Barcelona).
- [3] F. V. Fomin and D. M. Thilikos. Dominating Sets in Planar Graphs: Branch-Width and Exponential Speed-Up. *SIAM Journal on Computing*, 36(2):281–309, 2006.
- [4] F. V. Fomin and D. M. Thilikos. On self duality of pathwidth in polyhedral graph embeddings. *Journal of Graph Theory*, 55(1):42–54, 2007.
- [5] D. Lapoire. Treewidth and duality for planar hypergraphs, 1996: <http://www.labri.fr/perso/lapoire/papers/dualplanar-treewidth.ps>.
- [6] F. Mazoit. Tree-width of graphs and surface duality. To appear in *DIMAP workshop on Algorithmic Graph Theory (AGT)*, Warwick, U.K., March 2009.
- [7] B. Mohar and C. Thomassen. *Graphs on surfaces*. John Hopkins University Press, 2001.
- [8] N. Robertson and P. Seymour. Graph minors. X. Obstructions to Tree-decomposition. *J. Comb. Theory Series B*, 52(2):153–190, 1991.
- [9] P. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

A simple linear-time recognition algorithm for weakly quasi-threshold graphs¹

Stavros D. Nikolopoulos, Charis Papadopoulos

Department of Computer Science, University of Ioannina
P.O.Box 1186, GR-45110 Ioannina, Greece
{stavros, charis}@cs.uoi.gr

Abstract

Weakly quasi-threshold graphs form a proper subclass of the well-known class of cographs by restricting the join operation. In this paper we characterize weakly quasi-threshold graphs by a finite set of forbidden subgraphs: the class of weakly quasi-threshold graphs coincides with the class of $\{P_4, \text{co-}(2P_3)\}$ -free graphs. Moreover we give the first linear-time algorithm to decide whether a given graph belongs to the class of weakly quasi-threshold graphs, improving the previously known running time. Based on the simplicity of our recognition algorithm, we can provide certificates of membership (a structure that characterizes weakly quasi-threshold graphs) or non-membership (forbidden induced subgraphs) in additional $\mathcal{O}(n)$ time. Furthermore we give a linear-time algorithm for finding the largest induced weakly quasi-threshold subgraph in a cograph.

1. Introduction

The well-known class of cographs is recursively defined by using the graph operations of ‘union’ and ‘join’ [4]. Bapat et al. [1], introduced a proper subclass of cographs, namely the class of *weakly quasi-threshold* graphs, by restricting the join operation and studied their *Laplacian spectrum*. In the same work they proposed a quadratic-time algorithm for recognizing such graphs. Here we characterize the class of weakly quasi-threshold graphs by the class of graphs having no P_4 (chordless path on four vertices) or $\text{co-}(2P_3)$ (the complement of two disjoint P_3 ’s). This characterization also shows that the complement of a weakly quasi-threshold graph is not necessarily weakly quasi-threshold graph. Moreover we give a tree representation for such graphs, similar to the cotrees for cographs, and propose a linear-time recognition algorithm.

¹ This research work is co-financed by E.U.-European Social Fund (75%) and the Greek Ministry of Development-GSRT (25%).

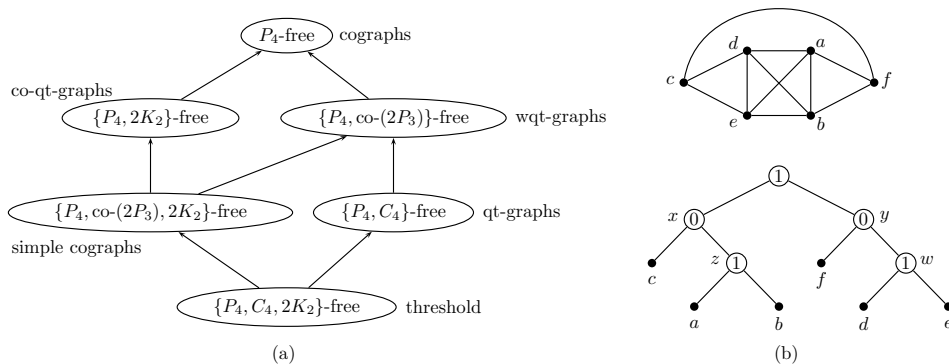


Fig. 1. (a) Subclasses of cographs and (b) a $\text{co-}(2P_3)$ and its cotree.

The class of cographs coincides with the class of graphs having no induced P_4 [5]. There are several subclasses of cographs. *Trivially-perfect* graphs, also known as *quasi-threshold* graphs, are characterized as the subclass of cographs having no induced C_4 (chordless cycle on four vertices), that is, such graphs are $\{P_4, C_4\}$ -free graphs, and are recognized in linear time [3; 6]. Another interesting subclass of cographs are the $\{P_4, C_4, 2K_2\}$ -free graphs known as *threshold* graphs, for which there are several linear-time recognition algorithms [3; 6]. Clearly every threshold graph is trivially-perfect but the converse is not true. Gurski introduced the class of $\{P_4, \text{co-}(2P_3), 2K_2\}$ -free graphs in his study of characterizing graphs of certain restricted clique-width [7]. Together with the class of weakly quasi-threshold graphs (that are exactly the class of $\{P_4, \text{co-}(2P_3)\}$ -free graphs as we show in this paper), we obtain the inclusion properties for the above families of graphs that we depict in Figure 1 (a).

For undefined terminology we refer to [3; 6]. A vertex x of G is *universal* if $N_G[x] = V(G)$ and is *isolated* if it has no neighbors in G . Two vertices x, y of G are called *false twins* if $N_G(x) = N_G(y)$. A *clique* is a set of pairwise adjacent vertices while an *independent set* is a set of pairwise non-adjacent vertices. A chordless cycle on k vertices is denoted by C_k and a chordless path on k vertices is denoted by P_k . The complement of the graph consisting of two disjoint P_3 's is denoted by $\text{co-}(2P_3)$. Given two vertex-disjoint graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their *union* is $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. Their *join* $G_1 + G_2$ is the graph obtained from $G_1 \cup G_2$ by adding all the edges between the vertices of V_1 and V_2 . The class of cographs, also known as *complement reducible graphs*, is defined recursively as follows:

- (c1) a single vertex is a cograph;
- (c2) if G_1 and G_2 are cographs, then $G_1 \cup G_2$ is also a cograph;
- (c3) if G_1 and G_2 are cographs, then $G_1 + G_2$ is also a cograph.

The class of cographs coincides with the class of P_4 -free graphs [5]. Along with other properties, it is known that cographs admit a unique tree representation, called a *cotree* [4]. For a cograph G its cotree, denoted by $T(G)$, is a rooted tree having $O(n)$ nodes. The vertices of G are precisely the leaves of $T(G)$ and every internal node of $T(G)$ is labelled by either 0 (0-node) or 1 (1-node). Two vertices are adja-

cent in G if and only if their least common ancestor in $T(G)$ is a 1-node. Moreover, if G has at least two vertices then each internal node of the tree has at least two children and any path from the root to any node of the tree consists of alternating 0- and 1-nodes. The complement of any cograph G is a cograph and the cotree of the complement of G is obtained from $T(G)$ with inverted labeling on the internal nodes of $T(G)$. Note that we distinguish between vertices of a graph and nodes of a tree. Cographs can be recognized and their cotrees can be computed in linear time [5; 8; 2].

2. A characterization of weakly quasi-threshold graphs

Bapat et al., introduced in [1] the class of *weakly quasi-threshold graphs* (or *wqt graphs* for short) and defined the given class as follows:

- (w1) a single vertex is a wqt graph;
- (w2) if G_1 and G_2 are wqt graphs then $G_1 \cup G_2$ is a wqt graph;
- (w3) if G is a wqt then adding a universal vertex in G results in a wqt graph;
- (w4) if G is a wqt graph then adding a vertex in G having the same neighborhood with a vertex of G results in a wqt graph.

By definition the class of cographs and wqt graphs have certain similarities. Clearly every wqt graph is a cograph but the converse is not true. Properties c1,c2 and w1,w2 completely coincide, whereas properties w3–w4 correspond to a restricted version of c3. Moreover it follows that in a connected wqt graph there is either a universal vertex or a false twin. Then it is not difficult to see that the class of wqt graphs is closed under taking induced subgraphs, that is, the class of wqt graphs is *hereditary*.

Lemma 2.1. The class of wqt graphs can be defined recursively as follows:

- (a1) an edgeless graph is a wqt graph;
- (a2) if G_1 and G_2 are wqt graphs then $G_1 \cup G_2$ is a wqt graph;
- (a3) if G is a wqt graph and H is an edgeless graph then $G + H$ is a wqt graph.

Proof. Properties w2 and a2 are exactly the same. By properties w1 and w2 we have that edgeless graphs are wqt graphs. We need to show that property a3 can substitute both properties w3–w4. If G is a wqt graph and H is an edgeless graph then the graph $G + H$ is obtained by first adding a universal vertex in G and then by the addition of false twins. Hence $G + H$ is a wqt graph. For the converse let G be a connected wqt graph. First observe that G can be reduced to a disconnected wqt graph $G[A]$ by repeatedly removing a universal vertex or a false twin vertex. Let S be a set of the removal vertices. Let x_n, \dots, x_k be an order of S where x_i is either universal or false twin in $G_i = G[\{x_i, \dots, x_k\} \cup A]$, $n \leq i \leq k$. We show that there is such an order of $\{x_n, \dots, x_k\}$ where all the false twin vertices appear consecutive. If there is a universal vertex x_j between two false twin vertices x_i and x_k then swapping the positions of x_j and x_k keeps the same property for the resulting order. We apply this operation for every universal vertex

between two false twin vertices and obtain an order of the vertices of S where the false twin vertices appear consecutive. Observe that the set of the false twin vertices induces an edgeless graph in G . Thus the join operation between a wqt graph and an edgeless graph is sufficient to construct a connected wqt graph.

Next we give a characterization of weakly quasi-threshold graphs through forbidden subgraphs based on Lemma 2.1.

Theorem 2.1. A graph G is weakly quasi-threshold if and only if G does not contain any P_4 or $\text{co-}(2P_3)$ as induced subgraphs.

3. A linear-time recognition algorithm

In this section we give a linear-time algorithm for deciding whether an arbitrary graph is wqt. Let G be the input graph. We first apply the linear-time recognition algorithm for checking whether G is a cograph [5]. If G is not a cograph then we know that G is not a wqt graph as it contains a P_4 . Otherwise G admits a cotree $T(G)$ that can be constructed in linear time [5; 8]. Now it suffices to efficiently check an induced $\text{co-}(2P_3)$ on G by using the cotree $T(G)$. For that purpose, we modify $T(G)$ and obtain T^* from $T(G)$ by applying the following two operations: (i) delete the subtree rooted at a 0-node having only leaves as children and (ii) remove a leaf that has 1-node as parent. Next we check if every 1-node in T^* has at most one child. In case of an affirmative answer we output that G is a wqt graph; otherwise, we output that G is not a wqt graph. Correctness of the algorithm is based on the following lemma.

Lemma 3.1. Let G be a cograph and let T^* be its modified cotree. Then G is wqt graph if and only if every 1-node of T^* has at most one child.

Theorem 3.1. Weakly quasi-threshold graphs can be recognized in $\mathcal{O}(n+m)$ time. Furthermore given a graph G there is an $\mathcal{O}(n+m)$ algorithm that reports either an induced P_4 or $\text{co-}(2P_3)$ of G whenever G is not a weakly quasi-threshold graph.

As already mentioned every wqt graph is a cograph but the converse is not necessarily true. We show that the problem of removing the minimum number of vertices from a cograph so that the resulting graph is wqt can be done in linear time. Note that the proposed algorithm can serve as a recognition algorithm as well. Let $T(G)$ be the cotree of G and let T^* be the modified cotree. Our algorithm starts by traversing both $T(G)$ and T^* from the leaves to the root and computes for each node of $T(G)$ a largest induced wqt subgraph; the one computed at the root of $T(G)$ provides the largest induced wqt subgraph of G . The computed graph is represented by a cotree T' that we construct during the traversal of $T(G)$. Let H_u be the induced subgraph of G corresponding to the leaves of the subtree rooted at a node u of $T(G)$. Every time the algorithm visits a node u of $T(G)$ it computes the triple

$(n(u), \text{MC}(u), \text{MI}(u))$ where $n(u)$ is the number of vertices of H_u , $\text{MC}(u)$ is the maximum clique of H_u , and $\text{MI}(u)$ is the maximum independent set of H_u . Let u_1, u_2, \dots, u_k be the children of u in $T(G)$. If u is a 0-node or a 1-node with at most one child then the algorithm assigns to u the correct triple by Lemma 3.1 and copies node u in T' . If u is a 1-node and u has at least two children in T^* then we need to modify the subtree rooted at u . Let $u_1^*, u_2^*, \dots, u_\ell^*$ be the children of u in T^* ; note that each child u_i^* is a 0-node, $1 \leq i \leq \ell$. Based on Lemma 3.1 we modify every subtree in $T(G)$ rooted at u_i^* except that u_i^* having the maximum value among $\min\{n(u_i^*) - |\text{MC}(u_i^*)|, n(u_i^*) - |\text{MI}(u_i^*)|\}$. For every other node $u_j^* \neq u_i^*$ we do the following operations: if $|\text{MC}(u_j^*)| > |\text{MI}(u_j^*)|$ then we delete the subtree rooted at u_j^* and add the vertices of $\text{MC}(u_j^*)$ as children of u ; otherwise we remove the nodes of the subtree rooted at u_j^* and add the vertices of $\text{MI}(u_j^*)$ as children of u_j^* .

Theorem 3.2. Given a cograph G there is an $\mathcal{O}(n + m)$ algorithm that finds a largest induced weakly quasi-threshold subgraph of G .

References

- [1] R.B. Bapat, A.K. Lal, and S. Pati. Laplacian spectrum of weakly quasi-threshold graphs. *Graphs and Combinatorics*, 24:273–290, 2008.
- [2] A. Bretscher, D. Corneil, M. Habib, and C. Paul. A simple linear time LexBFS cograph recognition algorithm. *SIAM Journal on Discrete Mathematics*, 22:1277–1296, 2008.
- [3] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey*. SIAM Monographs on Discrete Mathematics and Applications, 1999.
- [4] D.G. Corneil, H. Lerchs, and L.K. Stewart. Complement reducible graphs. *Discrete Applied Mathematics*, 3:163–174, 1981.
- [5] D.G. Corneil, Y. Perl, and L.K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14:926–934, 1985.
- [6] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Second edition. Annals of Discrete Mathematics 57. Elsevier, 2004.
- [7] F. Gurski. Characterizations for co-graphs defined by restricted NLC-width or clique-width operations. *Discrete Mathematics*, 306:271–277, 2006.
- [8] M. Habib and C. Paul. A simple linear time algorithm for cograph recognition. *Discrete Applied Mathematics*, 145:183–197, 2005.

From Sainte-Laguë to Claude Berge — French graph theory in the twentieth century^{**}

Harald Gropp^a

^a*Hans-Sachs-Str. 6, D-65189 Wiesbaden*
d12@ix.urz.uni-heidelberg.de

Key words: graph theory, Berge graph

1. Introduction

Dedicated to **CLAUDE BERGE (1926-2002)**,

mathematician and man of culture

In 1926 the zeroth book on graph theory was published by A. Sainte-Laguë [9]. It collects the knowledge on graphs at this early stage and particularly focusses on the French development of this new field in mathematics. The first French pioneer in graph theory, Georges Brunel (see [7]) prepared the first decades of the last century. Ten years after the zeroth book, in 1936, the first book on graph theory by D. König [8] was published.

Sainte-Laguë's life and work is discussed in [6]. His book [10] of 1937 (and reprinted in 1994) contains the analysis of many mathematical games and famous problems in combinatorics, e.g. *La Tour d'Hanoï*, *Les quinze demoiselles*, *Les trente-six officiers*, *La ville de Koenisberg* and *Le jeu d'Hamilton*.

In the *annexe* of the new edition of 1994 Claude Berge discusses the starting points of the abstract theory of graphs.

De très nombreux problèmes de ce livre ont été le point de départ de théorèmes généraux.
Encore fallait-il poser les bases d'une théorie abstraite.

^{**}This paper was not actually presented at the conference, as the author withdrew his participation.

One of the examples which Berge discusses is the *Problème du loup, de la chèvre, et du chou*. Berge also displays a graph of the problem. The general background and the history of these river-crossing problems is further described in [5].

1926 is also the birth year of Claude Berge who died in 2002 and was not only the most influential man in French graph theory in the second half of the twentieth century but also somebody very broadly interested in cultural fields like literature and Oceanic art.

In 1958 Claude Berge published his first book on graph theory [1] which was soon translated into several other languages.

2. Sainte-Laguë's zeroth book on graph theory

Sainte-Laguë's book [9] is not much known today. It is not available in many libraries. Even in France it is nearly forgotten. In König's book [8] it is mentioned as a reference several times. Claude Berge was one of the few mathematicians who really made use of it. It should be discussed whether it would be useful to reprint the book, together with comments and perhaps a translation into English.

This short report does not replace my paper [6] but will just give a brief introduction. André Sainte-Laguë was born in Saint-Martin-Curton (Dépt. Lot-et-Garonne) on April 20, 1882. He died in Paris on January 18, 1950. After he had studied mathematics till 1906 he became a teacher at different schools between 1906 and 1927 when he joined the CNAM (Conservatoire National des Arts et Métiers) in Paris. In 1937 he organized the mathematical presentations for the world exhibition in Paris, and in 1938 he got the chair of mathematics and applications at the CNAM. During the German occupation of France in World War II he was a leading member of the *Résistance*.

Sainte-Laguë wrote his dissertation on graphs in 1924 which contained already many proofs of theorems which he presented in his book of 1926. This book contains of 9 chapters on 64 pages. The 9 chapters are as follows: Introduction and definition, Trees, Chains and circuits, Regular graphs, Cubic graphs, Incidence matrices, Hamiltonian graphs, Chess problems, and Knight's move problems where the titles are given in modern terminology and not in the words of Sainte-Laguë. The list of 223 references is a good survey of the combinatorial literature earlier than 1926.

Sainte-Laguë describes important sources for graph theory such as recreational problems or physics or chemistry.

3. Claude Berge's books

In the following two of Claude Berge's books will be further discussed in order to show his attitude towards graph theory and combinatorics and their history. A third book will be briefly mentioned. It is not the aim of this short paper to give a full survey on Claude Berge's books. In this extended abstract the books will only be briefly described.

3.1 *Claude Berge: The Theory of Graphs and its applications (1962), French 1958*

Claude Berge's book of 1958 [1] was a breakthrough for the development of the new mathematical field, called graph theory, not only in France, but for the whole world. After the books of Sainte-Laguë (1926) [9] and König (1936) [8] who for different reasons did not become widely circulated the book of Berge found a broad acceptance and was soon translated into many other languages. In his introduction of two pages Berge introduces graphs as applied in physics, chemistry, economics, psychology etc. This close link to all possible areas of applications was certainly one of the main aspects of Claude Berge's graph theory.

3.2 *Claude Berge: Principles of Combinatorics (1971), French 1968*

What is Combinatorics ?

In the introduction of his book *Principles of Combinatorics* [2] Claude Berge describes the main characteristics of combinatorics by using *configurations* as combinatorial structures. Configurations are here just special objects with certain constraints, not configurations as defined by Reye and discussed in many of my papers (e.g. see [6]).

3.3 *Claude Berge: Graphes et Hypergraphes (1970)*

It should not be forgotten that sometimes Claude Berge was called *Monsieur la théorie des hypergraphes*. In fact he pushed forward this extended aspect of graph theory very much, also as the author of his book on hypergraphs [4]. Hypergraphs were only "invented" around 1960, but similar concepts had already been around much earlier. The real importance of these combinatorial structures will only become clear in the future and will not be further discussed in this short paper.

4. Claude Berge, literature and art

Last but not least let me mention here Claude Berge's activities in literature and art. He was a member of the group *OULIPO* (Ouvroir de Littérature Potentielle) which was founded in 1960 and works on the connections between mathematics and literature. Some prominent members as writers are Raymond Queneau and Georges Perec. Claude Berge himself wrote the novel *Qui a tué le Duc de Densmore ?* in 1994 [3] in which he used a combinatorial theorem of Hajós to tell a criminal story.

It is generally known that Claude Berge was very much interested in the cultures of the Pacific Ocean, in particular in the art of Papua - New Guinea. He himself had sculptured similar objects and collected all kind of information on Oceanic Art. As a small footnote let me mention here that he was in close contact to the Konrad family in Mönchengladbach (Germany) who gave an important Asmat collection to the *Völkerkundemuseum* in Heidelberg. The Asmat are one of the many peoples in Papua.

5. A few last words

Let me close this short paper which discusses two remarkable and unusual French mathematicians of the twentieth century by explicitly mentioning the extreme friendliness and kindness of Claude Berge. Although he became one of the most prominent and important experts in his field he always stayed a man just very much interested in many things, not only in mathematics and graph theory.

The very last words should just remind us of Claude Berge's enormous influence on the graph theory of the twentieth century and express our thanks (in the French language, of course):

Merci beaucoup !

References

- [1] C. Berge, *Théorie des graphes et ses applications*, Paris (1958), in English: *The theory of graphs and its applications*, London-New York (1962).
- [2] C. Berge, *Principes de combinatoire*, Paris (1968), in English: *Principles of combinatorics*, New York-London (1971).
- [3] C. Berge, *Qui a tué le Duc de Densmore ?*, *Bibliothèque Oulipienne* (1994).
- [4] C. Berge, *Graphes et hypergraphes*, Paris (1970).

- [5] H.Gropp, *Propositio de lupo et capra et fasciculo cauli* — On the history of river-crossing problems, in: *Charlemagne and his heritage, 1200 years of civilization and science in Europe*, Vol. 2, (eds. P.L. Butzer, H.Th. Jongen, W. Oberschelp), Turnhout (Belgium) (1998), 31-41.
- [6] H. Gropp, On configurations and the book of Sainte-Laguë, *Discrete Math.* 191 (1998), 91-99.
- [7] H. Gropp, "Réseaux réguliers" or regular graphs — Georges Brunel as a French pioneer in graph theory, *Discrete Math.* 276 (2004), 219-227.
- [8] D. König, *Theorie der endlichen und unendlichen Graphen*, Leipzig (1936).
- [9] A. Sainte-Laguë, *Les réseaux (ou graphes)*, Paris (1926).
- [10] A. Sainte-Laguë, *Avec des nombres et des lignes*, Paris (1937).

Combinatorial Optimization

Lecture Hall A

Tue 2, 10:30–12:30

Colored Independent Sets

J. Maßberg and T. Nieberg

*Research Institute for Discrete Mathematics
University of Bonn, Lennéstr. 2, 53113 Bonn, Germany
{massberg, nieberg}@or.uni-bonn.de*

Abstract

We study graphs with colored vertices and prove that it is NP-complete to decide if there is an independent set in the graph containing at least one vertex of each color. The proof immediately yields that the problem remains NP-complete even when restricting the graph to the class of Unit Disk Graphs (UDGs).

We present and discuss also an application where the problem arises in the area of VLSI routing: Conflict-free and thus disjoint wiring interconnects for a set of pins on a circuit have to be chosen from a precomputed set of paths.

Key words: Colored Graphs, Unit Disk Graph, VLSI Design

1. Introduction

Given a graph G together with a color $\kappa_v \in \{1, \dots, K\}$ for each vertex $v \in V(G)$, we seek an *independent set* $I \subseteq V(G)$ in G that maximizes the objective function

$$c(I) := \min_{k \in \{1, \dots, K\}} |\{i \in I \mid \kappa_i = k\}|.$$

Clearly, if all vertices have the same color, this amounts to the classical Maximum Independent Set problem, which is known to be NP-hard. The COLORED INDEPENDENT SET decision problem is the following:

Given an $r \in \mathbb{N}$, is there an independent set $I \subseteq V(G)$ with $c(I) \geq r$?

In the following, we restrict ourselves to the class of Unit Disk Graphs which capture the same geometric property of the application presented next.

Definition 1. A graph G is called a UNIT DISK GRAPH (UDG) if there exists a map $f : V(G) \rightarrow \mathbb{R}^2$ satisfying: $(u, v) \in E(G) \Leftrightarrow \|f(u) - f(v)\| \leq 1$.

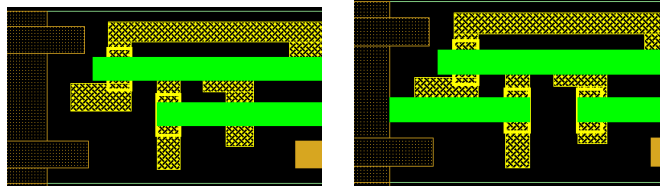


Fig. 1. Conflicting and conflict-free pin access paths (via and wiring on higher layer) for a circuit with three pins (shaded on lower layer).

2. Application

We consider the following problem in detailed VLSI routing, where the above COLORED INDEPENDENT SET problem occurs. Given a circuit (a collection of pins) placed on the chip-area, we want to connect each pin by a short access path that legally connects it to the overall routing grid used to cover larger distances.

Usually, the routing is done sequentially, i.e. one connection after the next, in order to create disjoint interconnects. Especially when the pins are situated very dense, this gives frequent complications when a path blocks the access to a not yet connected pin (see Figure 1).

Our solution to this problem is to preprocess each circuit by first computing a set of access paths for each pin of a given circuit, and then selecting a disjoint and conflict-free subset that is used in the sequential routing phase. For the latter, a conflict graph is built for a circuit so that each path connecting to a certain pin receives the same color, and two paths are connected by an edge if they create a short circuit when used at the same time.

Clearly, an independent set having a vertex (i.e. path) of each color (i.e. pin) corresponds to a conflict-free pin access situation.

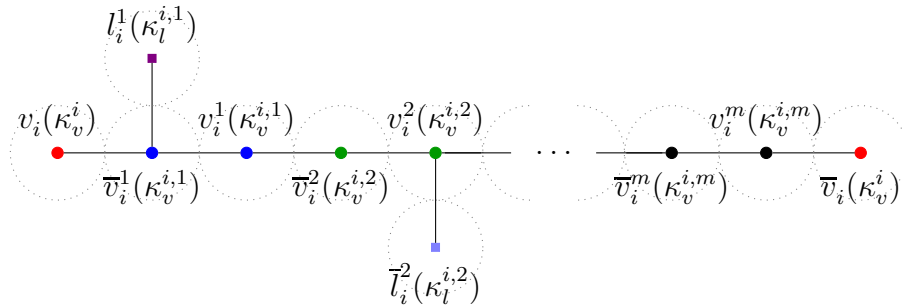


Fig. 2. Example for a 'variable' graph G_{x_i} . Here $x_i \in Z_1$ and $\bar{x}_i \in Z_2$. The color of the vertices are enclosed in brackets.

3. NP-Completeness

In this section we give our main result, formulated for Unit Disk Graphs.

Theorem 1. The COLORED INDEPENDENT SET IN UDGs decision problem is NP-complete even for $r = 1$.

Membership in NP is obvious. We prove that 3SATISFIABILITY polynomially transforms to COLORED INDEPENDENT SETS IN UDGs. Given a collection \mathcal{Z} of clauses Z_1, \dots, Z_m over $X = \{x_1, \dots, x_n\}$, each clause containing three literals, we shall construct a UDG G that contains an independent set I with $c(I) \geq 1$ iff \mathcal{Z} is satisfiable.

The graph G contains for each variable $x_i \in X$ two vertices v_i, \bar{v}_i of color κ_v^i , and for pair of variable $x_i \in X$ and clause $Z_j \in \mathcal{Z}$ two vertices v_i^j, \bar{v}_i^j of color $\kappa_v^{i,j}$. These vertices indicate whether the variables x_i are true or false.

Additionally the graph contains for each variable in a clause $x_i \in Z_j$ two vertices l_i^j, \bar{l}_i^j of color $\kappa_l^{i,j}$ linking the variables and the clauses.

Moreover we have for each clause Z_j containing the variables x_a, x_b, x_c four vertices z_a^j, z_b^j, z_c^j and \bar{z}^j of color κ_z^j showing if the corresponding literals and the complete clause are satisfied or not.

Finally we have a 'satisfiability' vertex s of color κ_s indicating if \mathcal{Z} is satisfiable in total. Note that all defined colors are pairwise disjoint.

Based on these vertices, the graph contains three different types of subgraphs representing the variables, the clauses and the last one showing if the problem is satisfiable.

The first type are the graphs G_{x_i} representing the variables $x_i \in X$. Let $A_{x_i} := \{v_i, \bar{v}_i\} \cup \{v_i^j, \bar{v}_i^j \mid 1 \leq j \leq m\}$, $B_{x_i} := \{l_i^j \mid x_i \in Z_j\}$, $\bar{B}_{x_i} := \{\bar{l}_i^j \mid \bar{x}_i \in Z_j\}$. Then,

$$\begin{aligned} V(G_{x_i}) &:= A_{x_i} \cup B_{x_i} \cup \bar{B}_{x_i} \text{ and} \\ E(G_{x_i}) &:= \{(v_i, \bar{v}_i^1), (\bar{v}_i^1, v_i^1), (v_i^1, \bar{v}_i^2), \dots, (\bar{v}_i^m, v_i^m), (v_i^m, \bar{v}_i)\} \\ &\quad \cup \{(l_i^j, \bar{v}_i^j) \mid x_i \in Z_j\} \cup \{(\bar{l}_i^j, v_i^j) \mid \bar{x}_i \in Z_j\}. \end{aligned}$$

It is easy to see that there is a UDG representation for G_{x_i} (see Figure 2): The subgraph induced by A_{x_i} is a path from v_i to \bar{v}_i . We place the vertices of the path on a horizontal line with distance 1 between two intermediate vertices. The vertices of B_{x_i} will be placed above and the vertices of \bar{B}_{x_i} below their adjacent vertices of A_{x_i} , again at distance 1.

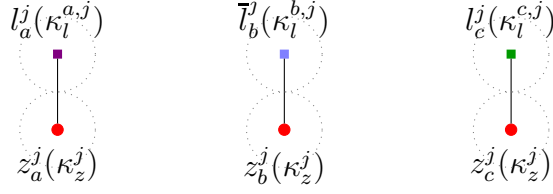


Fig. 3. Example for a 'clause' graph G_{Z_j} for $Z_j = \bar{x}_a \vee x_b \vee \bar{x}_c$.

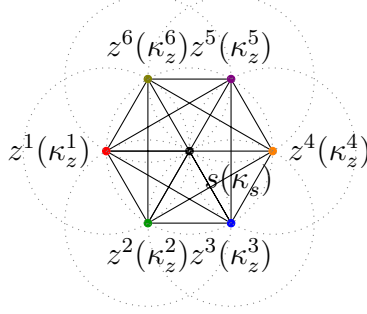


Fig. 4. Graph S for a 3SAT instance with 6 clauses.

The second type of graphs are the graphs G_{Z_j} representing each clause $Z_j \in \mathcal{Z}$ (Figure 3). We set

$$V(G_{Z_j}) := \{z_i^j \mid x_i \in Z_j \vee \bar{x}_i \in Z_j\} \cup \{\bar{l}_i^j \mid x_i \in Z_j\} \cup \{l_i^j \mid \bar{x}_i \in Z_j\} \text{ and}$$

$$E(G_{Z_j}) := \{(z_i^j, \bar{l}_i^j) \mid x_i \in Z_j\} \cup \{(z_i^j, l_i^j) \mid \bar{x}_i \in Z_j\}.$$

The last type just contains a complete graph S with $V(S) := \{z^1, \dots, z^j, s\}$ and $E(S) := \{(v, w) \mid v, w \in V(S), v \neq w\}$ (Figure 4).

The graph G is the union of $\{G_{x_i}\}_{1 \leq i \leq n}$, $\{G_{Z_k}\}_{1 \leq k \leq m}$ and S .

It is evident that this is a polynomial transformation, and it now remains to show that G correctly encodes the instance \mathcal{Z} .

- \mathcal{Z} is satisfiable $\Rightarrow G$ contains an independent set I with $c(I) \geq 1$.

Let $T : X \rightarrow \{\text{true}, \text{false}\}$ be a truth assignment that satisfies \mathcal{Z} . We show that there exists an independent set I in G with $c(I) \geq 1$. Set

$$I := \{v_i, v_i^1, \dots, v_i^m \mid T(x_i) = \text{true}\} \cup \{\bar{v}_i, \bar{v}_i^1, \dots, \bar{v}_i^m \mid T(x_i) = \text{false}\}$$

$$\cup \{l_i^j \mid x_i \in Z_j \vee \bar{x}_i \in Z_j, T(x_i) = \text{true}\}$$

$$\cup \{\bar{l}_i^j \mid x_i \in Z_j \vee \bar{x}_i \in Z_j, T(x_i) = \text{false}\}$$

$$\cup \{z_i^j \mid (x_i \in Z_j \wedge T(x_i) = \text{true}) \vee (\bar{x}_i \in Z_j \wedge T(x_i) = \text{false})\} \cup \{s\}.$$

It is easy to verify that this is indeed an independent set. Now we have to show that each color is represented by a vertex of I . Obviously there are vertices of colors $\kappa_v^i, \kappa_v^{i,j}, \kappa_l^{i,j}, \kappa_s, 1 \leq i \leq n, 1 \leq j \leq m$ in I .

It remains to show that for each $1 \leq j \leq m$ there is a vertex of color κ_z^j in I . As each clause Z_j is satisfied, there is an $x_i \in Z_j$ with $T(x_i) = \text{true}$ or an $\bar{x}_i \in Z_j$ with $T(x_i) = \text{false}$. This gives $z_i^j \in I$ in both cases.

- **G contains an independent set I with $c(I) \geq 1 \Rightarrow \mathcal{Z}$ is satisfiable.**

Let I be an independent set I with $c(I) \geq 1$. The task now is to construct a truth assignment $T : X \rightarrow \{\text{true}, \text{false}\}$ that satisfies \mathcal{Z} .

Set $P_i = \{v_i, v_i^1, \dots, v_i^m\}$ and $N_i = \{\bar{v}_i, \bar{v}_i^1, \dots, \bar{v}_i^m\}$ for $1 \leq i \leq n$. Note that all vertices of $P_i \cup N_i$ are on a path and elements of P_i are only adjacent to vertices of N_i and vice versa. By construction of G_{x_i} either $(P_i \subset I \text{ and } N_i \cap I = \emptyset)$ or $(N_i \subset I \text{ and } P_i \cap I = \emptyset)$. In the first case we set $T(x_i) = \text{true}$ and in the second case $T(x_i) = \text{false}$.

Now let $Z_j \in \mathcal{Z}$ be a clause. We claim that Z_j is satisfied. Since $c(I) \geq 1$, the vertex s must be in I as it is the only vertex of color κ_s . The vertices z^j and s are adjacent and I is an independent set so $z^j \notin I$. But there must be a vertex of color κ_z^j in I , i.e. there exists an i with $z_i^j \in I$. We have either $x_i \in Z_j$ or $\bar{x}_i \in Z_j$. In the case $x_i \in Z_j$, the vertex z_i^j is connected to \bar{l}_i^j and therefore $\bar{l}_i^j \notin I$. From this it follows that $l_i^j \in I$ as there must be a vertex of color $\kappa_l^{i,j}$ in I . But then $\bar{v}_i^j \notin I$ which means that we have set $T(x_i) = \text{true}$. The clause Z_j is satisfied. Similar arguments apply to the case that $\bar{x}_i \in Z_j$. Here we conclude that $T(x_i) = \text{false}$ and again get that Z_j is satisfied.

Therefore \mathcal{Z} is satisfied, and the proof is complete. □

A note on the parameterized complexity of the maximum independent set problem

Vadim V. Lozin^a

^a*DIMAP and Mathematics Institute, University of Warwick, Coventry, UK*

Key words: Parameterized complexity, Independent set, Ramsey Theory

1. Introduction

We study the MAXIMUM INDEPENDENT SET problem parameterized by the solution size k , which we call k -INDEPENDENT SET. A parameterized problem is *fixed-parameter tractable* (fpt for short) if it can be solved in $f(k)n^{O(1)}$ time, where $f(k)$ is a computable function depending on the value of the parameter only. In general, the k -INDEPENDENT SET problem is W[1]-hard, which means it is not fixed-parameter tractable unless $P = NP$. On the other hand, fpt-algorithms have been developed for segment intersection graphs with bounded number of directions [6], triangle-free graphs [8], graphs of bounded vertex degree [5], planar graphs, and more generally, graphs excluding a single-crossing graph as a minor [3]. A common feature of all these classes is that all of them are hereditary (i.e., closed under vertex deletion) and all of them are small in the following sense. It is known (see e.g. [2]) that for every hereditary class X , the number X_n of n -vertex graphs in X (also known as the speed of X) satisfies $\lim_{n \rightarrow \infty} \frac{\log_2 X_n}{\binom{n}{2}} = 1 - \frac{1}{k(X)}$, where $k(X)$ is a natural number called the *index* of the class. The triangle-free graphs have index 2 and the index of all other classes mentioned above is 1 (see [7] for the speed of minor-closed graph classes). In this paper, we focus on hereditary classes of index $k > 1$. Each class X in this range can be approximated by a minimal class of the same index. The main result of this paper is that the problem is fixed-parameter tractable in *all* minimal classes of index k for *all* values of k .

We use the following notations. For a subset $U \subseteq V(G)$, we denote by $G[U]$ the subgraph of G induced by U . K_n stands for the complete graph on n vertices and \overline{K}_n for its complement. Also, pK_2 is the disjoint union of p copies of K_2 . For a set of graphs M , we denote by $Free(M)$ the class of graphs containing no induced subgraphs isomorphic to graphs in M . It is known that a class X of graphs is hereditary if and only if $X = Free(M)$ for a certain set M . For two graph classes

X and Y , denote by XY the class of graphs whose vertices can be partitioned into two subsets, one of which induces a graph in X and another one a graph in Y . Let us denote by $G_1 \vee G_2$ the union of two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ with a common vertex set V , i.e., $G_1 \vee G_2 = (V, E_1 \cup E_2)$. If A and B are two classes of graphs, then $A \vee B := \{G_1 \vee G_2 : G_1 \in A, G_2 \in B\}$.

2. Complexity of the problem in classes of high speed

The main result of the paper is consequence of a series of technical lemmas.

Lemma 1. Let A, B be two classes of graphs such that (1) $A \subseteq \text{Free}(pK_2)$ for some constant p , (2) B is a hereditary class of graphs admitting an fpt-algorithm for the k -INDEPENDENT SET problem, (3) there is an algorithm that for any graph $G \in A \vee B$, finds in polynomial time two graphs $G_1 \in A$ and $G_2 \in B$ such that $G = G_1 \vee G_2$. Then the k -INDEPENDENT SET problem is fixed-parameter tractable in the class $A \vee B$.

Proof. An fpt-algorithm for graphs in $A \vee B$ can be outlined as follows. Given a graph $G \in A \vee B$, first, find two graphs $G_1 \in A$ and $G_2 \in B$ such that $G = G_1 \vee G_2$. Next, for each maximal under inclusion independent set I in G_1 , solve the k -INDEPENDENT SET problem in $G_2[I] \in Y$ by an fpt-algorithm. If the algorithm finds an independent set of size k in $G_2[I] \in Y$, output this set for the graph G . Otherwise (i.e., if the fpt-algorithm says NO for each graph $G_2[I] \in Y$), answer NO for the graph G . Correctness of the procedure follows from the fact that every independent set in G also is independent both in G_1 and G_2 . To estimate its time complexity, observe that the number of inclusionwise maximal independent sets in pK_2 -free graphs is bounded by a polynomial [1] and all of them can be found in polynomial time [9].

Lemma 2. If XY is a class of graphs with $X \subseteq \text{Free}(\overline{K}_m)$ and $Y \subseteq \text{Free}(K_n)$, then $XY \subset \text{Free}(mK_2) \vee \text{Free}(K_n)$.

Proof. Let $G = (V, E)$ be a graph in XY and let $V = V_1 \cup V_2$ be a partition of V such that $G[V_1] \in X$ and $G[V_2] \in Y$. Denoting $G_1 = (V, E - E(G[V_2]))$ and $G_2 = (V, E(G[V_2]))$, we conclude that $G = G_1 \vee G_2$. Obviously, $G_2 \in \text{Free}(K_n)$. To see that $G_1 \in \text{Free}(mK_2)$ observe that if M is an induced subgraph of degree 1 in G_1 then at least one endpoint of each edge of M belongs to V_1 (because V_2 is independent in G_1). Since V_1 can contain at most $m - 1$ independent vertices, the size of M is at most $m - 1$.

Lemma 3. For any constant q , the k -INDEPENDENT SET problem is fixed-parameter tractable in the class $\text{Free}(K_q)$.

Proof. It can be decided in time $O(n^q)$ if G has an independent set of size $k \leq q$. For $k > q$, we employ the Ramsey theory. Let $R(t)$ be the diagonal Ramsey number. It is known [4] that $R(t) \leq 2^{2t-3}$. Moreover, the proof given in [4] is constructive and shows how to find in a graph with $n \geq 2^{2t-3}$ vertices a subset inducing an independent set or a clique of size t in time $O(tn)$.

Let $k > q$. If the number of vertices of a graph $G \in \text{Free}(K_q)$ is at most 2^{2k-3} , then we can check in time $O(2^{(2k-3)k})$ if G has an independent set of size k . If G has $n > 2^{2k-3}$ vertices, we know that G has an independent set of size k (because G is K_q -free) and this set can be found in time $O(kn)$.

Lemma 4. For any $X \subseteq \text{Free}(\overline{K}_m)$ and $Y \subseteq \text{Free}(K_n)$, there exists a constant $\tau = \tau(X, Y)$ such that for every graph $G = (V, E) \in XY$ and every subset $B \subseteq V$ with $G[B] \in Y$, at least one of the following statements holds:

- (a) $\exists A \subseteq V$ such that $G[A] \in Y$, $G[V - A] \in X$, and $|A - B| \leq \tau$,
- (b) $\exists C \subseteq V$ such that $G[C] \in Y$, $|C| = |B| + 1$, and $|B - C| \leq \tau$.

Proof. By Ramsey Theorem, for each positive integers m and n , there is a constant $R(m, n)$ such that every graph with at least $R(m, n)$ vertices contains either a \overline{K}_m or a K_n as an induced subgraph. Given two sets $X \subseteq \text{Free}(\overline{K}_m)$ and $Y \subseteq \text{Free}(K_n)$, we define $\tau = \tau(X, Y)$ to be equal $R(m, n)$.

Let $G = (V, E)$ be a graph in XY , and B a subset of V such that $G[B] \in Y$. Consider an arbitrary subset $A \subseteq V$ such that $G[A] \in Y$ and $G[V - A] \in X$. If (a) does not hold, then $|A - B| > \tau$. In addition, $G[B - A] \in X \cap Y \subseteq \text{Free}(\overline{K}_m, K_n)$, and hence $|B - A| \leq \tau$. Consequently, $|A| > |B|$. But then any subset $C \subseteq A$ such that $A \cap B \subseteq C$ and $|C| = |B| + 1$ satisfies (b).

Theorem 1. If m and n are constants and XY is a class of graph such that $X \subseteq \text{Free}(\overline{K}_m)$ and $Y \subseteq \text{Free}(K_n)$, then the k -INDEPENDENT SET problem is fixed-parameter tractable in the class XY .

Proof. We apply Lemma 1. Conditions (1) and (2) of the lemma follows from Lemmas 2 and 3. For condition (3), we develop the following algorithm:

Input: A graph $G = (V, E) \in XY$ with $X \subseteq \text{Free}(\overline{K}_m)$ and $Y \subseteq \text{Free}(K_n)$

Output: Graphs $G_1 \in \text{Free}(mK_2)$, $G_2 \in \text{Free}(K_n)$ such that $G = G_1 \vee G_2$.

- (1) Find in G any maximal under inclusion subset $B \subseteq V$ inducing a graph in $\text{Free}(K_n)$.
- (2) If there is a subset $C \subseteq V$ satisfying condition (b) of Lemma 4, then set $B := C$ and repeat Step (2).
- (3) Find in G a subset $A \subseteq V$ such that $|B - A| \leq \tau$, $|A - B| \leq \tau$, $G[A] \in \text{Free}(K_n)$, $G_1 = (V, E - E(G[A])) \in \text{Free}(mK_2)$.
- (4) Output $G_1 = (V, E - E(G[A]))$ and $G_2 = (V, E(G[A]))$.

Correctness of the algorithm follows from Lemmas 2 and 4. To estimate its time complexity, observe that in Step (2) the algorithm inspects at most $\binom{|V|}{\tau} \binom{|V|}{\tau+1}$ subsets C and for each of them, verifies whether $G[C] \in \text{Free}(K_n)$ in time $O(|V|^n)$. Since Step (2) loops at most $|V|$ times, its time complexity is $O(|V|^{2\tau+n+2})$. In Step (3), the algorithm examines at most $\binom{|V|}{\tau}^2$ subsets A . For each A , it verifies whether $G[A] \in \text{Free}(K_n)$ in time $O(|V|^n)$ and whether $G[V - A] \in \text{Free}(\overline{K}_m)$ (and hence $G_1 \in \text{Free}(mK_2)$) in time $O(|V|^m)$. Summarizing, we conclude that the total time complexity of the algorithm is $O(|V|^{2\tau+m+n})$.

Denote by $\mathcal{E}_{i,j}$ the class of graphs whose vertices can be partitioned into at most i independent sets and j cliques. Then the index $k(X)$ of a class X is the maximum k such that X contains a class $\mathcal{E}_{i,j}$ with $i+j = k$, i.e. the classes $\mathcal{E}_{i,j}$ with $i+j = k$ are the only minimal classes of index k . Obviously, $\mathcal{E}_{i,j} \subseteq \text{Free}(\overline{K}_{i+1})\text{Free}K_{j+1}$. Therefore,

Corollary 1. For any natural i and j , the k -INDEPENDENT SET problem is fixed-parameter tractable in the class $\mathcal{E}_{i,j}$.

References

- [1] E. Balas, Ch.S. Yu, On graphs with polynomially solvable maximum-weight clique problem, *Networks* 19 (1989) 247–253.
- [2] J. Balogh, B. Bollobás, D. Weinreich, The speed of hereditary properties of graphs, *J. Combin. Theory, Ser. B* 79 (2000) 131–156.
- [3] E.D. Demaine, M. Hajiaghayi, D.M. Thilikos, Exponential speedup of fixed-parameter algorithms for classes of graphs excluding single-crossing graphs as minors, *Algorithmica*, 41 (2005) 245–267.
- [4] R. Diestel, Graph theory. Third edition. Graduate Texts in Mathematics, 173. Springer-Verlag, Berlin, 2005. xvi+411 pp.
- [5] J. Flum, M. Grohe, Parameterized complexity theory. Texts in Theoretical Computer Science. Springer-Verlag, Berlin, 2006. xiv+493 pp.
- [6] J. Kára, J. Kratochvíl, Fixed parameter tractability of Independent Set in segment intersection graphs, *LNCS*, 4169 (2006) 166–174.
- [7] S. Norine, P. Seymour, R. Thomas, P. Wollan, Proper minor-closed families are small, *J. Combinatorial Theory Ser. B*, 96 (2006) 754–757.
- [8] V. Raman, S. Saurabh, Triangles, 4-Cycles and Parameterized (In-)Tractability, *Lecture Notes in Computer Science*, 4059 (2006) 304–315.
- [9] S. Tsukiyama, M. Ide, H. Ariyoshi, I. Shirakawa, A new algorithm for generating all the maximal independent sets, *SIAM J. Computing*, 6 (1977) 505–517.

On multi-agent knapsack problems

Gaia Nicosia,^a Andrea Pacifici,^b Ulrich Pferschy^c

^a*Dipartimento di Informatica e Automazione, Università degli studi “Roma Tre”, Italy*
nicosia@dia.uniroma3.it

^b*Dipartimento di Ingegneria dell’Impresa, Università degli Studi di Roma “Tor Vergata”, Italy*
pacifici@disp.uniroma2.it

^c*Institut für Statistik und Operations Research, Universität Graz, Austria*
pferschy@uni-graz.at

Key words: multi-agent optimization, knapsack and subset sum problem, games.

1. Introduction

In this work we consider knapsack-like problems in a multi-agent setting. These kinds of problems occur in several different application environments and different methodological fields, such as artificial intelligence, decision theory, operations research etc. We focus on the following situation: There are two agents, each of them owning one of two disjoint sets of items. The agents have to select items from their set for packing them in a common knapsack and thus sharing a given common resource. Each agent wants to maximize a payoff function which is given by its own items’ profits. The problem is how to compute solutions which take into account each agent’s payoff function, and that can be used to support the negotiation among the agents.

We present some results about two different classes of problems, namely, a subset sum game and a special knapsack game with unitary weights. We characterize Pareto and global optima and provide solution algorithms both in the centralized and multi-agent scenarios.

The problem that we address in this work is relatively new, however 0-1 knapsack problems (KP) in a multi-decision environment have been considered in the literature for two decades: from game-theoretic to auction scenarios there is a variety of papers dealing with this classical combinatorial optimization problem. Hereafter, we limit to report a few of them.

A related problem in which different players try to fit their own items in a common knapsack is the so called *knapsack sharing problem* studied by several authors (see for instance [3; 4]). A single objective function that tries to balance the profits among the players is considered in a centralized perspective.

An interesting game, based on the maximum 0-1 knapsack, interpreted as a special on-line problem, is addressed in [5] where a two person zero-sum game, called *knapsack game*, is considered. Knapsack problems are also considered in the context of auctions. For instance, in [1], an application for selling advertisements on Internet search engines is considered. In particular, there are n agents wishing to place an item in the knapsack and each agent gives a private valuation for having an item in the knapsack, while each item has a publicly known size.

In the following, A and B indicate both the agents' names and the corresponding set of items, while $n(A)$ and $n(B)$ denote the number of items of each agent. Moreover, p_i^A, p_j^B are the profits and w_i^A, w_j^B are the weights of items $i \in A$ and $j \in B$, respectively. Finally, let c be the capacity of the knapsack.

2. Subset Sum game with rounds

Here we consider a subset sum game, i.e. $p_i^A = w_i^A$ and $p_j^B = w_j^B$ for all items $i = 1, \dots, n(A)$, $j = 1, \dots, n(B)$. The aim of the game is for each agent to select a subset of its items with maximum total weight. The game can be seen as a sequence of *rounds*, where in each round A selects one of its items (which was not selected before) and puts it into the knapsack. Then B does the same. The total weight of all selected items must not exceed the capacity c at any time. All information is public. We adopt a sort of online perspective, in which we want to determine the best strategy for agent A assuming that B is rational and is pursuing its own objective.

Given any deterministic strategy of B an optimal strategy of agent A can be computed via backward induction by enumerating all possible sequences of item selection in a decision tree, similar to a game in extensive form. Naturally, this takes exponential time.

In contrast to this intractable approach we provide a natural greedy algorithm for this problem, where agent A simply selects in every round the item with the largest weight that does not violate the capacity constraint. We show that such a greedy algorithm may reach only half of the weight attained by an optimal strategy but can not do worse than that.

It can also be shown that the price of anarchy, i.e. the ratio between a centrally determined optimal solution maximizing the sum of weights selected by both play-

ers over the sum of weights derived by a selfish optimization of each player, can be arbitrarily high.

3. Multi-Agent Knapsack

In this section, we consider the multi-agent problem in another scenario. Here, weights $w_i^A = 1$ and $w_j^B = 1$ for all items $i = 1, \dots, n(A)$, $j = 1, \dots, n(B)$. Therefore, given the capacity $c \in \mathbb{Z}_+$, exactly c items fit into the knapsack. We may view the game as a set of c rounds where, in each round, the two agents pick one of their (unpacked) items and submit it for being packed in the knapsack. Only one of the two agents items is packed (i.e. wins this round) and the item of the other agent is *discarded*. At the end of the c rounds, each agent has a profit corresponding to the total profit of its packed items. We assume the input list of available items is public, but the submissions in each round occur simultaneously and in secret. However most of the following results are sort of off-line centralized results.

Suppose agent A submits an item i and agent B an item j . Here, we consider two possible rules for deciding which of the two submitted items wins and is packed into the knapsack:

Rule 1: if $p_i^A > p_j^B$ then A wins;

Rule 2: if $p_i^A < p_j^B$ then A wins.*

A graph model is useful to represent this problem. Each agent's item is associated to a node of a complete bipartite graph $G = (A \cup B, E_A \dot{\cup} E_B)$. An arc (i, j) belongs to E_A or to E_B depending on the rule, namely

Rule 1: (i, j) , with profit $p_{ij} = \max\{p_i^A, p_j^B\}$, belongs to E_A if $p_i^A \geq p_j^B$ (i.e. if A wins), otherwise it belongs to E_B ;

Rule 2: (i, j) , with profit $p_{ij} = \min\{p_i^A, p_j^B\}$, belongs to E_A if $p_i^A \leq p_j^B$ (i.e. if A wins), otherwise it belongs to E_B .

Any solution may be represented as a c -matching M on G , where the payoff of A is $p_A(M) = \sum_{ij \in M \cap E_A} p_{ij}$ and that of B is $p_B(M) = \sum_{ij \in M \cap E_B} p_{ij}$. Thus, determining a global optimum can be done in polynomial time by solving a weighted cardinality assignment problem [2]. Note that matching problems on graphs with edges partitioned into two sets are explicitly addressed in [6].

Obviously, in case of Rule 1 each agent will always submit its c most profitable items. For this case we prove the following results.

- There is no *preventive* strategy for the agents, i.e. for any possible strategy one

* In case of a tie we assume that A always wins.

agent, say A , may choose, it cannot attain more than its worst solution since B can always maximize its own payoff (thus minimizing A 's payoff).

- There are at most c efficient solutions (Pareto optimal solutions) that can be computed in polynomial time. Each of these solutions corresponds to the fact that agent A wins in x rounds, with $0 \leq x \leq c$, and agent B wins the remaining $c - x$ rounds.
- There exist no Nash equilibria, except for trivial instances where only one Pareto optimum exists.
- The *best-worst* ratio, i.e. the ratio between the values of the global optimum and the sum of the agents' payoffs in any efficient solution, is no more than 2.

In case of Rule 2, when at each round the less profitable item wins, it is not obvious for the agents how to select the c items to submit.

- Also in this case, there is no preventive strategy for an agent and no Nash equilibria exist (except for trivial instances).
- However, differently from Rule 1, given an integer x , with $0 \leq x \leq c$, *exponentially many* Pareto optimal solutions may exist such that the number of winning rounds for agent A is equal to x .
- Given two arbitrary values Q_A and Q_B , it is NP-complete to find a solution M such that $p_A(M) \geq Q_A$ and $p_B(M) \geq Q_B$.
- Under Rule 2, the *best-worst* ratio, as defined above, can be arbitrarily high.

References

- [1] G. Aggarwal, J. D. Hartline. Knapsack auctions, Proceedings of the 17th annual ACM-SIAM Symposium on Discrete Algorithm, pp. 1083–1092, 2006.
- [2] M. Dell'Amico, S. Martello. The k -cardinality assignment problem, Discrete Applied Mathematics, 76, pp. 103–121, 1997.
- [3] M. Fujimoto, T. Yamada. An exact algorithm for the knapsack sharing problem with common items, European Journal of Operational Research, 171(2), pp. 693–707, 2006.
- [4] M. Hifi, H. M'Hallab, S. Sadfi. An exact algorithm for the knapsack sharing problem, Computers & Operations Research, 32(5), pp. 1311–1324, 2005.
- [5] V. Liberatore. Scheduling Jobs Before Shut Down, Lecture Notes in Computer Science, 1851, Algorithm Theory - SWAT 2000, pp. 461–466, Springer, 2000.
- [6] C. Nomikos, A. Pagourtzis, S. Zachos. Randomized and Approximation Algorithms for Blue-Red Matching. Lecture Notes in Computer Science 4708, pp. 715–725, Springer, 2007.

An exact method for the minimum caterpillar spanning problem

L. Simonetti,^a Y. Frota,^a C.C. de Souza^{a,??}

^aUniversidade Federal de Campinas, UNICAMP, Instituto de Computação, Caixa Postal 6176, 13084-971, Campinas, SP, Brazil, {luidi,yuri,cid}@ic.unicamp.br

Key words: caterpillar trees, combinatorial optimization, integer programming

Introduction. Let $G = (V, E)$ be an undirected graph where $V = \{v_1, \dots, v_n\}$ is the vertex set and E is the edge set. We assume that G is complete and associate to each edge (i, j) the costs c_{ij}^1 and c_{ij}^2 . A tree T in G is said to be a *caterpillar* if the subgraph remaining after removing all the leaves from T is a path. Vertices in this path are called *central*. The *Minimum Spanning Caterpillar Problem* (MSCP) consists in finding a caterpillar containing all the vertices of G whose cost is minimal. The cost of an edge (i, j) in the caterpillar is c_{ij}^2 if both its extremities are central vertices and c_{ij}^1 otherwise.

MSCP is \mathcal{NP} -hard [10] and applications of it are found in simplifications of complex real-world situations which, when considered in their full extent, are very difficult to deal with (e.g., [1], [8]). This includes problems arising in vehicle routing in hierarchical logistics, telecommunication networks design and fiber optics networks [2]. Not too much attention has been given to algorithms for the MSCP and, to our knowledge, no exact methods are available. However, the minimum ring-star problem (MRSP) is closely related to the MSCP and was investigated earlier (see [11; 7]). In fact, it can be shown that any algorithm that solves the MSCP also computes the MRSP and vice-versa. The relationship between the two problems is equivalent to the one existing between Hamiltonian paths and the traveling salesman problems.

In [5] good results were achieved by adapting an integer programming (IP) formulation for the minimum Steiner arborescence problem (MSAP) to a class of

¹ First author is supported by grant # 2008/01497-8 from *Fundação de Amparo à Pesquisa do Estado de São Paulo*, Brazil. Second author is supported by a scholarship from CAPES (Brazilian Ministry of Education). Third author is partially supported by *Conselho Nacional de Desenvolvimento Científico e Tecnológico*, Brazil, grants # 301732/2007-8 and # 472504/2007-0.

problems involving the computation of optimal trees in graphs. In the MSAP we are given a directed graph with costs associated to the edges, a special vertex r and a set R of terminal vertices. The goal is to find a minimum cost arborescence rooted at r and spanning all vertices in R . In this paper a similar idea is used to reduce the MSCP to the MSAP in a layered graph. This reduction is the basis for the development of an efficient branch-and-cut algorithm for the MSCP. In the sequel we formulate the problem as a Steiner problem and report on our numerical experiments.

IP model. We start by reducing the MSCP to the MSAP. To this end, we build a directed graph $G_N = (V_N, A_N)$ from the graph $G = (V, E)$ given at the input for the MSCP. The graph G_N has three layers numbered 0, 1 and 2. The first one is composed solely by the special vertex 0. Now, for every vertex $i \in V$, two vertices are created in V_N , namely, the vertex i_1 and the vertex i_2 in the second and third layers, respectively. To each edge (i, j) in E , two arcs (i_1, j_1) and (j_1, i_1) are created in A_N . The remaining arcs of A_N are of the form (i_1, i_2) and $(0, i_1)$, for all $i \in V$, and (i_1, j_2) , for all $(i, j) \in E$. As it can be seen, most of the arcs in G_N go from layer h to layer $h + 1$, $h \in \{0, 1\}$. As an abuse of language, we refer to G_N as a layered graph although the subgraph induced by the vertices in layer 1 is a complete digraph.

We mean to use the arcs with both extremities in layer 1 to identify the central vertices. Besides, the arcs from layer 1 to layer 2 are meant to identify the edges of E joining a central vertex to a leaf of the caterpillar. Therefore, the costs of the arcs in A_N are computed as follows. For every vertex $i \in V$, the costs of arcs $(0, i_1)$ and (i_1, i_2) are given by M and 0, respectively. The value of M is chosen to be large enough to ensure that any optimal solution for the MSAP defined over G_N contains exactly one arc leaving vertex 0. Now, given two distinct vertices i and j in V , the cost of the arc (i_1, j_2) is set to c_{ij}^1 while the costs of the arcs (i_1, j_1) and (j_1, i_1) are both set to c_{ij}^2 .

To cast the MSCP as an MSAP we have to define the root vertex and the set R of terminals. This is done by assigning vertex 0 to the root and all the vertices of V_N in layer 2 to the set R . In addition, side constraints are created requiring that at most one arc leaves a vertex in layer 1 to reach another vertex in this layer. Notice that these constraints are not present in the classical MSAP. Moreover, recall that, the high costs attributed to the arcs emanating from the root forces any optimal solution to have precisely one such arc. Together with the side constraints, this guarantees that the subgraph induced by the vertices of layer 1 in an optimal solution is a path.

We now turn our attention to the development of an IP model for the MSAP. Initially, we define the binary variables x_{uv} for each arc $(u, v) \in A_N$ and set it to one if and only if (u, v) belongs to the optimal Steiner arborescence. Then, the

MSAP for the graph G_N derived from G is formulated by the IP below:

$$\begin{aligned} \text{(StM)} \quad & \min \sum_{(u,v) \in A_N} c_{uv} x_{uv} \\ \text{s. t.} \quad & \sum_{j \in V - \{i\}} x_{i_1 j_1} \leq 1, \quad \forall i \in V \end{aligned} \quad (0.1)$$

$$\sum_{u \in V_N \setminus S} \sum_{v \in S} x_{uv} \geq 1, \quad \forall S \subset V_N \setminus \{0\}, S \cap R \neq \{\emptyset\} \quad (0.2)$$

$$x_{i_1 j_1} \in \{0, 1\}, x_{i_1 j_2} \geq 0, \quad \forall i \neq j, (i, j) \in E$$

$$x_{0j_1} \geq 0, \quad x_{j_1 j_2} \geq 0, \quad \forall j \in V$$

Constraints (0.1) forces the creation of a path joining the central vertices (those in layer 1). Constraints (0.2) are the Steiner cuts which ensure that the terminal vertices are spanned. Notice that in this formulation some of the integrality constraints have been relaxed. One can show that they are satisfied as long as we impose the integrality of the variables for arcs that are internal to layer 1. Besides the Steiner cuts in (0.2), our branch-and-cut algorithm also uses the 2-matching constraints discussed in [3] and given in Theorem 1 below.

Theorem 1. For $H \subset V$ and $T \subset \delta(H)$, inequality (0.3) is valid for $\text{conv}(\text{StM})$ if (i) $\{i, j\} \cap \{k, w\} = \emptyset, \forall (i, j)$ and $(k, w) \in T$; and (ii) $|T| \geq 3$ and odd.

$$\sum_{i \in H, j \in H, (i, j) \in E} x_{i_1 j_1} + \sum_{(i, j) \in T} x_{i_1 j_1} \leq \sum_{i \in H} x_{i_1 i_2} + \frac{|T| - 1}{2}. \quad (0.3)$$

Computational experiments. The StM model is the starting point for the development of our branch-and-cut algorithm. The code is implemented in C++ and uses XPRESS 2008 as the IP solver. All tests were carried out on an Intel Core2 Quad processor with 2.83GHz and 8Gb of RAM. A fast polynomial-time algorithm based on the minimum edge cut problem in graphs is used to separate the Steiner cuts (0.2). Moreover, the heuristic proposed in [4] was implemented to compute violated 2-matching inequalities from Theorem 1.

To test the algorithm, we modified 24 instances from TSPLIB 2.1 [9] with sizes ranging from 26 to 299 vertices. The edge costs were adapted from the original TSP instances through the following calculations. Let c_{ij} be the distance between vertices i and j in the TSP instance. The two costs assigned to edge (i, j) in the MSCP instance are given by: $c_{ij}^1 = \lceil (10 - \alpha)c_{ij} \rceil$ and $c_{ij}^2 = \lceil \alpha c_{ij} \rceil$ for $\alpha \in \{3, 5, 7, 9\}$. Since each TSP instance give rise to four MSCP instances, one for each value of α , in total, our benchmark is composed of 96 instances. Notice that, for higher values of α , the optimal solutions is expected to have many leaves while, for lower values, most vertices are likely to be central.

The experimental results are summarized in Table 1. The relative gaps displayed are computed by $(\text{UB} - \text{LB})/\text{LB}$, where UB and LB denote, respectively, the best upper and lower bounds achieved. The number of nodes explored during the enumeration and the total time spent to solve the instances are shown too. The data are also gathered by different values of α and instance sizes.

	All Avr	$\alpha = 3$		$\alpha = 5$		$\alpha = 7$		$\alpha = 9$		$ V < 100$		$ V < 200$		$ V < 300$	
		Max	Avr	Max	Avr	Max	Avr	Max	Avr	Max	Avr	Max	Avr	Max	Avr
gap (%)	0.11	0.85	0.31	0.32	0.10	0.33	0.04	0.03	0.00	0.33	0.05	0.85	0.15	0.41	0.08
nodes	90.9	1483	345.4	745	43.5	13	2.6	1	1	39	3.7	1483	120.6	1433	185.8
time(s)	1008	25893	1616	25387	1532	7025	369	8810	529	22	2.5	1602	132	25893	6564

Table 1. Summary of computational results.

The results revealed that our algorithm is capable to solve to optimality MSCP instances with up to 300 nodes in reasonable time. All those with at most 200 vertices were computed is less than 30 minutes. The linear relaxation at the root node contributes for this success, providing very dual bounds in all cases. As a matter of fact, 42 instances were solved at the root node. The strength of the linear relaxations can also be assessed by the small number of nodes explored by the enumeration. One can see that the algorithm performs better for larger values of α , when more vertices are expected to be leaves.

References

- [1] G. S. Adhar. Optimum interval routing in k -caterpillars and maximal outer planar networks. In *Applied Informatics*, pages 692–696, 2003.
- [2] R. Baldacci, M. Dell’Amico and J.J. Salazar. The Capacitated m -Ring Star Problem. *Operations Research*, 55:1147–1162, 2007.
- [3] P. Bauer. The circuit polytope: Facets. *Oper Res*, 22:110-145, 1997.
- [4] M. Fischetti, J.J. Salazar and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Oper Res*, 45:378-394, 1997.
- [5] L. Gouveia, L. Simonetti and E. Uchoa. Modelling the hop-constrained minimum spanning tree problem over a layered graph. In *International Network Optimization Conference*, 2007.
- [6] E. A. Hoshino and C. C. de Souza. Column Generation Algorithms for the Capacitated m -Ring-Star Problem. *Lecture Notes in Computer Science*, vol. 5092, pages 631–641, 2008.
- [7] M. Labbé, G. Laporte, I.R. Martín and J.S. González. The Ring Star Problem: Polyhedral Analysis and Exact Algorithm. *Networks*, 43:177–189, 2004.
- [8] A. Proskurowski and J. A. Telle. Classes of graphs with restricted interval models. *Disc. Math. and Theoretical Computer Science*, 3(4):167–176, 1999.
- [9] G. Reinelt. A traveling salesman problem library. *ORSA J Comp*, 3:376–384, 1991.
- [10] J. Tan and L. Zhang. Approximation Algorithms for the Consecutive Ones Submatrix Problem on Sparse Matrices. *Lecture Notes in Computer Science*, vol. 3341, pages 835–846, 2004.
- [11] J. Xu, S.Y. Chiu and F. Glover. Optimizing a ring-based private line telecommunication network using tabu search. *Manag Sci*, 45:330–345, 1999.

Coloring I

Lecture Hall B

Tue 2, 10:30–12:30

NP-completeness of determining the total chromatic number of graphs that do not contain a cycle with a unique chord

Raphael Machado,^{a,b} Celina de Figueiredo^a

^a*COPPE - Universidade Federal do Rio de Janeiro*

^b*Instituto Nacional de Metrologia, Normalização e Qualidade Industrial*

Abstract

The total chromatic number of a graph is the least number of colours sufficient to colour the elements (vertices and edges) of this graph in such a way that no incident or adjacent elements are coloured the same. The class \mathcal{C} of graphs that do not contain a cycle with a unique chord was recently studied by Trotignon and Vušković [5], who proved strong structure results for these graphs. In the same work, the authors determine, for the class \mathcal{C} , the complexity of vertex-colouring problem (polynomial), maximum clique problem (polynomial) and maximum stable set problem (NP-complete). The edge-colouring problem is NP-complete [3] when restricted to \mathcal{C} . In the present work, we show that also the total-colouring problem is NP-complete when restricted to \mathcal{C} .

Key words: total chromatic number, cycle with a unique chord, regular graphs

1. Introduction

Let $G = (V, E)$ be a simple graph. The maximum degree of a vertex in G is denoted $\Delta(G)$. A *total-colouring* of G is a function $\pi : V \cup E \rightarrow \mathbf{C}$ such that no two incident or adjacent elements receive the same colour $c \in \mathbf{C}$. If $\mathbf{C} = \{1, 2, \dots, k\}$, we say that π is a *k-total-colouring*. The *total chromatic number* of G , denoted by $\chi_T(G)$, is the least k for which G has a *k-total-colouring*. The Total Colouring Conjecture [1; 6], which states that every graph G is $(\Delta(G) + 2)$ -total colourable, is open since 1964.

The class \mathcal{C} of graphs that do not contain a cycle with a unique chord was first investigated by Trotignon and Vušković [5], who proved strong structure results for these graphs. Class \mathcal{C} is of interest also because it is an example of a χ -bounded class, that is, there exists a function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that, for each $G \in \mathcal{C}$, $\chi(G) \leq f(\Delta(G))$.

$f(\omega(G))$, where $\chi(G)$ is the (vertex-) chromatic number of G and $\omega(G)$ is the clique-number of G . In the present work, we prove that the total-colouring problem restricted to \mathcal{C} is NP-complete. Additionally, we propose the study of the Total Colouring Conjecture restricted to graphs in \mathcal{C} .

2. NP-completeness result

The term $\text{TOTCHR}(P)$ (resp. $\text{CHRIND}(P)$) denotes the problem of determining the total chromatic number (resp. chromatic index) restricted to graph inputs with property P . For example, $\text{TOTCHR}(\text{graph of } \mathcal{C})$ (resp. $\text{CHRIND}(\text{graph of } \mathcal{C})$) denotes the problem whose instance is a graph G of \mathcal{C} and that questions whether $\chi_T(G) = \Delta(G) + 1$ (resp. $\chi'(G) = \Delta(G)$). The problem $\text{TOTCHR}(\Delta\text{-regular bipartite graph})$ is NP-complete [4] for each $\Delta \geq 3$. The problem $\text{CHRIND}(\Delta\text{-regular graph})$ is NP-complete [2] for each $\Delta \geq 3$.

The NP-completeness gadget used in [4] has cycles with unique chords. The goal of our proposed NP-completeness proof is to modify the gadget used in [4] in order to have a gadget in \mathcal{C} . The proposed modification leads to a non-regular gadget. In order to obtain an NP-completeness result for regular graphs, we present, in Theorem 2 a novel technique based on an induction on the minimum degree of a graph.

Graph S_t , $t \geq 3$, is used in the present work exactly as defined in [4]: S_t is obtained from the complete bipartite graph $K_{t-1,t}$ by adding t pendant edges to the t vertices of degree $t-1$. We generalize the graph H_t of [4] by defining graph $H_{n,t}$, $t > n \geq 1$: $H_{2,t} = H_t$, and, more generally, $H_{n,t}$ is constructed by putting together two copies of S_t and identifying $t-n$ pendant edges of the first copy with $t-n$ edges of the second copy. (See S_t and $H_{n,t}$ in Figure 1 at Appendix A.)

The original ‘‘replacement’’ graph R of [4] contains cycles with unique chords. We modify and extend R to a family R_t , $t \geq 3$, of ‘‘replacement’’ graphs in \mathcal{C} . Take $t+1$ copies of $H_{n,t}$, with $n = \lceil (t+1)/2 \rceil$, and denote these copies by $H^{(1)}, H^{(2)}, \dots, H^{(t+1)}$. The ‘‘replacement’’ graph R_t is such that each copy of $H_{n,t}$ in R_t has one pendant edge – which is called *real* – or two pendant edges – one of which is called *real*. For, identify each of t pendant vertices of $H^{(i)}$, $i = 1, 2, \dots, t+1$, with a distinct $H^{(j)}$, $j \neq i$, by choosing one of the pendant vertices of $H^{(j)}$ (see R_3 and R_4 in Figure 2 at Appendix A).

Lemma 1. Let π be a partial $(t+1)$ -total-colouring of R_t , $t \geq 3$, in which the $t+1$ real pendant edges have different colours and the pendant vertices of the $t+1$ real pendant edges are also coloured (and nothing else is coloured). Then π extends to a $(t+1)$ -total-colouring of R_t . Moreover, in any $(t+1)$ -total-colouring of R_t the $t+1$ real pendant edges have all different colours.

The “forcer” graph $F_{n,t}$, $t > n \geq 2$, is used exactly as in [4]. Graph $F_{n,t}$ is constructed by linking n copies of the graph $H_{2,t}$ (see Figure 3 at Appendix A).

Lemma 2. (McDiarmid and Sánchez-Arroyo [4]) Let π be a partial $(t + 1)$ -total-colouring of $F_{n,t}$, $t > n \geq 2$, in which the pendant edges have the same colour and the pendant vertices are also coloured (and nothing else is coloured). Then π extends to a $(t + 1)$ -total-colouring of $F_{n,t}$. Moreover, in any $(t + 1)$ -total-colouring of $F_{n,t}$ the pendant edges have all the same colour.

Theorem 2 proves the NP-completeness of total-colouring Δ -regular graphs that do not contain a cycle with a unique chord, for each fixed degree $\Delta \geq 3$. Before proving Theorem 2 for regular graphs, we prove a weaker result: Lemma 3 proves the NP-completeness of problem $P_{\Delta,\delta} = \text{TOTCHR}(\text{graph in } \mathcal{C} \text{ with maximum degree } \Delta, \text{ minimum degree } \geq \delta, \text{ and such that every edge is incident to a maximum-degree vertex})$ for $\delta = 1$. Theorem 2 obtains a regular graph based on a novel strategy of induction on the minimum degree.

Lemma 3. For each $\Delta \geq 3$, $P_{\Delta,1}$ is NP-complete.

Proof (Sketch). Let G be an instance of the NP-complete problem CHRIND (Δ -regular graph). We construct an instance G' of problem $P_{\Delta,1}$ satisfying that G' is $(\Delta + 1)$ -total-colourable if and only if G is Δ -edge-colourable. The construction of graph G' is carried out with the following procedure (see Figure 4 at Appendix A for an example where $G = K_4$). Construct a graph L by replacing each vertex v of G with a copy of R_Δ . Two different copies of the “replacement” graph have pendant edges identified according to the adjacencies in the original graph G . Observe that L has $|V(G)|$ pendant edges if Δ is odd – each of them is real – and $(\Delta + 2)|V(G)|$ pendant edges if Δ is even – $|V(G)|$ of which are real and $(\Delta + 1)|V(G)|$ of which are not real. Construct G' by identifying the $|V(G)|$ real pendant edges of L with $|V(G)|$ pendant edges of a forcer graph $F_{\lfloor |V(G)|/2 \rfloor, \Delta}$. \square

Theorem 2. For each $\Delta \geq 3$, $\text{TOTCHR}(\Delta\text{-regular graph in } \mathcal{C})$ is NP-complete.

Proof (Sketch). By Lemma 3, the problem $P_{\Delta,1}$ is NP-complete. Assume, as induction hypothesis, that the problem $P_{\Delta,k}$, $k < \Delta$, is NP-complete. We prove the theorem by induction on k . Let G be an instance of the problem $P_{\Delta,k}$ and construct an instance G' of problem $P_{\Delta,k+1}$ as follows. Let G_1 and G_2 be two graphs isomorphic to G . Let H_1, \dots, H_x be as many graphs isomorphic to $H_{1,\Delta}$ as there are degree- k vertices in G . We prove that there is a $(\Delta + 1)$ -total-colouring of $H_{1,\Delta}$ such that its two pendant vertices receive the same colour and its two pendant edges receive the same colour. Denote the degree- k vertices of G_1 (resp. G_2) by v_1, \dots, v_x (resp. by w_1, \dots, w_x). Construct G' by taking graphs G_1 and G_2 and, for each H_i , identifying one pendant vertex with v_i and the other pendant vertex with w_i . Graph G' belongs to \mathcal{C} , has maximum degree Δ , minimum degree $\geq k + 1$, and every edge is incident to a maximum-degree vertex. Moreover, G' is $(\Delta + 1)$ -total-colourable if and only if G is $(\Delta + 1)$ -total-colourable. So, $P_{\Delta,k+1}$ is NP-complete and the

theorem follows by induction. \square

Remark our proposed inductive strategy is not used in [4]. The gadgets constructed in [4] are regular, while the proposed gadgets in \mathcal{C} are not.

3. Final remarks and current work

We consider the total-colouring problem restricted to \mathcal{C} . We propose an inductive strategy for NP-completeness proofs that may be applied to general regular graphs. At the moment we investigate two additional problems on total-colouring. First, we investigate whether it is possible to extend the NP-completeness proof of the present work to bipartite graphs that do not contain a cycle with a unique chord. Second, we investigate the validity of the Total Colouring Conjecture in \mathcal{C} : the upper bound $\chi_T(G) \leq \Delta(G) + 4$ follows from the results of [5].

References

- [1] M. Behzad. *Graphs and their chromatic numbers*. Ph.D. Thesis. Michigan State University (1965).
- [2] D. Leven and Z. Galil. *NP-completeness of finding the chromatic index of regular graphs*. *J. Algorithms* **4** (1983) 35–44.
- [3] R. C. S. Machado, C. M. H. de Figueiredo, and K. Vušković. *Edge-colouring graphs with no cycle with a unique chord*. *Proc. of Alio/Euro Workshop* (2008).
- [4] C. J. H. McDiarmid, A. Sánchez-Arroyo. *Total colouring regular bipartite graphs is NP-hard*. *Discrete Math.* **124** (1994) 155-162.
- [5] N. Trotignon and K. Vušković. *A structure theorem for graphs with no cycle with a unique chord and its consequences*. To appear in *J. Graph Theory*.
- [6] V. G. Vizing. *On an estimate of the chromatic class of a p -graph*. *Metody Diskret. Analiz.* **3** (1964) 25–30. In Russian.

Figures

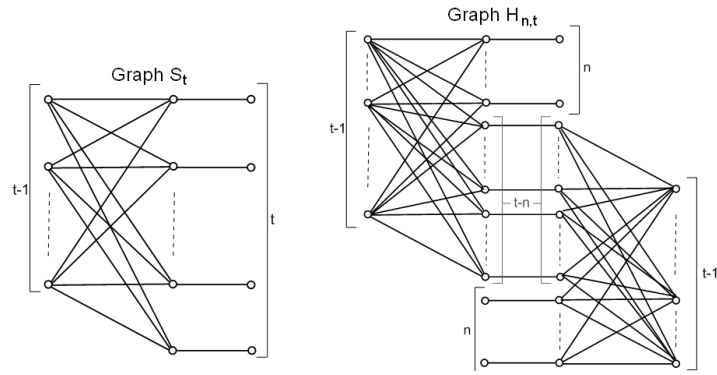


Fig. 1. Graphs S_t (left) and $H_{n,t}$ (right).

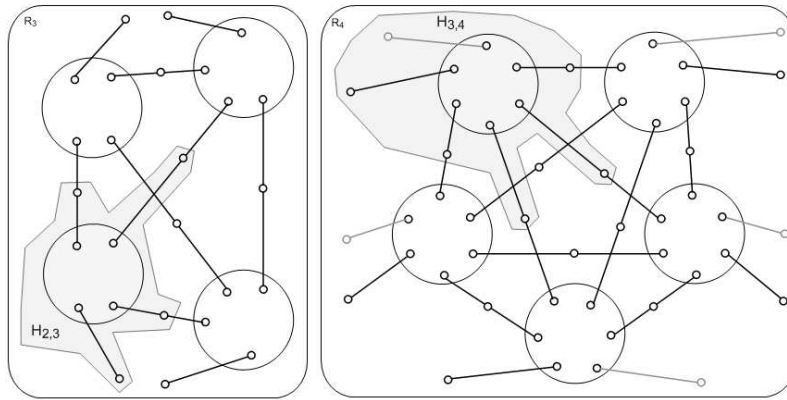


Fig. 2. Graphs R_3 (left) and R_4 (right).

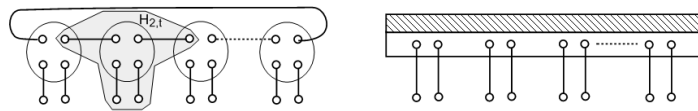


Fig. 3. The forcer graph and its schematic representation.

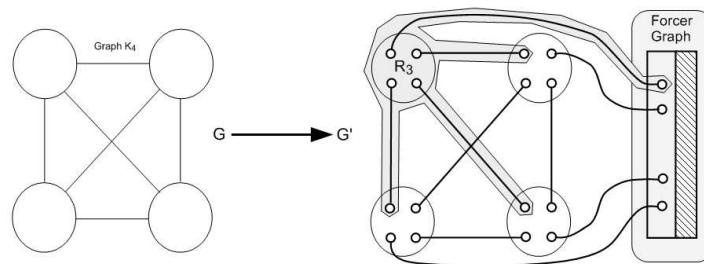


Fig. 4. Construction of G' for the proof of Lemma 3, in the case where $G = K_4$.

Acyclic and frugal colourings of graphs

Ross J. Kang,^{a,1} Tobias Müller^b

^a*School of Computer Science, McGill University, 3480 University Street, Montréal, Québec, H2A 2A7, Canada.*

^b*School of Mathematical Sciences, Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel.*

Key words: graph colouring, frugal, acyclic, improper

1. Introduction

In this paper, a (*vertex*) *colouring* of a graph $G = (V, E)$ is any map $f : V \rightarrow \mathbb{Z}^+$. The *colour classes* of a colouring f are the preimages $f^{-1}(i), i \in \mathbb{Z}^+$. A colouring of a graph is *proper* if adjacent vertices receive distinct colours; however, in this paper, we will devote considerable attention to colourings that are not necessarily proper, but that satisfy another condition. A colouring of G is *t-frugal* if no colour appears more than t times in any neighbourhood. The notion of frugal colouring was introduced by Hind, Molloy and Reed [5]. They considered proper t -frugal colourings as a way to improve bounds related to the Total Colouring Conjecture (cf. [6]). In Section 2, we study t -frugal colourings for graphs of bounded maximum degree.

In Section 3, we impose an additional condition that is well-studied in the graph colouring literature (cf. [3]). A colouring of V is *acyclic* if each of the bipartite graphs consisting of the edges between any two colour classes is acyclic. In other words, a colouring of G is acyclic if G contains no *alternating cycle* (that is, an even cycle that alternates between two distinct colours). For graphs of bounded maximum degree, the study of acyclic proper colourings was instigated by Erdős (cf. [2]) and more or less settled asymptotically by Alon, McDiarmid and Reed [3]. Extending the work of Alon *et al.*, Yuster [9] investigated acyclic proper 2-frugal colourings. In Section 3, we expand this study to different values of t and colourings that are not necessarily proper.

¹ Part of this research was done while this author was a doctoral student at Oxford University. He was partially supported by NSERC of Canada and the Commonwealth Scholarship Commission in the UK.

Let us outline our notation. As usual, the *chromatic number* $\chi(G)$ (resp. *acyclic chromatic number* $\chi_a(G)$) denotes the least number of colours needed in a proper (resp. acyclic proper) colouring. Analogously, for $t \geq 1$, we define the *t-frugal chromatic number* $\varphi^t(G)$, *proper t-frugal chromatic number* $\chi_\varphi^t(G)$, *acyclic t-frugal chromatic number* $\varphi_a^t(G)$ and *acyclic proper t-frugal chromatic number* $\chi_{\varphi,a}^t(G)$. We have designated φ as a mnemonic for frugal. We are interested in graphs G of bounded degree, so let $\chi(d)$ denote the maximum possible value of $\chi(G)$ over all graphs G with $\Delta(G) = d$. We analogously define $\chi_a(d)$; $\varphi^t(d)$, $\chi_\varphi^t(d)$, $\varphi_a^t(d)$ and $\chi_{\varphi,a}^t(d)$ for $t \geq 1$. The *square* of a graph G , i.e. the graph formed from G by adding edges between any two vertices at distance two, is denoted G^2 . Note the following basic observations.

Proposition 1. For any graph G and any $t \geq 1$, the following hold:

- (i) $\chi_\varphi^1(G) = \chi_{\varphi,a}^1(G) = \chi(G^2)$;
- (ii) $\varphi^t(G) \leq \chi_\varphi^t(G)$, $\varphi_a^t(G) \leq \chi_{\varphi,a}^t(G)$; $\varphi^t(G) \leq \varphi_a^t(G)$, $\chi_\varphi^t(G) \leq \chi_{\varphi,a}^t(G)$;
- (iii) $\varphi^{t+1}(G) \leq \varphi^t(G)$, $\chi_{\varphi,a}^{t+1}(G) \leq \chi_\varphi^t(G)$, $\varphi_a^{t+1}(G) \leq \varphi_a^t(G)$, $\chi_{\varphi,a}^{t+1}(G) \leq \chi_{\varphi,a}^t(G)$;
- and
- (iv) $\varphi^t(G) \geq \Delta(G)/t$.

We may invoke basic probabilistic tools such as the Lovász Local Lemma, details of which can be found in various references, e.g. Molloy and Reed [7].

2. Frugal colourings

As a way to improve bounds for total colouring (cf. [6]), Hind *et al.* [5], showed that $\chi_\varphi^{(\ln d)^5}(d) \leq d + 1$ for sufficiently large d . Recently, this was improved.

Theorem 1. (Molloy and Reed [8]) $\chi_\varphi^{50 \ln d / \ln \ln d}(d) \leq d + 1$ for sufficiently large d .

Since $\chi_\varphi^t(K_{d+1}) \geq d + 1$, it follows that $\chi_\varphi^t(d) = d + 1$ for $t = t(d) \geq 50 \ln d / \ln \ln d$. For smaller frugalities, Hind *et al.* [5] also showed the following.

Theorem 2. (Hind *et al.* [5]) For any $t \geq 1$ and sufficiently large d , $\chi_\varphi^t(d) \leq \max \left\{ (t+1)d, \left\lceil e^3 d^{1+1/t} / t \right\rceil \right\}$.

By Proposition 1(i), $\chi_\varphi^1(d) \sim d^2$. We note that an example based on projective geometries due to Alon (cf. [5]), to lower bound $\chi_\varphi^t(d)$, is also valid for $\varphi^t(d)$.

Proposition 2. For any $t \geq 1$ and any prime power n , $\varphi^t(n^t + \dots + 1) \geq (n^{t+1} + \dots + 1)/t$.

The following consequence shows (by Proposition 1(ii)) that Theorem 2 is asymptotically tight up to a constant multiple when $t = o(\ln d / \ln \ln d)$.

Corollary 1. Suppose that $t = t(d) \geq 2$, $t = o(\ln d / \ln \ln d)$, and $\epsilon > 0$ fixed. Then, for sufficiently large d , $\varphi^t(d) \geq (1 - \epsilon)d^{1+1/t}/t$.

Theorems 1 and 2 determine the behaviour of $\chi_\varphi^t(d)$ up to a constant multiple for all t except for the range such that $t = \Omega(\ln d / \ln \ln d)$ and $t \leq 50 \ln d / \ln \ln d$. Recall from Proposition 1(iv) that $\varphi^t(d) \geq d/t$. For the case $t = \omega(\ln d)$, we give a tight upper bound for $\varphi^t(d)$.

Theorem 3. Suppose $t = \omega(\ln d)$ and $\epsilon > 0$ fixed. Then, for sufficiently large d , $\varphi^t(d) \leq \lceil (1 + \epsilon)d/t \rceil$.

proof 1. Let $G = (V, E)$ be a graph with maximum degree d and let $x = \lceil (1 + \epsilon)d/t \rceil$. Let f be a random colouring where for each $v \in V$, $f(v)$ is chosen uniformly and independently at random from $\{1, \dots, x\}$. For a vertex v and a colour $i \in \{1, \dots, x\}$, let $A_{v,i}$ be the event that v has more than t neighbours with colour i . If none of these events hold, then f is t -frugal. Each event is independent of all but at most $d^2 x \ll d^3$ others. By a Chernoff bound,

$$\begin{aligned} \Pr(A_{v,i}) &= \Pr(\text{BIN}(d, 1/x) > t) \leq \Pr(\text{BIN}(d, 1/x) > d/x + ct) \\ &\leq \exp\left(-c^2 t^2 / (2d/x + 2ct/3)\right) \end{aligned}$$

where $c = \epsilon/(1 + \epsilon)$. Thus, $e \Pr(A_{v,i}) (d^3 + 1) = \exp(-\Omega(t))d^3 < 1$ for large enough d , and by the Lovász Local Lemma, f is t -frugal with positive probability for large enough d .

3. Acyclic frugal colourings

Using the Lovász Local Lemma, Alon *et al.* [3] established a $o(d^2)$ upper bound for $\chi_a(d)$, answering a long-standing question of Erdős (cf. [2]). Using a probabilistic construction, they also showed this upper bound to be asymptotically correct up to a logarithmic multiple.

Theorem 4. (Alon *et al.* [3]) $\chi_a(d) \leq \lceil 50d^{4/3} \rceil$, $\chi_a(d) = \Omega(d^{4/3}/(\ln d)^{1/3})$.

Yuster [9] considered acyclic proper 2-frugal colourings of graphs and showed that $\chi_{\varphi,a}^2(d) \leq \lceil \max\{50d^{4/3}, 10d^{3/2}\} \rceil$. For acyclic frugal colourings, we first consider the smallest cases then proceed to larger values of t . For $t = 1, 2, 3$, notice that Corollary 1, Proposition 1(i) and Yuster's result imply that $\varphi_a^1(d) = \Theta(d^2)$, $\varphi_a^2(d) = \Theta(d^{3/2})$, $\chi_{\varphi,a}^2(d) = \Theta(d^{3/2})$ and $\varphi_a^3(d) = \Omega(d^{4/3})$. Next, we show that $\chi_{\varphi,a}^3(d) = O(d^{4/3})$. This implies that $\chi_{\varphi,a}^t(d) = O(d^{4/3})$ for any $t \geq 3$, a bound that

is within a logarithmic multiple of the lower bound implied by Theorem 4. This answers a question of Esperet, Montassier and Raspaud [4].

Theorem 5. $\chi_{\varphi,a}^3(d) \leq \lceil 40.27d^{4/3} \rceil$.

proof 2. (Outline.) Our proof is an extension of the proof of Theorem 4 in which we add a fifth event to ensure that the random colouring f is 3-frugal:

V For vertices v, v_1, v_2, v_3, v_4 with $\{v_1, v_2, v_3, v_4\} \subseteq N(v)$, let $E_{\{v_1, \dots, v_4\}}$ be the event that $f(v_1) = f(v_2) = f(v_3) = f(v_4)$.

For acyclic frugal colourings which are not necessarily proper, for larger values of t , we have adapted a result of Addario-Berry *et al.* [1] to show the following.

Theorem 6. For any $t = t(d) \geq 1$, $\varphi_a^t(d) = O(d \ln d + (d - t)d)$.

This implies, for instance, that $\varphi_a^{d-1}(d)$ and $\chi_{\varphi,a}^{d-1}(d)$ differ by a multiplicative factor of order at least $d^{1/3}/(\ln d)^{4/3}$. The result is obtained by studying *total k -dominating sets* — given $G = (V, E)$, $\mathcal{D} \subset V$ is total k -dominating if each vertex has at least k neighbours in \mathcal{D} .

References

- [1] Addario-Berry, L., Esperet, L., Kang, R. J., McDiarmid, C. J. H., and Pinlou, A. *Acyclic t -improper colourings of graphs with bounded maximum degree*, 2009+. To appear in *Discrete Mathematics*.
- [2] Albertson, M. O. and Berman, D. M. *The acyclic chromatic number*, Proceedings of the Seventh Southeastern Conference on Combinatorics, Graph Theory, and Computing (Louisiana State Univ., Baton Rouge, La., 1976), 1976, pages 51–69.
- [3] Alon, N., McDiarmid, C. J. H., and Reed, B. *Acyclic coloring of graphs*. Random Structures and Algorithms, **2** (1991), no. 3, pages 277–288.
- [4] Esperet, L., Montassier, M., and Raspaud, A. *Linear choosability of graphs*. Discrete Math., **308** (2008), no. 17, pages 3938–3950.
- [5] Hind, H., Molloy, M., and Reed, B. *Colouring a graph frugally*. Combinatorica, **17** (1997), no. 4, pages 469–482.
- [6] Hind, H., Molloy, M., and Reed, B. *Total coloring with $\Delta + \text{poly}(\log \Delta)$ colors*. SIAM J. Comput., **28** (1999), no. 3, pages 816–821 (electronic).
- [7] Molloy, M. and Reed, B. *Graph Colouring and the Probabilistic Method*. Algorithms and Combinatorics, **23**, Springer-Verlag, Berlin, 2002.
- [8] Molloy, M. and Reed, B. *Asymptotically optimal frugal colouring*. SODA '09: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2009, pages 106–114.
- [9] Yuster, R. *Linear coloring of graphs*. Discrete Math., **185** (1998), no. 1-3, pages 293–297.

On resistance of graphs

P.A. Petrosyan,^{a,b} H.E. Sargsyan^b

^a*Institute for Informatics and Automation Problems of NAS of RA*
pet_petros@{ipia.sci.am, ysu.am, yahoo.com}

^b*Department of Informatics and Applied Mathematics, YSU*
sargsyanhasmik@gmail.com

Key words: edge coloring, interval coloring, resistance of a graph

1. Introduction

The graph coloring problems play a crucial role in Discrete Mathematics. The reason for that is the fact of existence of many problems in Discrete Mathematics which can be formulated as graph coloring problems (factorization problems, problems of Ramsey theory, etc.), the tight relationship between graph coloring problems and scheduling of various timetables. For example, the problem of constructing an optimal schedule for an examination session can be reduced to the problem of finding the chromatic number of a graph. On the other hand, the sport scheduling problems can be reduced to the problem of finding the chromatic index of a graph, etc..

One of the aspects of the problems of scheduling theory is the construction of timetables without “gaps”. For studying the coloring problems corresponding to ones of constructing a timetable without a “gap”, a definition of an interval coloring of a graph was introduced [1]. Many bipartite graphs such as regular bipartite graphs [1; 2], trees, complete bipartite graphs [11], subcubic bipartite graphs [8], doubly convex bipartite graphs [3], grids [5], outerplanar bipartite graphs [7], $(2, \Delta)$ – biregular bipartite graphs [9; 13; 15] and some classes of $(3, 4)$ – biregular bipartite graphs [4; 16] have interval colorings. Unfortunately, it is known that not all graphs have interval colorings, therefore, it is expedient to consider a measure of closeness for a graph to be interval colorable. First attempt to introduce such a measure was done in [6]. The deficiency [6] of a graph is the minimum number of pendant edges whose attachment to the graph makes the resulting graph interval colorable.

In this work we introduce a new measure of closeness for a graph to be interval

colorable. We call it a resistance. More precisely, the resistance of a graph is the minimum number of edges that should be removed from a given graph to obtain an interval colorable graph.

2. Main results

All graphs considered in this work are finite, undirected and have no loops or multiple edges. Let $V(G)$ and $E(G)$ denote the sets of vertices and edges of G , respectively. An edge coloring of a graph G with colors $1, 2, \dots, t$ is called an interval t -coloring if at least one edge of G is colored by $i, i = 1, 2, \dots, t$, the colors of edges incident to each vertex of G are distinct and form an interval of integers. The set of all interval colorable graphs is denoted by \mathfrak{N} [1; 12].

We define the resistance of a graph G ($res(G)$) in the following way:

$$res(G) \equiv \min_{G \setminus F \in \mathfrak{N}} |F|.$$

Clearly, $0 \leq res(G) \leq |E(G)| - 1$ for every graph G , and $res(G) = 0$ iff $G \in \mathfrak{N}$.

First, we give some general facts on resistance of graphs.

Proposition 1. Let G be a connected graph with $|V(G)| = p$, $|E(G)| = q$. Then

$$res(G) \leq q - p + 1.$$

Proposition 2. Let G be an r -regular graph with an odd number of vertices. Then

$$res(G) \geq \frac{r}{2}.$$

Proposition 3. For any $k \in \mathbb{N}$ there is a graph G such that $G \notin \mathfrak{N}$ and $res(G) = k$.

A.S. Asratian and R.R. Kamalian [1] proved that the problem “Does a given regular graph is interval colorable or not?” is NP -complete. This immediately implies the following result:

Proposition 4. Let G be an r -regular ($r \geq 3$) graph and k is a nonnegative integer. Then the problem of deciding $res(G) \leq k$ is NP -complete.

In [18] S.V. Sevast'janov showed that it is an NP -complete problem to decide whether a bipartite graph has an interval coloring. From here we have the following result:

Proposition 5. Let G be a bipartite graph and k is a nonnegative integer. Then the problem of deciding $res(G) \leq k$ is NP -complete.

Next, we determine the exact value of $res(G)$ for simple cycles, wheels, complete graphs, Schwartz's graphs [17] and we obtain upper bounds for $res(G)$ in case of complete balanced k -partite graphs [14] and Hertz's graphs [10; 6].

Proposition 6. For any $n \geq 3$

$$res(C_n) = \begin{cases} 0, & \text{if } n \text{ is even,} \\ 1, & \text{if } n \text{ is odd.} \end{cases}$$

Theorem 1. For any $n \geq 4$

$$res(W_n) = \begin{cases} 0, & \text{if } n = 4, 7, 10, \\ 1, & \text{otherwise.} \end{cases}$$

Theorem 2. For any $n \in N$

$$res(K_n) = \begin{cases} 0, & \text{if } n \text{ is even,} \\ \lfloor \frac{n}{2} \rfloor, & \text{if } n \text{ is odd.} \end{cases}$$

Theorem 3. For any $n, k \in N$

- (1) $res(K_{n,n,\dots,n}) = 0$, if nk is even,
- (2) $\frac{(k-1)n}{2} \leq res(K_{n,n,\dots,n}) \leq \frac{(k-1)n^2}{2}$, if nk is odd.

Theorem 4. For any odd $k \geq 3$

$$res(S(k)) = k - 1.$$

Theorem 5. For any $p \geq 4, q \geq 3$

- (1) $res(H_{p,q}) = 0$, if $p \leq \lfloor \frac{2(q+1)}{q-1} \rfloor$
- (2) $res(H_{p,q}) \leq p - \lfloor \frac{2(q+1)}{q-1} \rfloor$, if $p > \lfloor \frac{2(q+1)}{q-1} \rfloor$.

References

- [1] A.S. Asratian, R.R. Kamalian, Interval colorings of edges of a multigraph, Appl. Math. 5 (1987) 25-34 (in Russian).

- [2] A.S. Asratian, R.R. Kamalian, Investigation on interval edge-colorings of graphs, *J. Combin. Theory Ser. B* 62 (1994) 34-43.
- [3] A.S. Asratian, T.M.J. Denley, R. Haggkvist, *Bipartite Graphs and their Applications*, Cambridge University Press, Cambridge, 1998.
- [4] A.S. Asratian, C.J. Casselgren, J. Vandenbussche, D.B. West, Proper path-factors and interval edge-coloring of $(3, 4)$ -biregular bigraphs, *Journal of Graph Theory*, 2009, to appear (arXiv:0704.2650v1, math.co, 2007).
- [5] K. Giaro, M. Kubale, Consecutive edge-colorings of complete and incomplete Cartesian products of graphs, *Congr. Numer.* 128 (1997) 143-149.
- [6] K. Giaro, M. Kubale, M. Malafiejski, On the deficiency of bipartite graphs, *Discrete Appl. Math.* 94 (1999) 193-203.
- [7] K. Giaro, M. Kubale, Compact scheduling of zero-one time operations in multi-stage systems, *Discrete Appl. Math.* 145 (2004) 95-103.
- [8] H.M. Hansen, Scheduling with minimum waiting periods, Master's Thesis, Odense University, Odense, Denmark, 1992 (in Danish).
- [9] D. Hanson, C.O.M. Loten, B. Toft, On interval colorings of bi-regular bipartite graphs, *Ars Combin.* 50 (1998) 23-32.
- [10] A. Hertz, unpublished result, 1991.
- [11] R.R. Kamalian, Interval colorings of complete bipartite graphs and trees, preprint, *Comp. Cen. of Acad. Sci. of Armenian SSR*, Erevan, 1989 (in Russian).
- [12] R.R. Kamalian, Interval edge colorings of graphs, Doctoral Thesis, Novosibirsk, 1990.
- [13] R.R. Kamalian, A.N. Mirumian, Interval edge colorings of bipartite graphs of some class, *Dokl. NAN RA*, 97 (1997) 3-5 (in Russian).
- [14] R.R. Kamalian, P.A. Petrosyan, On interval colorings of complete k -partite graphs K_n^k , *Math. probl. of comp. sci.* 25 (2006) 28-32.
- [15] A.V. Kostochka, unpublished manuscript, 1995.
- [16] A.V. Pyatkin, Interval coloring of $(3, 4)$ -biregular bipartite graphs having large cubic subgraphs, *J. Graph Theory* 47 (2004) 122-128.
- [17] A. Schwartz, The deficiency of a regular graph, *Discrete Math.* 306 (2006) 1947-1954.
- [18] S.V. Sevast'janov, Interval colorability of the edges of a bipartite graph, *Metody Diskret. Analiza* 50 (1990) 61-72 (in Russian).

Colored Resource Allocation Games¹

Evangelos Bampas,^a Aris Pagourtzis,^a George Pierrakos,^b
Vasileios Syrgkanis^a

^a*National Technical University of Athens, School of Elec. & Comp. Engineering*
{ebamp, pagour}@cs.ntua.gr, syrganis@corelab.ntua.gr

^b*School of Electrical Engineering & Computer Sciences, UC Berkeley*
georgios@cs.berkeley.edu

Potential Games are a widely used tool for modeling network optimization problems under a non-cooperative perspective. Initially studied in [18] with the introduction of congestion games and further extended in [15] in a more general framework, they have been successfully applied to describe selfish routing in communication networks (e.g. [19]). The advent of optical networks as the technology of choice for surface communication has introduced new aspects of networks that are not sufficiently captured by the models proposed so far. This work comes to close this gap and presents a class of potential games useful for modeling selfish routing and wavelength assignment in multifiber optical networks.

In optical networking it is highly desirable that all communication be carried out *transparently*, that is, each signal should remain on the same wavelength from source to destination. The need for efficient access to the optical bandwidth has given rise to the study of several optimization problems in the past years. The most well-studied among them is the problem of assigning a path and a color (wavelength) to each communication request in such a way that paths of the same color are edge-disjoint and the number of colors used is minimized. Nonetheless, it has become clear that the number of wavelengths in commercially available fibers is rather limited—and will probably remain such in the foreseeable future. Therefore, the use of multiple fibers has become inevitable in large scale networks. In the context of multifiber optical networks several optimization problems have been defined and studied, the objective usually being to minimize either the maximum fiber multiplicity per edge or the sum of these maximum multiplicities over all edges of the graph.

¹ This work has been funded by the project PENED 2003. The project is cofinanced 75% of public expenditure through EC–European Social Fund, 25% of public expenditure through Ministry of Development–General Secretariat of Research and Technology of Greece and through private sector, under measure 8.3 of Operational Programme “Competitiveness” in the 3rd Community Support Programme.

Preliminaries. We introduce *Colored Resource Allocation Games*, a class of games that can model non-cooperative versions of routing and wavelength assignment problems in multifiber all-optical networks. They can be viewed as an extension of congestion games where each player has his strategies in multiple copies (colors). When restricted to (optical) network games, facilities correspond to edges of the network and colors to wavelengths. The number of players using an edge in the same color represents a lower bound on the number of fibers needed to implement the corresponding physical link. We consider both egalitarian (max) and utilitarian (sum) player costs. For our purposes it suffices to restrict our study to identity latency functions.

Definition 1. (Colored Resource Allocation Games) A Colored Resource Allocation Game is defined as a tuple $\langle F, N, W, \{\mathcal{E}_i\}_{i \in [N]} \rangle$, such that F is a set of facilities, N is the number of players, W is the number of colors, and \mathcal{E}_i is a set of possible facility combinations for player i . For any player i , $\mathcal{E}_i \subseteq 2^F$.

For any player i , the set of possible strategies is $S_i = \mathcal{E}_i \times [W]$. We denote by $A_i = (E_i, c_i)$ the strategy that player i actually chooses, where $E_i \in \mathcal{E}_i$ denotes the set of facilities she chooses, and c_i denotes her color. Furthermore, we use the standard notation $A = (A_1, \dots, A_N)$ for a strategy profile for the game, and the notation $n_{f,c}(A)$ for the number of players that use facility f in color c in the strategy profile A .

Definition 2. Depending on the player cost function we define two subclasses of Colored Resource Allocation Games. *Colored Congestion Games*, with player cost $C_i(A) = \sum_{e \in E_i} n_{e,c_i}(A)$, and *Colored Bottleneck Games*, with player cost $C_i(A) = \max_{e \in E_i} n_{e,c_i}(A)$.

We use the price of anarchy (PoA) introduced in [11] as a measure of the deterioration caused by lack of coordination. We estimate the PoA of our games under three different social cost functions. Two of them are standard in the literature (see e.g. [8]): the first (SC_1) is equal to the maximum player cost and the second (SC_2) is equal to the sum of player costs (equivalently, the average player cost). The third one is specially designed for the setting of multifiber all-optical networks; it is equal to the sum over all facilities of the maximum color congestion on each facility. Note that in the optical network setting this function represents the total fiber cost needed to accommodate all players; hence, it captures the objective of a well-studied optimization problem [17; 16; 1]. Let us also note that the SC_1 function under the egalitarian player cost captures the objective of another well known problem, namely minimizing the maximum fiber multiplicity over all edges of the network (see e.g. [12]).

Related work. Bottleneck games have been studied in [7; 4; 9; 13]. In [7] the authors study atomic routing games on networks, where each player chooses a path

Table 2. *The pure price of anarchy of Colored Congestion Games (utilitarian player cost). Results for classical congestion games are shown in the right column.*

	Colored Congestion Games	Congestion Games
$SC_1(A) = \max_{i \in [N]} C_i(A)$	$\Theta\left(\sqrt{\frac{N}{W}}\right)$	$\Theta(\sqrt{N})$ [8]
$SC_2(A) = \sum_{i \in [N]} C_i(A)$	$\frac{5}{2}$	$\frac{5}{2}$ [8]
$SC_3(A) = \sum_{f \in F} \max_{a \in [W]} n_{f,a}(A)$	$\Theta\left(\sqrt{W F }\right)$	—

Table 3. *The pure price of anarchy of Colored Bottleneck Games (egalitarian player cost). Results for classical bottleneck games are shown in the right column.*

	Colored Bottleneck Games	Bottleneck Games
$SC_1(A) = \max_{i \in [N]} C_i(A)$	$\Theta\left(\frac{N}{W}\right)$	$\Theta(N)$ [7]
$SC_2(A) = \sum_{i \in [N]} C_i(A)$	$\Theta\left(\frac{N}{W}\right)$	$\Theta(N)$ [7]
$SC_3(A) = \sum_{f \in F} \max_{a \in [W]} n_{f,a}(A)$	$\frac{ E_A }{ E_{\text{OPT}} } \frac{N}{W}$	—

to route her traffic from an origin to a destination node, with the objective of minimizing the maximum congestion on any edge of her path. A further generalization is the model of Banner and Orda [4], where they introduce the notion of bottleneck games.

Selfish path coloring in single fiber all-optical networks has been studied in [6; 5; 10; 14]. Bilò and Moscardelli [6] consider the convergence to Nash Equilibria of selfish routing and path coloring games. Bilò et al. [5] consider several information levels of local knowledge that players may have and give bounds for the PoA in chains, rings and trees. The existence and complexity of computing Nash equilibria under various payment functions are considered by Georgakopoulos et al. [10]. In [14] they study the PoA of selfish routing and path coloring, under functions that charge a player only according to her own strategy. Selfish path multicoloring games are introduced in [3] where it is proved that the pure price of anarchy is bounded by the number of available colors and by the length of the longest path; constant bounds for the PoA in specific topologies are also provided. In those games, in contrast to the ones studied here, routing is given in advance and players choose only colors.

Our results. Our main contribution is the derivation of tight bounds on the price of anarchy for Colored Resource Allocation Games. These bounds are summarized in Tables 2 and 3. It can be shown that the bounds for Colored Congestion

Games remain tight even for network games. Observe that known bounds for classical congestion and bottleneck games can be obtained from our results by simply setting $W = 1$. On the other hand one might notice that our games can be casted as classical congestion or bottleneck games with $W |F|$ facilities. However, we are able to derive better upper bounds for most cases by exploiting the special structure of the players' strategies. Finally, we provide a potential function for Colored Bottleneck Games in order to prove the existence and convergence to pure equilibria and we show that the price of stability (as defined in [2]) is equal to 1.

References

- [1] M. Andrews and L. Zhang. Minimizing maximum fiber requirement in optical networks. *J. Comput. Syst. Sci.*, **72**(1): 118–131 (2006).
- [2] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. FOCS 2004, 295–304.
- [3] E. Bampas, A. Pagourtzis, G. Pierrakos, and K. Potika. On a non-cooperative model for wavelength assignment in multifiber optical networks. ISAAC 2008, 159–170.
- [4] R. Banner and A. Orda. Bottleneck routing games in communication networks. INFOCOM 2006.
- [5] V. Bilò, M. Flammini, and L. Moscardelli. On Nash equilibria in non-cooperative all-optical networks. STACS 2005, 448–459.
- [6] V. Bilò and L. Moscardelli. The price of anarchy in all-optical networks. SIROCCO 2004, 13–22.
- [7] C. Busch and M. Magdon-Ismail. Atomic routing games on maximum congestion. AAIM 2006, 79–91.
- [8] G. Christodoulou and E. Koutsoupias. The price of anarchy of finite congestion games. STOC 2005, 67–73.
- [9] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. ICALP 2002, 123–134.
- [10] G.F. Georgakopoulos, D.J. Kavvadias, and L.G. Sioutis. Nash equilibria in all-optical networks. WINE 2005, 1033–1045.
- [11] E. Koutsoupias and C.H. Papadimitriou. Worst-case equilibria. STACS 1999, 404–413.
- [12] G. Li and R. Simha. On the wavelength assignment problem in multifiber WDM star and ring networks. *IEEE/ACM Trans. Netw.*, **9**(1): 60–68 (2001).
- [13] L. Libman and A. Orda. Atomic resource sharing in noncooperative networks. *Telecommunication Systems*, **17**(4): 385–409 (2001).
- [14] I. Milis, A. Pagourtzis, and K. Potika. Selfish routing and path coloring in

- all-optical networks. CAAN 2007, 71–84.
- [15] D. Monderer and L.S. Shapley. Potential games. *Games and Economic Behavior*, **14**: 124–143 (1996).
 - [16] C. Nomikos, A. Pagourtzis, K. Potika, and S. Zachos. Routing and wavelength assignment in multifiber WDM networks with non-uniform fiber cost. *Computer Networks*, **50**(1): 1–14 (2006).
 - [17] C. Nomikos, A. Pagourtzis, and S. Zachos. Routing and path multicoloring. *Inf. Process. Lett.*, **80**(5): 249–256 (2001).
 - [18] R.W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *Int. J. Game Theory*, **2**: 65–67 (1973).
 - [19] T. Roughgarden. Selfish routing and the price of anarchy. The MIT Press, 2005.

Cutting and Packing

Lecture Hall A

Tue 2, 14:00–15:30

A Lower Bound for the Cutting Stock Problem with a Limited Number of Open Stacks

Claudio Arbib,^a Fabrizio Marinelli,^b Carlo M. Scoppola^c

^a*Università degli Studi di L'Aquila, Dipartimento di Informatica,
via Vetoio, I-67010 Coppito, L'Aquila, Italia
arbib@di.univaq.it*

^b*Università Politecnica delle Marche,
Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione
via Brezze Bianche, I-60131 Ancona, Italy
marinelli@diiga.univpm.it*

^c*Università degli Studi di L'Aquila, Dipartimento di Matematica Pura e Applicata,
via Vetoio, I-67010 Coppito, L'Aquila, Italia
scoppola@univaq.it*

Abstract

This paper proposes an algorithm to compute a lower bound for the cutting stock problem subject to a limited number s of open stacks. The algorithm employs an enumeration scheme based on algebraic properties of the problem. Search is shortened by bounds computed via column generation and by symmetry breaking, implemented to avoid the repeated evaluation of equivalent solutions. A preliminary computational experience confirms the effectiveness of the method.

Key words: Cutting Stock Problem, Lower Bounds

1. The problem

Consider a sufficiently large set of stock items having standard length w , and a finite set B of m batches, the i -th consisting of a known amount d_i of required parts of length w_i ($i \in B$). A *cutting pattern* k specifies the number $a_{ik} \geq 0$ of items of type i that are produced when the pattern is applied on a single stock item. The Cutting Stock Problem (*CSP*) calls for finding a set P of feasible cutting patterns and deciding how many stock items must be cut according to each $k \in P$, with the objective of satisfying the requirements of parts with a minimum total number of

stock items [3].

Any *CSP* solution P is in general implemented by applying its patterns in some order π , but not all the orders are feasible if the cutting machine has an s -slot outbuffer able to maintain up to s distinct batches at a time, and a slot occupied by parts of some type can be released only if the relevant batch has been completed. With this kind of technological constraint we speak of a *Cutting Stock Problem with a Limited Number s of Open Stack* (CSP_s) [2].

More formally, we say that a *CSP* solution P is *schedulable* if it can be sequenced in an order π so that, at any time, the number of distinct batches which are not completed (*open stacks*) never exceeds s . Then, CSP_s can be stated as follows:

Problem 1. Find a *schedulable* cutting stock solution that produces all the required batches with a minimum trim loss.

Let \mathbf{Q} be a 0-1 matrix with $m = |B|$ rows, none of which null. We call \mathbf{Q} a *track* for a *CSP* solution P if, for any $k \in P$, there exists a column t with $q_{it} > 0$ for all $i \in B$ such that $a_{ik} > 0$. Reciprocally, we say that P is *supported* by \mathbf{Q} , and in particular that a single part type i is supported by a column \mathbf{q}_t of \mathbf{Q} if $q_{it} = 1$. A track \mathbf{Q} is dominated by a track \mathbf{R} if the set of *CSP* solutions supported by \mathbf{R} includes that supported by \mathbf{Q} . With no loss of generality, from now on we assume the columns of \mathbf{Q} mutually non-dominated, that is, no two columns $\mathbf{q}_r, \mathbf{q}_t$ are such that $q_{ir} \leq q_{it}$ for all $i \in B$.

Let $\omega(\mathbf{Q})$ denote the largest number of non-zero elements in a column of a track \mathbf{Q} . We then say that \mathbf{Q} is *feasible* if

- it has the *consecutive one property* (C1P) by rows, that is, $q_{ij} = q_{ik} = 1 \Rightarrow q_{ih} = 1$ for $j < h < k$ and all $i \in B$;
- $\omega(\mathbf{Q}) \leq s$.

Proposition 1. A *CSP* solution P is schedulable if and only if it is supported by a feasible track \mathbf{Q} .

Proposition 2. Every feasible track \mathbf{Q} with $\omega(\mathbf{Q}) = s$ is dominated by a feasible track \mathbf{R} having (i) each column with exactly s non-zero elements and (ii) any two adjacent columns different for exactly two elements.

2. Computing a lower bound

Let $CSP(s)$ indicate a cutting stock problem where no pattern can produce more than s distinct types. The solution of (the linear relaxation of) $CSP(s)$ provides a valid lower bound to CSP_s . An improved lower bound z_{LB} can be obtained on the basis of Propositions 1 and 2 by enumerating all the non-dominated feasible tracks. In fact, let \mathbf{R}_k denote the submatrix consisting of the first k columns

of a track \mathbf{R} . An implicit enumeration can be performed by calling, for any 0-1 m -vector \mathbf{r} with s non-zeroes, the following recursive function `track_enum()` with parameters 1, \mathbf{r} and B .

```

track_enum( $k, \mathbf{R}_k, B_k$ )
For all columns  $\mathbf{r}$  with  $s$  non-zeroes in  $B_k$  and such that  $\mathbf{r} \cdot \mathbf{r}_k = s - 1$  do
  (1)  $\mathbf{R}_{k+1} := [\mathbf{R}_k | \mathbf{r}]$ 
  (2) For  $i \in B_k$  do if  $r_{ik} > r_i$  then  $B_{k+1} := B_k - \{i\}$ 
  (3) if  $z_{LB}(\mathbf{R}_{k+1}) \leq z_{LB}$  then track_enum( $k + 1, \mathbf{R}_k, B_k$ )

```

Removing row i from B_k at Step 2 corresponds to fixing to 0 the i -th element of any columns generated from then on: this ensures that the resulting matrix has the C1P. As soon as $m - s + 1$ elements have been fixed, non-dominated columns with s non-zeroes cannot be added any longer. Step 3 performs fathoming: z_{LB} is the best lower bound found so far and $z_{LB}(\mathbf{R}_{k+1})$ is a lower bound to the optimum attainable with patterns supported by the uncompleted track \mathbf{R}_{k+1} .

Call K the set of the cutting patterns that either (i) are supported by \mathbf{R}_{k+1} , or (ii) produce $\leq s$ part types in B_{k+1} . Then $z_{LB}(\mathbf{R}_{k+1})$ is the optimum value of linear program (2.1), which can be computed by a standard column generation procedure.

$$z_{LB}(\mathbf{R}_{k+1}) = \min \left\{ \sum_{k \in K} x_k \mid \sum_{k \in K} a_{ik} x_k = d_i, i \in B, x_k \geq 0, k \in K \right\} \quad (2.1)$$

3. Symmetry breaking

Symmetry breaking means in general to fathom unnecessary equivalent tracks in order to reduce the search space. Tracks corresponding to column permutations are equivalent in the sense that produce by program (2.1) identical optimal solutions and lower bounds. A maximal set of equivalent tracks is an *orbit* of a permutation group \mathcal{G} . Of course, we are not interested in all the column permutations of \mathcal{G} , but only in those, called feasible, which preserve the C1P: one is for instance the *mirror* permutation μ that sends each column $j \in C$ into $|C| - j$. To this purpose, let us introduce the following notion.

Definition 2. Let \mathbf{Q} be a track with columns indexed in C . Then $S \subseteq C$ is said to be *permutable* if any permutation π such that $\pi(j) = j$ for $j \in C - S$ is feasible.

Maximal permutable sets are defined in the obvious way. We call trivial a permutable set consisting of a single column, and we say that a row $i \in B$ of a track \mathbf{Q} is a *unit row* if it contains exactly one non-zero element. In order to identify equivalent tracks we characterize permutable sets as follows.

Theorem 3. $S \subseteq C$ is permutable if and only if for any row $i \in B$ either $q_{ij} = q_{ik}$ for all $j, k \in S$ or i is a unit row of \mathbf{Q} .

On the basis of Theorem 3 it can be observed that every non-trivial permutable set S_t hits at least $|S_t|$ unit rows, for otherwise C would have identical columns. Thus a permutable set is identified as soon as a unit row is detected during the generation of a track, i.e., when the current uncompleted track \mathbf{R}_k contains a row i for which $r_{ik-2} = 0, r_{ik-1} = 1$ and $r_{ik} = 0$. Hence to avoid tracks having the same permutable set it is sufficient to apply the lexicographic order to the row-subsequences 010.

4. Preliminary computational results

A preliminary computational test was done on a set of 80 random instances, of which 40 with $s = 2$ and 40 with $s = 3$, and all with $m = 10$ part types. A feasible solution was computed by the *first-sequence-then-cut* heuristic proposed by [1]. In 49 cases the lower bound obtained by the linear relaxation of $CSP(s)$ provided the same value of the feasible solution, which therefore turned out to be optimal. In the remaining 31 cases our algorithm improved the bound, and in 3 of these allowed closing the gap.

References

- [1] Arbib, C., F. Marinelli, F. Pezzella, Trim Loss Minimization Under a Given Number of Open Stacks, XXXIX Conference, AIRO2008, Ischia, Italy, September 7-11, 2008.
- [2] Belov, G. and G. Scheithauer, Setups and open-stacks minimization in one-dimensional stock cutting, *INFORMS J. on Computing*, 19 1 (2007) 27-35.
- [3] Wäscher G., H. Haußner, H. Schumann, An improved typology of cutting and packing problems, *European Journal of Operational Research*, 183 (2007) 1109-1130.

Packing Paths: Recycling Saves Time

Henning Fernau,^a Daniel Raible^a

^a*Universität Trier, FB 4—Abteilung Informatik, D-54286 Trier, Germany*
{fernau,raible}@uni-trier.de

Key words: parameterized algorithms, graphs, packings

1. Introduction and Definitions

A P_d is a path of length d and therefore has $d + 1$ vertices. A P_d -packing of a graph $G(V, E)$ is a set of vertex-disjoint copies of a P_d in G . It is maximal if any addition of a further P_d would violate the vertex disjointness property. The problem we investigate in this paper is P_d -PACKING ($d \geq 3$):

Given $G(V, E)$, and the parameter k .

We ask: Is there a P_d -packing of size k ?

P. Hell and D. Kirkpatrick [5; 4] showed that general MAXIMUM H -PACKING is \mathcal{NP} -complete. Here, H is a graph with at least three vertices in some connected component. A subcase of H -packing is of course a P_d -packing if $d \geq 2$. $d = 1$ corresponds to the classical matching problem.

For the special case of P_2 -packing there have been already publications in the fields of parameterized and approximation algorithms. The currently fastest algorithm solving P_2 -packing in time $\text{MCO}^*(2.482^{3k})$ is the one of H. Fernau and D. Raible [3]. This algorithm combines the $7k$ -kernel of J. Wang *et al.* [8] together with the idea to improve the recyclability of vertices in an inductive approach. Although no linear kernels are known for P_d -packing (for $d > 2$), we propose here a similar approach that helps with the second, color-coding phase of hitherto published packing algorithms.

More specifically, in [7] it was shown that for any maximal 3-set packing $CRRAP$ of size j there is a packing \mathcal{Q} of size $j + 1$ reusing at least $2j$ vertices from $CRRAP$. We show that this result is also valid for general ℓ -SET PACKING and therefore also for P_d -packing. Thus, the algorithm of [7] can easily be adapted to P_d -PACKING. In [3], we showed that for P_2 -packings, we can reuse at least $2.5j$ vertices of a P_2 -packing of size j . We prove similar results for P_d -PACKING if $3 \leq d \leq 5$. Namely, we show that one can reuse $3j$ vertices for $d = 3$ and $d = 5$

Algorithm-type	P_3 -PACKING	P_4 -PACKING	P_5 -PACKING
ℓ -SET PACKING	$\text{MCO}^*(6.99^{4k})$	$\text{MCO}^*(8.98^{5k})$	$\text{MCO}^*(10.6^{6k})$
P_d -PACKING	$\text{MCO}^*(4.18^{4k})$	$\text{MCO}^*(6.99^{5k})$	$\text{MCO}^*(6.99^{6k})$

Table 4. The tables gives the running times of ordinary ℓ -SET PACKING-algorithm (see Alg. 1) and their improvements due to better reusability results in case of P_d -packing.

and $2.5j$ vertices for $d = 4$. Considering this results we can speed up the algorithm for P_d -PACKING yielding run times of $\text{MCO}^*(5.8006^{4k})$, $\text{MCO}^*(6.9857^{5k})$ and $\text{MCO}^*(6.99^{6k})$ for $d = 3$, $d = 4$ and $d = 5$, respectively. Table 4 lists our run times, where for P_3 -packings we employed a refined analysis technique.

A path is an ordered set of vertices $p_1 \dots p_j$ such that $\{p_i, p_{i+1}\} \in E$ for $1 \leq i \leq j - 1$. Generally, the paths $p_1 \dots p_j$ and $p_j \dots p_1$ are considered the same, as their edge-sets are the same. Nevertheless, at some points we need to order the considered path. We set $E(p) := \{\{p_i, p_{i+1}\} \mid 1 \leq i < j\}$ and $V(p) := \{p_i \mid 1 \leq i \leq j\}$. For a set of sets $S = \{S_1, \dots, S_m\}$ we let $V(S) = \bigcup_{1 \leq i \leq m} S_i$, i.e., $V(S)$ comprises the elements of which the S_i consist. A path p is called *subpath* of a path p' if a $E(p) \subseteq E(p')$. Two paths p and p' intersect if $V(p) \cap V(p') \neq \emptyset$.

2. Properties of P_d -Packings

The general strategy is that we use an already existent maximal solution $CRRAP$ of a ℓ -set packing instance of size j to obtain a solution \mathcal{Q} of size $j + 1$. For this task it is of importance how many of the $d \cdot j$ vertices of $V(CRRAP)$ appear also in $V(\mathcal{Q})$. Among all ℓ -set-packings of size $(j + 1)$, we will consider those packings \mathcal{Q} that maximize

$$|CRRAP \cap \mathcal{Q}|. \quad (2.1)$$

We subsume these packings under the name $\text{MCL}_{(1)}^d$. In $\text{MCL}_{(1)}^d$, we find those packings \mathcal{Q} that 'reuse' the maximum number of sets from the packing $CRRAP$. The authors of [3] showed the next proposition with respect to P_2 -packings by strengthening a proposition of [7]. But browsing their proof shows that it can be generalized straightforward for ℓ -set packings.

Proposition 2.1. If $\mathcal{Q} \in \text{MCL}_{(1)}^d$, then for any $p \in CRRAP$ with $p \notin \mathcal{Q}$, there are $q^1, q^2 \in \mathcal{Q}$, $q^1 \neq q^2$, with $|V(p) \cap V(q^i)| \geq 1$ ($i = 1, 2$).

The algorithm of [7] for 3-SET-PACKING also serves for ℓ -SET-PACKING if we modify it slightly, see Alg. 1:

The color-coding and the dynamic programming part in Alg. 1 can be upperbound by $\text{MCO}^*(6.1^{(\ell-2)k})$ and $\text{MCO}^*(\sum_{z=1}^{j+1} \binom{2j(d-1)+d}{dz}) \subseteq \text{MCO}^*(2^{2k(\ell-1)})$,

respectively.

Lemma 2.2. We can find a ℓ -set packing in time $\text{MCO}^*(c_\ell^{\ell k})$ ($c_\ell = \sqrt[\ell]{24 \cdot 4^{\ell-2} \cdot 4}$).

Any P_d -PACKING instance can be transferred into a $(d + 1)$ -SET-PACKING instance. In the cases of P_3 , P_4 and P_5 -PACKING we will show that we can 'recycle' more than $2j$ vertices. Similar improvements have been achieved by [3] for P_2 -PACKING. This accelerates Alg. 1 quite drastically.

Let $CRRAP$ be a maximal P_d -packing of size j . Among all P_d -packings of size $j + 1$ we will only consider those who maximize property (2.1). We subsumed them in $\mathfrak{Q}_{(1)}^d$. Consequently, we have $\mathfrak{Q}_{(1)}^d = \text{MCL}_{(1)}^{d+1}$ with respect to the corresponding $d + 1$ -SET-PACKING instance. Furthermore, from the set $\mathfrak{Q}_{(1)}^d$ we only comprise those P_d -packings Q' , which maximize the following second property:

$$\sum_{p \in \text{MCP}} \sum_{q \in \text{MCQ}'} |E(p) \cap E(q)|. \quad (2.2)$$

The set of the remaining P_d -packings will be called $\mathfrak{Q}_{(2)}^d$. $\mathfrak{Q}_{(2)}^d$ contains those packings from $\mathfrak{Q}_{(1)}^d$ which reuse the maximum number of edges in $E(CRRAP)$. We further distinguish the packings in $\mathfrak{Q}_{(2)}^d$ by considering only those minimizing

$$|\text{MCP}_2(\text{MCQ})|, \text{ where } \text{MCP}_i(\text{MCQ}) = \{p \in \text{MCP} \mid i = |p \cap V(\text{MCQ})|\}. \quad (2.3)$$

These packings are referred to as $\mathfrak{Q}_{(3)}^d$ and contain those packings from $\mathfrak{Q}_{(2)}^d$ with the least number of paths from $CRRAP$ such that only two vertices are reused.

Path-packings can benefit from the flexibility of folding and shifting paths on graph edges. We refrain from giving formal definitions due to reasons of space.

- Lemma 2.3.** (i) If $q \in \mathcal{Q}$ with $\mathcal{Q} \in \mathfrak{Q}_{(2)}^d$ is ℓ -shiftable on $p \in CRRAP$ with respect to q_1 (q_{d+1} , resp.) then there is $p' \in CRRAP$ such that $q_{d+1} \dots q_{d+1-\ell}$ ($q_1 \dots q_{\ell+1}$, resp.) is a subpath of $p' \neq p$.
- (ii) If $\mathcal{Q} \in \mathfrak{Q}_{(2)}^d$ then no $q \in \mathcal{Q}$ is ℓ -foldable, $\ell \geq 1$.
- (iii) If $\mathcal{Q} \in \mathfrak{Q}_{(3)}^d$ then no $q \in \mathcal{Q}$ is 2-shiftable on $p \in CRRAP$ with $|V(p) \cap V(\mathcal{Q})| = 2$.

Algorithm 1 An Algorithm for general path packing

- 1: Greedily find a maximal ℓ -set packing $CRRAP$ of G .
 - 2: if $j := |CRRAP| \geq k$ return $CRRAP$.
 - 3: Color the vertices $V \setminus V(CRRAP)$ with $(\ell - 2)j + \ell$ colors by color-coding.
 - 4: Color $V(CRRAP)$ by ℓj additional colors arbitrarily.
 - 5: Check if there are $\ell(j + 1)$ vertices with pairwise different colors that can be perfectly packed by a ℓ -set packing $CRRAP'$ using dynamic programming. $CRRAP \leftarrow CRRAP'$.
 - 6: **goto** 2.
-

3. P_3, P_4 and P_5 -packings

Henceforth, we will only consider P_d -packings from $\Omega_{(3)}^d$. We show that if $CRRAP$ is a maximal P_d -packing with $d \in \{3, 4, 5\}$ of size j , we can reuse more than $2j$ of its vertices. More formally: If there is a P_d -packing \mathcal{Q} of size $j + 1$ we can rely on $|V(CRRAP) \cap V(\mathcal{Q})| \geq 2.5j$. Actually, in most cases we prove a sharper statement. Namely, for all $p \in CRRAP$ we have $|V(p) \cap V(\mathcal{Q})| \geq 3$. Suppose a path $p = p_1 \dots p_{d+1} \in CRRAP$ shares exactly one vertex $p_{q'}, p_{q''}$ with paths $q', q'' \in \mathcal{Q}$ each (i.e., $|V(p) \cap V(\mathcal{Q})| = 2$). Due to Proposition 2.1, q', q'' must exist. Subsequently, $p_{q'}, p_{q''}$ are the cut vertices of these $q', q'' \in \mathcal{Q}$ with $p \in CRRAP_2(\mathcal{Q})$.

Let $p_i := p_{q'}$ and $p_j := p_{q''}$; w.l.o.g., $i < j$. Then p_i and p_j define three (possibly empty) subpaths on p : $X_{q'} := p_1 \dots p_{i-1}$, $X_{q'q''} := p_{i+1} \dots p_{j-1}$ and $X_{q''} := p_{j+1} \dots p_{d+1}$. Subsequently, we write $|X_i|$ for $|V(X_i)|$. Next we will discuss the case where $p_{q'}$ and $p_{q''}$ are not end-points of q' and q'' , respectively. For any path p of length d the *mid-points* is the set $\{p_{\lceil d+1/2 \rceil}, p_{\lfloor d+1/2 \rfloor}\}$. A vertex m_q is a \mathcal{Q} -mid-point if there is a $q \in \mathcal{Q}$ with m_q being a mid-point of q .

Lemma 3.1. Let $CRRAP$ be a maximal P_d -packing, $p \in CRRAP$, $V(p) \cap V(\mathcal{Q}) = \{p_{q'}, p_{q''}\}$.

- (i) If $d \in \{3, 5\}$ then one of $p_{q'}, p_{q''}$ must be a \mathcal{Q} -end-point, w.l.o.g., $p_{q'}$.
- (ii) If $d \in \{3, 4, 5\}$, such that $p_{q''}$ is not an \mathcal{Q} -end-point but $p_{q'}$ is. Then $p_{q'}$ is 1-shiftable for $d = 4$ and 2-shiftable for $d \in \{3, 5\}$.
- (iii) For $d \in \{3, 5\}$ $p_{q''}$ is a \mathcal{Q} -end-point.

Lemma 3.1.(i) does not hold for P_4 -packings and any P_d -packing with $d > 5$. For P_5 -packings, this lemma immediately gives the desired recycling:

Lemma 3.2. Let $CRRAP$ be a maximal P_5 -packing of size j . If there is a packing of size $j + 1$, then there is also a packing $\mathcal{Q} \in \Omega_{(3)}^5$ such that for all $p \in CRRAP$ we have $|V(p) \cap V(\mathcal{Q})| \geq 3$.

P_3 -packings are more subtle. Among the packings $\Omega_{(3)}^3$ are those packings \mathcal{Q} that maximize

$$\sum_{p \in \text{MCP}} \sum_{q \in \text{MCQ}} |end(p) \cap E(q)| \quad (3.4)$$

where $end(p)$ denotes the set of two end-edges, i.e., when $p = p_1 \dots p_4$ is a path, then $\{p_1, p_2\}$ and $\{p_3, p_4\}$ are comprised in the set $end(p)$. We call those $\Omega_{(4)}^3$. In $\Omega_{(4)}^3$ are those packings from $\Omega_{(3)}^3$ such that greatest number of end-edges is reused. We call a path $q \in \mathcal{Q}$ *end-1-shiftable* on some $p \in P$ if q is 1-shiftable and we can shift q by one in a way that we cover an end-edge of p .

Theorem 3.3. Let $CRRAP$ be a maximal P_3 -packing of size j . If there is a P_3 -

packing of size $j + 1$ then exists $\mathcal{Q} \in \Omega_{(4)}^3$ such that for all $p \in CRRAP$ we have $|V(p) \cap V(\mathcal{Q})| \geq 3$.

Further speed-up techniques, using ideas from [8], are necessary to show the claimed running time. We could only prove a weaker lemma for P_4 -packings:

Lemma 3.4. Let $CRRAP$ be a maximal P_4 -Packing with size j . If there is a P_4 -packing of size $j + 1$ then there is also a packing $\mathcal{Q} \in \Omega_{(3)}^4$ such that we have $|V(CRRAP) \cap V(\mathcal{Q})| \geq 2.5 \cdot j$.

4. Conclusion

Our algorithmic approach is of the iterative expansion type. Starting from a maximal solution, investigate the relation to a possible larger solution. For path packing problems, a certain amount of vertices in the old solution also appears in the larger one, reducing the cost of the expansion step. The question of reusability is an interesting issue in extremal combinatorics on its own right. So, the following type of questions should be explored independently of possible algorithmic consequences: Given a maximization problem and a feasible solution S of size k to that problem for a certain instance, is it possible to either construct a solution of size $(k + 1)$ (or larger), re-using as many elements from S as possible, or to conclude that no such larger solution exists?

References

- [1] J. Chen and S. Lu. Improved algorithms for weighted and unweighted set splitting problems. In *COCOON, LNCS 4598*, 537–547, 2007.
- [2] J. Chen and S. Lu. Improved parameterized set splitting algorithms: A probabilistic approach. *Algorithmica*, To appear.
- [3] H. Fernau and D. Raible. A parameterized perspective on packing paths of length two. In *COCOA, LNCS 5165*, 54–63, 2008.
- [4] P. Hell and D. G. Kirkpatrick. Star factors and star packings. Technical Report 82-6, Computing Science, Simon Fraser University, Burnaby, Canada, 1982.
- [5] D. G. Kirkpatrick and P. Hell. On the completeness of a generalized matching problem. In *Symposium on Theory of Computing STOC*, 240–245, 1978.
- [6] I. Koutis. Faster algebraic algorithms for path and packing problems. In *ICALP, LNCS 5125*, 575–586, 2008.
- [7] Y. Liu, S. Lu, J. Chen, S.-H. Sze. Greedy localization and color-coding: improved matching and packing algorithms. In *IWPEC, LNCS 4169*, 84–95, 2006.
- [8] J. Wang, D. Ning, Q. Feng, J. Chen. An improved parameterized algorithm for a generalized matching problem. In *TAMC, LNCS 4978*, 212–222, 2008.

Rectangle Packing with Additional Restrictions

Jens Maßberg,^a Jan Schneider^a

^a*Research Institute for Discrete Mathematics
Lennéstr. 2, 53113 Bonn, Germany
{massberg, schneid}@or.uni-bonn.de*

Key words: rectangle packing, NP-completeness

1. Problem Formulation

RECTANGLE PACKING as a decision problem, i.e. the question if a set of rectangles can be disjointly packed into a bounding box of given dimensions or area, is easily seen to be NP-complete [1].

However, in practical applications it is often possible to restrict the instances in one or another way. It might be possible to guarantee that the rectangles do not have extreme aspect ratios, or do not differ very much in their area consumption. Also, the bounding box could be a given percentage larger than the total size of the rectangles in the instance. In [3], we parameterized RECTANGLE PACKING to incorporate restrictions of these kinds and analyzed the computational complexity of the resulting problems. The decision problem (α, β, γ) -PACKING is defined as follows:

Instance: A set of n rectangles with widths w_i and heights h_i and a real number A satisfying

- $A \geq \alpha \cdot \sum_{i=1}^n w_i h_i$
- $w_i h_i \leq \beta \cdot w_j h_j$ for $1 \leq i, j \leq n$
- $\max\{w_i, h_i\} \leq \gamma \cdot \min\{w_i, h_i\}$ for $1 \leq i \leq n$

Question: Is there a disjoint packing of the rectangles such that their bounding box has an area of at most A ?

To simplify notation, (α, β, ∞) -PACKING shall denote the version of the problem where only the first two conditions hold with the given α and β . Analogously, (α, ∞, γ) -PACKING stands for the version where only the first and last conditions hold.

2. Results

The parameters α , β , and γ can be varied in many ways to generate different problems. The first one considered is the one with $\beta = 1$, meaning that all rectangles in the instance have the same area, and $\gamma = \infty$, so that the aspect ratio of single rectangles can be arbitrarily large. Theorem 1 states that this problem is *NP*-complete even if the packing's density is arbitrarily low:

Theorem 2.1. $(\alpha, 1, \infty)$ -PACKING is *NP*-complete for every $\alpha \geq 1$.

Corollary 1. (α, β, ∞) -PACKING is *NP*-complete for every $\alpha \geq 1$ and $\beta \geq 1$.

The next theorem considers squares (i.e. $\gamma = 1$) whose areas differ by a factor of at most $1 + \epsilon$:

Theorem 2.2. $(1, 1 + \epsilon, 1)$ -PACKING is *NP*-complete for every $\epsilon > 0$.

We found a similar result for rectangles which are almost squares (aspect ratio below $1 + \epsilon$) and have the same area:

Theorem 2.3. $(1, 1, 1 + \epsilon)$ -PACKING is *NP*-complete for every $\epsilon > 0$.

It becomes apparent that many versions of (α, β, γ) -PACKING are still *NP*-complete. But there are also some cases where the answer is trivial – the most obvious one being $(1, 1, 1)$ -PACKING, whose instances only contain squares of the same size. As soon as the rectangles in the instance have similar size and bounded aspect ratio, it is clear that packings with a certain density are possible. The first result is achieved by simply arranging all rectangles in a row:

Theorem 2.4. If $\alpha \geq \sqrt{\beta\gamma}$, then the answer to (α, β, γ) -PACKING is yes.

We present a strip packing method to find a more elaborate result:

Lemma 2.5. Let r_1, \dots, r_n be the rectangle set from an instance of (α, β, γ) -PACKING and P the output of the strip packing algorithm running on the input r_1, \dots, r_n . If A denotes the area of P 's bounding box, then $A < (1 + \beta + \frac{\beta\gamma}{n}) \cdot \sum_{i=1}^n w_i h_i$.

Theorem 2.6. If $\alpha > \beta + 1$, then (α, β, γ) -PACKING can be decided in time $\mathcal{O}(1)$.

Pavel Novotný proved in [2] that any set of squares with a total area of 1 can be packed into a rectangle of area 1.53. Hence, the following theorem holds:

Theorem 2.7. If $\alpha \geq 1.53$, then the answer to $(\alpha, \infty, 1)$ -PACKING is yes.

To conclude, the following table shows a summary of the results on (α, β, γ) -PACKING.

	$\gamma = 1$	$1 < \gamma < \infty$	$\gamma = \infty$
$\alpha = 1$	Trivial for $\beta = 1$ NPC otherwise	NPC	NPC
$\alpha > 1$	Trivial for $\alpha \geq 1.53$ Trivial for $\alpha \geq \sqrt{\beta}$	Trivial for $\alpha > \beta + 1$ Trivial for $\alpha \geq \sqrt{\beta\gamma}$	NPC

References

- [1] Richard E. Korf. Optimal Rectangle Packing: Initial Results. In *ICAPS '03: Proceedings of the International Conference on Automated Planning and Scheduling*, pages 287–295, AAAI, 2003.
- [2] Pavel Novotný. On Packing of Squares Into a Rectangle. *Archivum Mathematicum (Brno)*, 32(1):75–83, 1996.
- [3] Jan Schneider. *Macro Placement in VLSI Design*. Diplomarbeit, Rheinische Friedrich-Wilhelms-Universität Bonn, 2009.

Paths

Lecture Hall B

Tue 2, 14:00–15:30

The open capacitated arc routing problem

Fábio Luiz Usberti,^a Paulo Morelato França,^b
André Luiz Morelato França^a

^a*FEEC/UNICAMP, C.P. 6102, 13083-970 Campinas SP, Brazil*
fusberti@yahoo.com

^b*FCT/UNESP, C.P. 468, 19060-900 Presidente Prudente SP, Brazil*

Key words: Arc Routing Problem, NP-hard Problem, Problem Reduction

Introduction. The Capacitated Arc Routing Problem (CARP) [1] is a well-known combinatorial optimization problem in which, given an undirected graph $G(V, E)$ with non-negative costs and demands associated to the edges, we have M identical vehicles with capacity D that must traverse all edges with positive demand. The vehicles must start and end their routes at a depot node, without transgressing their capacity. The objective is to search a solution of minimum cost. This work introduces the Open Capacitated Arc Routing Problem (OCARP), where vehicle's routes are not constrained to form cycles, therefore we are searching for minimum cost paths.

Problem Definition. The Open Capacitated Arc Routing Problem (OCARP) can be defined on an undirected graph $G(V, E)$ with edge costs $c_{ij} = c_{ji}$ and demands $d_{ij} = d_{ji}$. Edges with positive demands are called required ($R \subseteq E$) and must be serviced. M identical vehicles of capacity D are available. $N(i)$ denotes the nodes adjacent to node i in G . There are two set of decision variables: $x_{ij}^k = 1$ if vehicle k traverses edge (i, j) , $x_{ij}^k = 0$ otherwise; $l_{ij}^k = 1$ if vehicle k services (i, j) , $l_{ij}^k = 0$ otherwise. The OCARP objective is to find a set of paths with minimum total cost without overloading any vehicle capacity. The model uses the following auxiliary variables: $\alpha_i^k, \beta_i^k, y_S^k, u_S^k$ and v_S^k ($i \in V, k \in \{1, \dots, M\}, S \subseteq V$). An integer linear programming model for the OCARP is given.

$$\text{MIN} \quad \sum_{k=1}^M \sum_{(i,j) \in E} c_{ij} x_{ij}^k \quad (0.1)$$

st

$$\left. \begin{array}{l} \sum_{j \in N(i)} (x_{ji}^k - x_{ij}^k) \leq \alpha_i^k \\ \sum_{j \in N(i)} (x_{ji}^k - x_{ij}^k) \geq -\beta_i^k \end{array} \right\} \quad (i \in V, k \in \{1, \dots, M\}) \quad (0.2)$$

$$\left. \begin{array}{l} \sum_{i \in V} \alpha_i^k \leq 1 \\ \sum_{i \in V} \beta_i^k \leq 1 \end{array} \right\} \quad (k \in \{1, \dots, M\}) \quad (0.3)$$

$$x_{ij}^k \geq l_{ij}^k \quad ((i, j) \in E, k \in \{1, \dots, M\}) \quad (0.4)$$

$$\sum_{k=1}^M (l_{ij}^k + l_{ji}^k) = 1 \quad ((i, j) \in R) \quad (0.5)$$

$$\sum_i \sum_j l_{ij}^k d_{ij} \leq D \quad (k \in \{1, \dots, M\}) \quad (0.6)$$

$$\left. \begin{array}{l} \sum_{(i,j) \in (S,S)} x_{ij}^k - |S|^2 y_S^k \leq |S| - 1 \\ \sum_{(i,j) \in (S,\tilde{S})} x_{ij}^k + u_S^k \geq 1 \\ \sum_{(i,j) \in (\tilde{S},\tilde{S})} x_{ij}^k - |\tilde{S}|^2 v_S^k \leq 0 \\ y_S^k + u_S^k + v_S^k \leq 2 \end{array} \right\} \quad (S \subseteq V, \tilde{S} = V \setminus S, k \in \{1, \dots, M\}) \quad (0.7)$$

$$x_{ij}^k, l_{ij}^k \in \{0, 1\} \quad ((i, j) \in E, k \in \{1, \dots, M\}) \quad (0.8)$$

$$\alpha_i^k, \beta_i^k \in \{0, 1\} \quad (k \in \{1, \dots, M\}) \quad (0.9)$$

$$y_S^k, u_S^k, v_S^k \in \{0, 1\} \quad (k \in \{1, \dots, M\}, S \subseteq V) \quad (0.10)$$

The objective function (0.1) minimizes the solution cost. Constraints (0.2) and (0.3) guarantee that the nodes visited by a vehicle will have indegree equal to their outdegree, except for at most two nodes, which can have an unitary difference between indegree and outdegree (likewise a path). Constraints (0.4) state that serviced edges must be traversed; (0.5) force all required edges to be serviced; (0.6) are the capacity constraints; finally, constraints (0.7) assures path connectivity.

OCARP Applications. In this section, we consider some problems of practical interest which can be easily modeled (i.e., polynomially reduced) into an OCARP.

The Routing Meter Reader Problem [2; 3] interests major electric, water and gas distribution companies which periodically meter read their clients. This problem inspired the conception of OCARP and concerns the creation of a set of open routes for meter readers with limited amount of work time which visit all street segments containing clients in minimum traversal time. A service time is incurred always that a worker meter reads, while a shorter deadheading time is computed when the worker is not reading. While all street segments have positive deadhead time, some of them may have zero service time, which means there is no client on that segment.

In the Cut Path Determination Problem [4] the trajectories of a set of blowtorchs are defined on a rectangular steel plate in order to produce a pre-defined set of polygonal shaped pieces in minimum time. A piece is produced when its shape is fully traversed by one or more blowtorchs. The blowtorchs have a limited amount of energy to spend and must not traverse the interior of any shape, but they may deslocate above the plate level, which reflects additional elevating and lowering maneuvers times.

In the Parallel Machine Scheduling with Resource Constraints [5], we have a set of distinct jobs that must be executed by a number of identical machines, with limited amount of resources. Each job has a processing time and demands an amount of resources. Each job may not be processed in more than one machine simultaneously and no machine can process more than one job at a time. The objective is minimize the schedule lenght (makespan).

Complexity Results. This section gives a polynomial reduction $CARP \leq OCARP$, which concludes OCARP NP-hardness. Given any CARP instance $G(V, E)$, with M vehicles with capacity D , add $2M$ dummy nodes (V_0) and $2M$ dummy required edges (R_0), with demands $d(r \in R_0) = B > D$, and zero cost. These dummy edges link dummy nodes with the depot v_0 . The vehicles capacity should be increased to $D + 2B$. A new graph $G_1(V_0 \cup V, R_0 \cup E)$ is then formed. This transformation has complexity $O(M)$, and assuming $M \leq |R|$, then it is linear with respect to the size of G . The relationship between the CARP optimal solution L_G^* and the OCARP optimal solution $P_{G_1}^*$ is the following: $L_G^* = P_{G_1}^* \setminus \{R_0 \cup V_0\}$ and $c(L_G^*) = c(P_{G_1}^*)$.

Solution Strategy. This work considers reducing OCARP into CARP and then adopting a CARP heuristic to solve the former. The $OCARP \leq CARP$ reduction is given next. Consider an OCARP instance $G(V, E)$ with M vehicles and capacity D . Add a dummy depot node v_0 and a set N_0 of non-required edges, with costs $c(e \in N_0) = B \gg \max_{e \in E} [c(e)]$, linking v_0 to every node in G . We then form a new graph $G_1(v_0 \cup V, N_0 \cup E)$. This reduction has complexity $O(|V|)$, hence is linear with the size of G . The relationship between the OCARP optimal solution

P_G^* and the CARP optimal solution $L_{G_1}^*$ is the following: $P_G^* = L_{G_1}^* \setminus \{N_0 \cup v_0\}$ and $c(P_G^*) = c(L_{G_1}^*) - 2B$.

Computational Tests. The standard set of CARP instances $*$, which includes 23 *gdb*, 34 *val* and 24 *egl* instances, was used to form the set of OCARP instances, simply by considering the depot a regular node. Lower bounds were obtained by summing the costs of all required edges. Upper bounds were achieved through a path-scanning CARP heuristic [6], after applying the OCARP \leq CARP reduction of the previous section. The overall average deviation from lower bounds were 15,1% (*gdb* 0,61%, *val* 7,75%, *egl* 42,74%). From the set of 81 solutions, 19 solutions from *gdb* were proven optimal.

Conclusions and Future Works. This work introduced a new NP-hard combinatorial problem belonging to the arc routing problems family. It has presented many practical problems that can be modeled as an OCARP. A solving strategy has been given by transforming an OCARP into a CARP. Computational experiments were conducted with a set of 81 OCARP instances, using an efficient path-scanning heuristic. The first lower and upper bounds are given, with some proven optimal solutions. Future works should focus on specific OCARP heuristics design in order to further tighten the upper bounds. Exact algorithms, using column generation and cutting planes approaches, as well as lower bounding procedures, should also be investigated.

Acknowledgment. This work was supported by CNPq (processes 305114/2006-9 and 474099/2006-7).

References

- [1] B. L. Golden, R. T. Wong (1981), "Capacitated arc routing problems", *Networks*, Vol. 11, pp. 305-315.
- [2] H. I. Stern, M. Dror (1979), "Routing electric meter readers", *Computers and Operations Research*, Vol. 6, pp. 209-223.
- [3] J. Wunderlich, M. Collette, L. Levy, L. Bodin (1992), "Scheduling meter readers for Southern California gas company", *Interfaces*, Vol. 22, pp. 22-30.
- [4] L. M. Moreira, J. F. Oliveira, A. M. Gomesa, J. S. Ferreira (2007), "Heuristics for a dynamic rural postman problem", *Computers and Operations Research*, Vol. 34, pp. 3281-3294.
- [5] E. Mokotoff (2001), "Parallel machine scheduling problems: A survey", *Asia - Pacific Journal of Operational Research*, Vol. 18(2), pp. 193-243.
- [6] L. Santos, J. Coutinho-Rodrigues, J. R. Current (2008), "An improved heuristic for the capacitated arc routing problem", *Computers and Operations Research*, doi:10.1016/j.cor.2008.11.005.

* <http://www.uv.es/~belengue/carp.html>

Optimising node coordinates for the shortest path problem

Mirko Maischberger^a

^a*Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze, Italy*

Key words: goal oriented shortest path, coordinate generation

1. Introduction

In the shortest path problem [1] the exploitation of geographical coordinates is a common mean to obtain shorter computational times.

However there are cases in which geographical coordinates are not known and cannot be easily obtained. In other cases, when the objective to be minimized is loosely coupled with the geographical displacement, straight-line distance can simply be a bad choice.

The contribution of this work is the introduction of a new model and a new method for the generation of good artificial coordinates for goal oriented algorithms. The only previous work on the subject of generation of coordinates for shortest path computation is [5].

In [5] coordinates are generated with two methods. The first method they use is the barycentric one, which is very fast in the generation phase, but gives suboptimal results. The second method proposed in the same article, the tailored model, uses a Kamada-Kawai [7] like objective with fewer terms: they only consider members directly connected by an edge. Both models could lead to euclidean distances between nodes greater than their effective distance. This kind of unconstrained approach forces them to scale the generated coordinates down so that the euclidean distance is an admissible heuristic again (w.r.t. the goal oriented algorithm they use). In their experiments, results from the tailored method yield results comparable with real geographical coordinates.

Our contribution is reasonably fast in the preprocessing phase, yields admissible coordinates without the need for rescaling, and, to the best of our knowledge, outperforms the previous techniques halving the average query time.

2. A constrained model for the assignment of coordinates to be used by goal oriented searches

Given a graph $G = (V, E)$ with $|V| = n$, $|E| = m$, where V is the vertex set with coordinates $p_i \in \mathbb{R}^2$, and $E \subseteq V \times V$ is the edge set with non-negative weights $w \in \mathbb{R}^m$, we are interested in finding the best admissible heuristic $h : V \times V \rightarrow \mathbb{R}$ such that, for every pair of nodes, h assigns an estimate distance near to, but not exceeding, the cost of the shortest path between them.

We will refer to $P \in \mathbb{R}^{2 \times n}$ as the coordinates matrix, so that $P = \begin{bmatrix} p_x^T \\ p_y^T \end{bmatrix}$ where p_x and p_y are vectors containing the x and y coordinates of the vertices and $P = [p_1 p_2 \dots p_n]$ so that $p_i \in \mathbb{R}^2$ is the vector containing the coordinates of the i -th vertex.

Let d_{ij} be the cost of the shortest path from $i \in V$ to $j \in V$, our ideal objective will be to have $h(i, j) = d_{ij} \quad \forall (i, j) \in V \times V$ so that the heuristic is exactly accurate for every pair of nodes. This cannot be achieved in the general practice since not all graphs are realizable in a limited number of dimensions [2]. We will try to make $h(i, j)$ closest to d_{ij} while posing $h(j, j) \leq d_{ij}$ to preserve admissibility.

A very simple model that has proved effective in practice can be obtained using an objective that maximizes the weighted sum of the squared euclidean distance between all pairs of vertices.

$$\begin{aligned} \max_P \quad & \sum_{(i,j) \in V \times V, i \neq j} \frac{1}{d_{ij}^2} (\|p_i - p_j\|^2), \\ \text{s.t.} \quad & \|p_i - p_j\|^2 \leq w_{i,j}^2 \quad \forall (i, j) \in E. \end{aligned} \tag{2.1}$$

The resultant model is a convex maximization problem on a convex set, an hard problem [4].

We solved our problem with up to about twenty thousand variables and ten thousand constraints using the Frank-Wolfe method [3, 215-218]. Given a problem in the form

$$\begin{aligned} \max_x \quad & f(x) \\ \text{s.t.} \quad & x \in X, \end{aligned}$$

the Frank-Wolfe method [3, 215-218] is a way to generate a feasible direction $\bar{x}^k - x^k$ that satisfies the descent condition $\nabla f(x^k)'(\bar{x}^k - x^k) < 0$ to be used in the update rule.

The sub-problem obtained applying the method has a linear objective func-

tion, it's quadratically constrained, and it can be solved efficiently with a dedicated solver for QCQP (Quadratically Constrained Quadratic Program):

$$\begin{aligned} \max_P & \nabla f_x(p_x^k)^T p_x + \nabla f_y(p_y^k)^T p_y, \\ \text{s.t.} & \|p_i^k + p_i - p_j^k - p_j\|^2 \leq w_{i,j}^2 \quad \forall (i, j) \in E. \end{aligned}$$

3. Testing environment, computational results and conclusions

In our experiments we used two road networks, Florence and Lisbon (c.f.r. table 5). We also generated a new set of coordinates for the timetable data used in [8; 5] and, as they made their code publicly available, we were able to compare with the original implementation on the original coordinates.

graph	nodes	edges
Florence	4 466	8 932
Lisbon	10 056	11 499
de-org	6 960	931 746

Table 5. The number of nodes and edges of the test graphs.

Sources was built with GNU Compiler Collection version 4.3.2. All the binary was run on a Pentium 4 processor running a 32bit Linux 2.6.27 kernel at 3GHz equipped with 1GiB of main memory.

We have generated new sets of coordinates for all the graphs in table 5. A summary of results, including results from the graph of Lisbon, are presented in table 6 and 7. In particular table 7 is a direct comparison with previous coordinate generation approaches.

graph	ms		expanded nodes	
	original	generated	original	generated
Florence	0.34	0.25	553	378
Lisbon	0.25	0.13	5026	1847

Table 6. Average query response time and average number of nodes expanded by the goal oriented algorithm on 250 000 random queries (the same for all the runs) on the two cities with original and generated coordinates.

In table 7 we used the original code in [8] to compare the average query time and average number of expanded nodes of our coordinates (`de-constrained`), the geographical ones (`de-org`) and the best result from the tailored model presented in [5] (`de-tailored`).

graph	ms	edges
de-org	22.5	20 995
de-tailored	–	20 052
de-constrained	0.4	9 899

Table 7. Average query response time and number of nodes touched by the goal oriented algorithm. *de-org* is the original Germany’s timetable, *de-tailored* reports results from the best of the tailored models from Brandes et al., and *de-constrained* is the same timetable with coordinates generated by our proposed method.

In conclusion we proposed a new model and a new solution approach for the generation of nodes coordinates w.r.t. goal oriented shortest path calculation. This approach does not require any change in the goal oriented algorithms: generated coordinates can be used as a drop-in replacement for geographical ones in existing implementations [6].

References

- [1] AHUJA, R. K., MAGNANTI, T. L., AND ORLIN, J. B. *Network flows: theory, algorithms and applications*. Prentice Hall, New Jersey, 1993.
- [2] ALFAKIH, A. Y. Graph rigidity via euclidean distance matrices. *Linear Algebra and its Applications* 310 (2000), 149–165.
- [3] BERTSEKAS, D. P. *Nonlinear Programming*, second edition ed. Athena Scientific, Belmont, Massachusetts, 1999.
- [4] BOYD, S. P., AND VANDENBERGHE, L. *Convex Optimization*. Cambridge University Press, 2004.
- [5] BRANDES, U., SCHULZ, F., WAGNER, D., AND WILLHALM, T. Generating node coordinates for shortest-path computations in transportation networks. *J. Exp. Algorithmics* 9 (2004), 1.1.
- [6] DELLING, D., SANDERS, P., SCHULTES, D., AND WAGNER, D. Engineering route planning algorithms. *Algorithmics of Large and Complex Networks* (to appear.).
- [7] KAMADA, T., AND KAWAI, S. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.* 31, 1 (1989), 7–15.
- [8] SCHULZ, F., WAGNER, D., AND WEIHE, K. Dijkstra’s algorithm on-line: an empirical case study from public railroad transport. *J. Exp. Algorithmics* 5 (2000), 12.

The Sliding Shortest Path Algorithms

Ramesh Bhandari

*Laboratory for Telecommunications Sciences
8080 Greenmead Drive
College Park, Maryland 20740, USA
rbhandari@ieee.org*

Key words: constrained routing, rerouting, sliding shortest path, algorithm, minimal weight increment, edge cutset, OSPF, networks, optimization, graphs

1. Introduction

The problem of finding optimal administrative link weights in response to a given network demand matrix has received immense attention in the recent past, see, e.g., [1-4]. In this paper, we address the problem of changing the link weights for a specific demand (a single source-destination pair) to be rerouted through a desired link or vertex not already on the shortest path of the given demand. Such rerouting may be necessary in practice to satisfy the quality of service, as perhaps warranted by a service-level agreement, or to meet a special request of an important business customer. For this rerouting, we require that the number of links on which the weights are changed be as small as possible in order to reduce the implementation time of link weight changes within the current OSPF-type network environment [2]. We further require that the weight changes (increments) be as small as possible in order to minimize the number of other shortest paths (routes of other demands) that might be affected. The problem is treated as an uncapacitated problem [3-4]. To our knowledge, it has not been dealt with before.

2. Problem Definition

Let $G = (V, E)$ denote an undirected graph, representing a bidirectional network (e.g., a single autonomous system); V is the set of vertices (or nodes), and E is the set of weighted edges (or links); weights are non-negative integers; routing of traffic demands from one point to another within the network takes place along single shortest paths. We also make the assumption that graph G is biconnected (or

2-connected), i.e., there always exists a simple path, which connects a given pair of vertices, s and t , and includes a given arc pq [5]. A simple path refers to a path in which a given vertex is not visited more than once. Unless otherwise stated, in what follows, the term path refers to a simple path.

Let $SP(s, t)$ denote a shortest path from vertex s to t in the given graph $G = (V, E)$. Let Γ_a denote a set of edges $\in E$, whose weights are incremented to change the shortest path $SP(st)$. Let $x_i, i = 1, \dots, |\Gamma_a|$ denote the corresponding increments. The problem to solve can be stated as:

Given an undirected, weighted graph $G = (V, E)$, and a pair of vertices $s, t \in V$, and an edge $pq \in E$,

$$\text{minimize } |\Gamma_a| \tag{2.1}$$

subject to arc pq (or arc qp) $\in SP(s, t)$,

$$\text{minimize } x_i, i = 1, \dots, M \tag{2.2}$$

subject to arc pq (or arc qp) $\in SP(s, t)$; $M = |\Gamma_M|$, the result obtained in Eq. (1) above.

The problem in Eq. (2.1), which is the primary problem, is to identify the minimum cardinality edge set whose weights should be incremented to alter the given $s - t$ flow path to include edge pq ; the problem in Eq. (2), which is the secondary problem, is a set of multiobjective functions to minimize the weight increments on the edges found in Eq. (1). To solve it, it can be reformulated as a minimization over the sum of the individual increments, x_i ; this corresponds to equal importance of all edges involved in the increment process.

It can be shown that the problem in Eq. (1) is equivalent to the following problem:

$$\text{minimize } |\Gamma_c| \tag{2.3}$$

subject to arc pq (or arc qp) $\in SP(s, t)$,

where $|\Gamma_c|$ is interpreted to mean a set of edges $\in E$, which when cut, alters $SP(s, t)$. The optimal value in Eq. (3) is the same as in Eq. (1), i.e., $M = |\Gamma_M|$.

Problems, Eq. (3) and Eq. (2), are difficult problems, which appear to be NP-hard. In this paper, we solve these problems, using simple heuristics (Section 3).

3. The Sliding Shortest Path Algorithm

In a given, undirected, weighted graph $G = (V, E)$, this algorithm determines (in accordance with a certain cutting criterion) the minimal cardinality set of edges, which, when cut, force the shortest path between vertices s and t to include a given constraint edge pq . Let Γ denote this set. Then the following steps determine Γ :

- (i) Assign $\Gamma = \emptyset$
- (ii) Compute the initial flow path, the shortest path $SP(s, t)$ from s to t . If this path contains edge pq , terminate; otherwise, go to Step 3.
- (iii) Compute the shortest pair of vertex-disjoint paths [6], one path connecting s to p (or q) (call it $SP1$), and the other connecting vertex t to q (or p) (call it $SP2$); the vertex-disjoint path algorithms compute $SP1$ and $SP2$ simultaneously and automatically determine whether $SP1$ is a connection from s to p or s to q ; if $SP1$ turns out to be a connection from s to p , $SP2$ is a connection from t to q , and vice versa.
- (iv) Initialize $i = 1$
- (v) Assign $\Gamma(i) = \emptyset$, where $\Gamma(i)$ denotes the i th set of cut edges.
- (vi) Find the first edge of $SP(s, t)$, which does not overlap with $SP1$ (*cut-edge selection criterion*); cut this edge from the graph; denote this edge by L .
- (vii) Set $\Gamma(i) = \Gamma(i) \cup L$.
- (viii) Compute new $SP(s, t)$ in the trimmed graph.
- (ix) If the new path contains edge pq , terminate; otherwise go to Step 6.
- (x) Set $i = i + 1$
- (xi) If $i < 3$, repeat Steps 5-9 in the original graph, replacing $SP(s, t)$ with $SP(t, s)$, and $SP1$ with $SP2$; otherwise, terminate; $SP(t, s)$ denotes the shortest path from t to s , which is taken to be $SP(s, t)$ in the reverse order.
- (xii) If $|\Gamma(1)| < |\Gamma(2)|$, set $\Gamma = \Gamma(1)$; if $|\Gamma(2)| < |\Gamma(1)|$, set $\Gamma = \Gamma(2)$; if $|\Gamma(1)| = |\Gamma(2)|$, set $\Gamma = \Gamma(1)$ or $\Gamma(2)$.

If the $SP(s, t)$ path already contains edge pq , the algorithm terminates, returning $\Gamma = \emptyset$; otherwise it performs two runs of an iterative process. The iterative process consists of trimming the graph by cutting one edge at a time and recomputing the shortest path after each edge cut until the shortest path between s and t slides over the given constraint edge pq . The edge to cut is determined by an edge-selection criterion (Step 6). In the first run ($i = 1$) of the iterative process, path $SP1$ acts as the reference path for the cut-edge selection, and in the second run ($i = 2$), path $SP2$ acts as the reference path for edge-cut selection (Step 6). The two runs of the iterative process of the algorithm yield two cut-sets, $\Gamma(1)$ and $\Gamma(2)$, which can be different. In Step 12, the desired set Γ is identified with the one, which has fewer edges. If there is a tie, Γ is set equal to either of the two. Below we state some theorems without proving them:

Theorem 1: In a given graph $G = (V, E)$, the shortest path from s to t , including edge pq , is comprised of the edge pq (traversed along the arc pq or arc qp) and the shortest pair of vertex-disjoint paths, one path connecting s to p (or q) and the other connecting q (or p) to t .

Theorem 2: In the given algorithm, the process of cutting one edge at a time until the shortest path from s to t slides over edge pq does not disconnect t from s , i.e., the algorithm always converges to a feasible solution.

Theorem 3: Path $SP(s, t)$ after termination of the algorithm comprises paths $SP1$, $SP2$, and edge pq (arc pq or qp)

Once a solution is obtained, using the above heuristic, the problem, Eq. (2), is solved as follows:

Instead of cutting the first non-overlapping edge (see Step 6 of the algorithm), increment its weight:

$$x_j = l(P_f) - l(P_j) + e, \quad (3.4)$$

where x_j is the weight increment for the first non-overlapping edge encountered in the j th iteration of Steps 6-9 of the algorithm; $l(P_j)$ is the length of the corresponding shortest path ($SP(s, t)$ or $SP(t, s)$, as the case may be); $l(P_f)$ is the length of the final desired path; $l(P_f) = l(SP1) + l(SP2) + w_{pq}$, where w indicates an edge weight; e is an infinitesimally small positive number. For the integral weights, $e = 1$. The increment x_j defined above is the minimal amount needed to make path P_j greater (in length) than the desired path P_f ; as a result, the latter becomes the shortest path in the final (modified) graph.

4. Discussion

The Sliding Shortest Path Algorithm is presented as a heuristic for the difficult problem, Eq. (1). Its extension via Eq. (4) then solves the problem, Eq. (2). The algorithm is easily extended to the case of rerouting over a specified vertex by collapsing the constraint edge into a single vertex. The efficiency of the algorithm is determined by the number of times the Dijkstra algorithm has to be run. In the worst-case scenario, where almost all the edges have to be cut, the efficiency is i) $O(|V|^2)\rho$ for dense graphs (almost fully connected), 2) $O(|V|)\rho$ for sparse graphs (almost tree-like), where ρ denotes the efficiency of the Dijkstra algorithm. ρ is $O(|V|^2)$, and there are improvements due to more efficient implementations [7]. The heuristic is therefore very fast and its computer code has successfully run on graphs consisting of as many as 200,000 vertices.

The edge cuts in the algorithm emanate from the reference path $SP1$ or $SP2$,

depending upon whether it is the first run or the second. Larger the degree of the vertices of the reference path, larger the number of edges that would be cut on the average. As a result, $|\Gamma(1)|$ and $|\Gamma(2)|$ can be significantly different, depending upon the densities (degrees of vertices) of the subgraphs the paths $SP1$ and $SP2$ lie in. Furthermore, based on our initial theoretical studies of very small graphs (10 vertices or so), we expect the algorithm to perform well (i.e., give a solution close to the true solution) in graphs with approximately uniform density, but, in those (uncommon) instances, where the density of the graph in the region between the reference paths, $SP1$ and $SP2$, may drop sharply, we expect the performance to degrade because the algorithm looks only along the paths $SP1$ and $SP2$ for cuts, and not away from them. One way to assess the performance of this heuristic computationally is by comparing its results directly with the optimal solutions. The optimal solution to the problem can be obtained in the following way: try all possible cutsets, starting with cutsets of cardinality unity, then cutsets of cardinality two, and so on (at least one edge in the cutset always belonging to the initial path, $SP(s, t)$) until the desired shortest path is obtained. Such a method, however, quickly becomes exponential in run time. In the detailed version of the paper and the talk, we will provide numerical results of the algorithms's performance, keeping in mind the inefficiency of the optimal solution method.

References

- [1] B. Fortz and M. Thorup, *Internet Traffic Engineering by Optimizing OSPF Weights*, Proc. of the 19th IEEE INFOCOM, Tel-Aviv, Israel (2000) pp.519-528
- [2] B. Fortz and M. Thorup, *Optimizing OSPF/IS-IS Weights in a Changing World*, IEEE Journal on Selected Areas in Communications, 20 (2002) pp 756-767
- [3] W. Ben-Ameur and E. Gourdin, *Internet Routing and Related Topology Issues*, SIAM Journal of Discrete Mathematics, 17(1) (2003) pp. 18-49
- [4] A. Bley, *Finding Small Administrative Lengths for Shortest Path Routing*, Proc. of 2nd International Network Optimization Conference, Lisbon, Portugal (2005) pp.121-128.
- [5] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, 1993.
- [6] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*, Kluwer Academic Publishers, 1998.
- [7] M. Gondran and M. Minoux, *Graphs and Algorithms*, John Wiley, 1990.

Quadratic Programming

Lecture Hall A

Tue 2, 15:45–16:45

A polynomial-time recursive algorithm for some unconstrained quadratic optimization problems

Walid Ben-Ameur,^a José Neto^a

^a*Institut TELECOM, TELECOM & Management SudParis, CNRS Samovar,
9, rue Charles Fourier, 91011 Evry, France
{Walid.Benameur, Jose.Neto}@it-sudparis.eu*

Key words: arrangements, unconstrained quadratic programming, complexity

1. Introduction

Consider a quadratic function $q : \mathbb{R}^n \rightarrow \mathbb{R}$ given by: $q(x) = x^t Q x$, with $Q \in \mathbb{R}^{n \times n}$. An unconstrained $(-1, 1)$ -quadratic optimization problem can be expressed as follows:

$$(QP) Z^* = \min\{q(x) \mid x \in \{-1, 1\}^n\},$$

where $\{-1, 1\}^n$ denotes the set of n -dimensional vectors with entries either equal to 1 or -1 . We consider here that the matrix Q is symmetric and given by its spectrum, i.e. the set of its eigenvalues and associated unit pairwise orthogonal eigenvectors.

Problem (QP) is a classical combinatorial optimization problem with many applications, e.g. in statistical physics and circuit design [2; 8; 10]. It is well-known that any $(0,1)$ -quadratic problem expressed as: $\min\{x^t A x + c^t x \mid x \in \{0, 1\}^n\}$, $A \in \mathbb{R}^{n \times n}$, $c \in \mathbb{R}^n$, can be formulated in the form of problem (QP) and conversely [9; 4].

The contribution of this work is 3-fold:

- (i) We slightly extend the known polynomially solvable cases of (QP) to when the matrix Q has fixed rank and the number of positive diagonal entries is $\mathcal{O}(\log(n))$.
- (ii) We introduce a new (to our knowledge) polynomial-time algorithm for solving problem (QP) when it corresponds to such a polynomially solvable case.
- (iii) Preliminary experiments indicate that the proposed method may be computationally efficient. [7].

2. Properties of optimal solutions for peculiar instances of (QP)

Let us firstly introduce some notation to be used hereafter. The eigenvalues of the matrix Q will be noted $\lambda_1(Q) \leq \lambda_2(Q) \leq \dots \leq \lambda_n(Q)$ (or more simply $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ when clear from the context) and the corresponding unit (in Euclidean norm) and pairwise orthogonal eigenvectors: v_1, \dots, v_n . The j -th entry of the vector v_i is noted v_{ij} . Given some set of vectors $a_1, \dots, a_q \in \mathbb{R}^n$, $q \in \mathbb{N}$, we note $Lin(a_1, \dots, a_q)$ the subspace spanned by these vectors.

In this section we shall make the following assumptions on the matrix Q :

- (i) Q has rank $p \leq n$,
- (ii) Q has nonpositive diagonal entries only, and
- (iii) Q is given by its set of rational eigenvalues and eigenvectors: $Q = \sum_{i=1}^p \lambda_i v_i v_i^t$.

Any optimal solution y^* to the problem $\min_{y \in \{-1, 1\}^n} y^t Q y$ can be shown to satisfy the following implication:

$$\sum_{i=1}^p \lambda_i \alpha_i v_{ij} > 0 \Rightarrow y_j^* = -1 \quad (2.1)$$

And analogously:

$$\sum_{i=1}^p \lambda_i \alpha_i v_{ij} < 0 \Rightarrow y_j^* = 1. \quad (2.2)$$

From this simple property we can namely show that in order to find an optimal solution of problem (QP) , it is sufficient to enumerate over all vectors $y \in \{-1, 1\}^n$ for which there exists a vector $\alpha \in \mathbb{R}^p$ such that $y_j = -\text{sign}(\sum_{i=1}^p \lambda_i \alpha_i v_{ij})$ (or equivalently $y_j = \text{sign}(\sum_{i=1}^p \lambda_i \alpha_i v_{ij})$, see hereafter), $\sum_{i=1}^p \lambda_i \alpha_i v_{ij} \neq 0, \forall j \in \{1, \dots, n\}$, with $\text{sign}(x) = 1$ if $x > 0$ and -1 if $x < 0$. In the next section we focus on finding such a set of vectors.

3. Determining cells in an arrangement of n hyperplanes

Let $v_1, \dots, v_p \in \mathbb{R}^n$ denote p independent vectors. Let $V \in \mathbb{R}^{n \times p}$ denote the matrix whose columns correspond to the vectors v_1, \dots, v_p and V_i the i -th row of V . From this set of vectors we define n hyperplanes in \mathbb{R}^p : $H_j = \{\alpha \in \mathbb{R}^p \mid V_j \cdot \alpha = 0\}$ with $j \in \{1, \dots, n\}$. Then we can notice that there is a one-to-one correspondence between the set of vectors in $\{-1, 1\}^n$ for which there exists a vector $\alpha \in \mathbb{R}^p$ such that $y_j = \text{sign}(\sum_{i=1}^p \alpha_i v_{ij})$, with $\sum_{i=1}^p \alpha_i v_{ij} \neq 0, \forall j \in \{1, \dots, n\}$ and the cells (i.e. the full dimensional regions) in \mathbb{R}^p of the hyperplane arrangement $\mathcal{A}(H)$ that is defined by the family of hyperplanes $(H_j)_{j=1}^n$. To see this just interpret the sign vector y as the position vector of the corresponding cell c w.r.t. an orientation of

the space by the vector V_j : cell c is *above* hyperplane H_j iff $y_j > 0$ and *under* otherwise.

For a general arrangement in \mathbb{R}^p that is defined by n hyperplanes (see e.g. [6; 11] for further elements on arrangements), the number of cells is upper bounded by $\sum_{i=0}^p \binom{n}{i}$ (which is in $\mathcal{O}(n^p)$). (For a proof we refer the reader e.g., to Lemma 1.2 in [6]). In our case, since all the hyperplanes considered contain the origin (i.e. the arrangement is *central*), this number reduces to $\mathcal{O}(n^{p-1})$ (see Section 1.7 in [6]).

We have introduced [3] a simple procedure with time complexity lying between the time complexity of the incremental algorithm [6; 5] (in $\mathcal{O}(n^{p-1})$) and that of the reverse search algorithm [1; 7] (in $\mathcal{O}(n LP(n, p) C)$ where C denotes the number of cells in $\mathcal{A}(H)$ and $LP(n, p)$ is the time needed to solve a linear program with n inequalities and p variables) in order to compute a set of vectors in $\{-1, 1\}^n$ corresponding to a description of a set containing the cells of the arrangement $\mathcal{A}(H)$. Space complexity can be shown to be polynomially bounded by the output size. To our view the interest of the proposed method by comparison with the former ones is 2-fold:

- (i) it is very easy to understand and implement,
- (ii) computationally, by using proper data structures (to be specified latter) we could solve instances of the same magnitude as the ones reported in [7], without parallelization and substantially improved computation times.

The basic principle of the proposed method may be expressed as follows. Given some integer $q \in \{1, \dots, n\}$, let $\mathcal{B}^q(H)$ denote the arrangement in the subspace $\{\alpha \in \mathbb{R}^p \mid V_q \alpha = 0\}$ that is defined by the hyperplanes (in \mathbb{R}^{p-1}) $\{H_j \cap H_q \mid j \in \{1, \dots, n\} \text{ and } j \neq q\}$. Any cell of $\mathcal{B}^q(H)$ (which is a region of dimension $p - 1$) corresponds to a facet of exactly two cells of the arrangement $\mathcal{A}(H)$ i.e. one on each side of the hyperplane H_q . The other cells of $\mathcal{A}(H)$ i.e. those not intersecting H_q , are cells of the arrangement $\mathcal{C}^q(H)$ in \mathbb{R}^p which is defined by the $n - 1$ hyperplanes $\{H_j \mid j \in \{1, \dots, n\} \text{ and } j \neq q\}$. Since each cell of $\mathcal{A}(H)$ intersects at least one of the hyperplanes $(H_j)_{j=1}^n$, it follows that all the cells of $\mathcal{A}(H)$ can be derived from the ones of all the arrangements $\mathcal{B}^q(H)$, $q = 1, \dots, n$. A recursive use of this argument leads to the generation of all the cells of $\mathcal{A}(H)$.

From a complexity study of the proposed method we can show the following result.

Theorem 3.1. For a fixed integer $p \geq 2$, if the matrix Q (given by its nonzero eigenvalues and associated eigenvectors) has rank at most p and $\mathcal{O}(\log(n))$ positive diagonal entries, then problem (QP) can be solved in strongly polynomial time.

4. Conclusion

We propose a new (up to our knowledge) approach for solving in polynomial time some unconstrained quadratic optimization problems. Preliminary computational results illustrate that the recursive procedure briefly presented here can be a valuable approach on some instances by comparison with a reverse search w.r.t. computation times.

Further computational studies are under work and could involve a parallelization of the code in order to deal with larger instances.

References

- [1] D. Avis and K. Fukuda: Reverse search for enumeration. *Discrete Applied Mathematics* **65**, 21-46 (1996)
- [2] F. Barahona and M. Grötschel and M. Jünger and G. Reinelt: An application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design. *Operations Research* **36**, 493–513 (1988)
- [3] W. Ben-Ameur and J. Neto: Spectral bounds for the maximum cut problem. Institut TELECOM, TELECOM & Management SudParis, Evry. Rapport de recherche 08019-RS2M (2008)
- [4] C. De Simone: The cut polytope and the Boolean quadric polytope. *Discrete Mathematics* **79**, 71-75 (1990)
- [5] H. Edelsbrunner and J. O'Rourke and R. Seidel: Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing* **15**, 341-363 (1986)
- [6] H. Edelsbrunner: *Algorithms in Combinatorial Geometry*. Springer (1987)
- [7] J.A. Ferrez and K. Fukuda and T.M. Lieblich: Solving the fixed rank convex quadratic maximization in binary variables by a parallel zonotope construction algorithm. *European Journal of Operational Research* **166**(1), 35-50 (2005)
- [8] M. Grötschel and M. Jünger and G. Reinelt: Via minimization with pin preassignment and layer preference. *Zeitschrift für Angewandte Mathematik und Mechanik*, **69**, 393–399 (1989)
- [9] P. Hammer: Some network flow problems solved with pseudo-Boolean programming. *Operations Research* **32**, 388-399 (1965)
- [10] R.Y. Pinter: Optimal Layer Assignment for Interconnect. *J. VLSI Comput. Syst.* **1**, 123–137 (1984)
- [11] G. M. Ziegler: *Lectures on Polytopes*, Graduate Texts in Mathematics **152**. Springer-Verlag New-York (1995)

Finding tight RLT formulations for Quadratic Semi-Assignment Problems

Ingmar Schüle,^a Hendrik Ewe,^a Karl-Heinz Küfer^a

^a*Fraunhofer Institute for Industrial Mathematics, Kaiserslautern, Germany*
ingmar.schuele@itwm.fraunhofer.de

Key words: Quadratic Semi-Assignment Problem, Reformulation Linearization Technique

1. Introduction

A wide range of combinatorial optimization problems can be formulated as Quadratic Semi-Assignment Problems (QSAP). The QSAP usually describes the assignment of resources to consumers and is a generalization of the widely studied Quadratic Assignment Problem (QAP).

Not only are both QAP and QSAP hard to solve in theory (the decision problems are NP-hard) but also in practice today's computer systems are often unable to solve even small instances to optimality. In order to get a lower bound for the solution, a Reformulation Linearization Technique (RLT) can be applied resulting in a Linear Program that is much easier to solve, cf. [1] and [2].

In this paper we present a graph-theoretical analysis of the RLT applied to the QSAP. A class of graphs is constructed the size of which can be proven to be minimal. It is then used to determine the level of the RLT necessary for a tight formulation of the problem. A tight formulation here means that for all possible b_{ij} and c_{ijkl} , the optimal objective function value of the RLT- t formulation equals the optimal objective function value of the QSAP.

2. RLT formulation of the Quadratic Semi-Assignment Problem

Given the index sets $M = \{1, \dots, m\}$ and $N_i = \{1, \dots, n_i\} \forall i \in M$, the Quadratic Semi-Assignment Problem is to assign to each $i \in M$ exactly one ele-

ment $j \in N_i$. It is defined by the Mixed Integer Program (MIP)

$$\begin{aligned} \min & \sum_{i=1}^m \sum_{j=1}^{n_i} b_{ij} x_{ij} + \sum_{i=1}^{m-1} \sum_{k=i+1}^m \sum_{j=1}^{n_i} \sum_{l=1}^{n_k} c_{ijkl} x_{ij} x_{kl} \\ \text{s.t.} & \sum_{j=1}^{n_i} x_{ij} = 1 \quad \forall i \in M, \\ & x_{ij} \in \{0, 1\}. \end{aligned}$$

Note that we focus on the symmetric form where $c_{ijkl} = c_{klij}$. When we apply the level-1 Reformulation Linearization Technique we introduce new variables $y_{ijkl} = x_{ij} \cdot x_{kl}$ and some additional constraints. After relaxing all 0/1-variables we get the linearized formulation

$$\begin{aligned} \min & \sum_{i=1}^m \sum_{j=1}^{n_i} b_{ij} x_{ij} + \sum_{i=1}^{m-1} \sum_{k=i+1}^m \sum_{j=1}^{n_i} \sum_{l=1}^{n_k} c_{ijkl} y_{ijkl} \\ \text{s.t.} & \sum_{j=1}^{n_i} x_{ij} = 1 \quad \forall i \in M, \\ & \sum_{j=1}^{n_i} y_{ijkl} = x_{kl} \quad \forall i, k \in M (i < k), l \in N_k, \\ & \sum_{l=1}^{n_k} y_{ijkl} = x_{ij} \quad \forall i, k \in M (i < k), j \in N_i, \\ & x_{ij} \in [0, 1], \quad y_{ijkl} \in [0, 1]. \end{aligned}$$

3. Level- t RLT

To obtain the general level- t RLT formulation for the QSAP, we add new variables and constraints to the level- $(t-1)$ formulation. Accordingly, the new constraints are of the form

$$\sum_{j_k=1}^{n_k} \vartheta_{i_1 j_1, \dots, i_{t+1} j_{t+1}}^{(t+1)} = \vartheta_{i_1 j_1, \dots, i_{k-1} j_{k-1}, i_{k+1} j_{k+1}, \dots, i_{t+1} j_{t+1}}^{(t)},$$

$$\forall k \in \{1, \dots, t+1\}, \forall i_1, \dots, i_{t+1} \in M (i_1 < \dots < i_{t+1}), \forall j_s \in N_{i_s}, s \in M \setminus \{k\}.$$

To complete the recursive definition of the RLT- t formulation, we set

$$\vartheta^{(0)} = 1, \quad \vartheta_{i_1 j_1}^{(1)} = x_{i_1 j_1} \quad \text{and} \quad \vartheta_{i_1 j_1, i_2 j_2}^{(2)} = y_{i_1 j_1 i_2 j_2}.$$

Each feasible solution of the RLT can be transformed into a solution-graph. In this duality, a variable x_{ij} corresponds to a vertex v_{ij} and the linearized variables y_{ijkl} with $y_{ijkl} > 0$ correspond to edges $e_{ijkl} = \{v_{ij}, v_{kl}\}$. By induction it is easy to prove

that a solution-graph of a level- t RLT formulation contains at least one $(t + 1)$ -clique. Furthermore any solution-graph that contains an m -clique has a subgraph that corresponds to a solution of the QSAP. Thus it follows that there is a finite t for which the optimal objective function value of both the RLT and the original MIP are identical. Note that any solution of the original MIP is also valid for the RLT formulation since the additional constraints solely arise from multiplying both sides of already existing constraints by certain variables.

4. Tightness of the RLT formulation

In this section we present a minimal graph G_{RLT}^t and a corresponding ϑ -variable assignment that help to determine the minimal RLT-level that is necessary for a tight RLT formulation. This graph satisfies the RLT-constraints up to level t but does not contain a clique of size $t + 2$. We say that a graph satisfies the RLT-constraints if there exists a corresponding ϑ -variable assignment that satisfies the constraints.

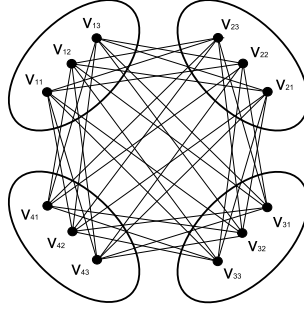


Fig. 1. Graph G_{RLT}^2 not containing a 4-clique.

We define the graph $G_{RLT}^t = (V_{RLT}^t, E_{RLT}^t)$ by

$$V_{RLT}^t = \{v_{ij} : i \in \{1, \dots, t + 2\}, j \in \{1, \dots, t + 1\}\},$$

$$E_{RLT}^t = \{e_{ijkl} : i, k \in \{1, \dots, t + 2\} (i < k), j, l \in \{1, \dots, t + 1\} (j \neq l)\}.$$

The corresponding variable assignment that satisfies the RLT-constraints is

$$\vartheta_{ij}^{(1)} = x_{ij} = \frac{1}{t + 1}, \text{ if } v_{ij} \in V_{RLT}^t,$$

$$\vartheta_{i_1 j_1, \dots, i_v j_v}^{(v)} = \prod_{s=1}^v \frac{1}{(t + 2 - s)}, \quad \forall v \in \{2, \dots, t + 1\},$$

$$\text{if } \exists k, l \in \{1, \dots, v\} (k < l) : e_{i_k j_k i_l j_l} \in E_{RLT}^t,$$

and zero for all other variables. The following theorem deals with the minimality of G_{RLT}^t in the context of the given problem.

Theorem 4.1. G_{RLT}^t is the minimal graph that satisfies all RLT-constraints up to level t and that contains no $(t + 2)$ -clique.

We omit the full proof due to space limitations. The main ideas are the following four steps:

- show that G_{RLT}^t satisfies the RLT- t -constraints,
- show that G_{RLT}^t contains no $(t + 2)$ -clique,
- show that for any graph that satisfies the RLT- t -constraints and that contains no $(t + 2)$ -clique $\exists S \subseteq M$, $|S| \geq t + 2$, such that $\forall i \in S : n_i \geq t + 1$,
- show that there is no graph with less edges than $|E_{RLT}^t|$ that satisfies the RLT- t -constraints and that contains no $(t + 2)$ -clique.

As a result from Theorem 1 we directly obtain for each M and the corresponding sets N_i , $i \in M$, the smallest number t_{\min} for which the RLT- t formulation is tight. If the sets N_i are ordered according to their size ($n_1 < \dots < n_m$), t_{\min} is defined by

$$t_{\min} = \min\{t \in \mathbb{N} : n_{t+2} < t + 1\}.$$

5. Conclusion and future work

In this paper we presented some theoretical results on the tightness of the RLT- t formulation for the QSAP. The gained insights can be used e.g. in a stepwise elimination process of possible occurrences of the minimal graphs G_{RLT}^t in the problem formulation. A first implementation of such an algorithm showed promising results compared to established approaches. As future work we plan to transfer our results to the QAP.

References

- [1] Adams, W.P., Guignard, M., Hahn, P.M. and Hightower, W.L. A level-2 reformulation-linearization technique bound for the quadratic assignment problem, *European Journal of Operational Research*, 180, 3, p. 983–996, 2007.
- [2] Hahn, P.M., Kim, B., Guignard, M., Smith, J.M. and Zhu, Y., An algorithm for the generalized quadratic assignment problem *Comput. Optim. Appl.* 40, 3, p. 351–372, Kluwer Academic Publishers, Norwell, MA, USA, 2008.

Trees

Lecture Hall B

Tue 2, 15:45–16:45

Colored trees in edge-colored graphs

A. Abouelaoualim,^a V. Borozan,^a Y. Manoussakis,^a
C. Martinhon,^b R. Muthu,^a R. Saad^a

^aUniversity of Paris-XI, Orsay LRI, Bât. 490, 91405 Orsay Cedex, France

^bInst. of Comp. / Fluminense Federal University, Niterói, Brazil

Key words: Colored spanning trees, acyclic graph, c -edge-colored graph, nonapproximability bounds, polynomial algorithms, NP-completeness

1. Introduction

The study of problems modeled by edge-colored graphs has given rise to important developments during the last few decades. For instance, the investigation of spanning trees for graphs provide important and interesting results both from a mathematical and an algorithmic point of view (see for instance [1]). From the point of view of applicability, problems arising in molecular biology are often modeled using colored graphs, i.e., graphs with colored edges and/or vertices [6]. Given such an edge-colored graph, original problems translate to extracting subgraphs colored in a specified pattern. The most natural pattern in such a context is that of a proper coloring, i.e., adjacent edges having different colors. Refer to [2; 3; 5] for a survey of related results and practical applications. Here we deal with some colored versions of spanning trees in edge-colored graphs. In particular, given an edge-colored graph G^c , we address the question of deciding whether or not it contains properly edge colored spanning trees or rooted edge-colored trees with a given pattern.

Formally, let $I_c = \{1, 2, \dots, c\}$ be a given set of colors, $c \geq 2$. Throughout, G^c denotes an edge-colored simple graph, where each edge is assigned some color $i \in I_c$. The vertex and edge-sets of G^c are denoted $V(G^c)$ and $E(G^c)$, respectively. The *order* of G^c is the number n of its vertices. A subgraph of G^c is said to be *properly edge-colored* if any two of its adjacent edges differ in color. A *tree* in G^c is a subgraph such that its underlying non-colored graph is connected and acyclic. A *spanning tree* is one covering all vertices of G^c . From the earlier definitions, a *properly edge-colored tree* is one such that no two adjacent edges are on a same color. A tree T in G^c with fixed root r is said to be *weakly properly edge-colored* if any path in T , from the root r to any leaf is a properly edge-colored one. To facilitate discussions, in the sequel a properly edge-colored (weakly properly edge-colored) tree will be called a *strong (weak) tree*. Notice that in the case of weak

trees, adjacent edges may have the same color, while this may not happen for strong trees. When these trees span the vertex set of G , they are called *strong spanning tree* (SST) and *weak spanning tree* (WST).

Here we prove that the problems of finding SST and WST in colored graphs are both NP-complete. The problem of SST remains NP-complete even when restricted to the class of edge-colored complete graphs. We present nonapproximability results by considering the optimization versions of these problems. We provide polynomial time algorithms for these problems on the important class of colored acyclic graphs, i.e. graphs without properly edge-colored cycles. We also present an interesting graph theoretic characterization of colored complete graphs which have SSTs.

2. NP-completeness and nonapproximability

The SST problem is NP-complete for G^c if c is a constant, because it generalizes the degree-constrained spanning tree problem, which extends the Hamiltonian path problem. Here, the degree constraint of a node v is the number of different colors used on its incident edges. The next result is a stronger one, and is proved using a kind of self-reduction from the SST problem on a constant number of colors.

Theorem 2.1. The SST is NP-complete even for $c = \Omega(n^2)$.

The hardness result for WST stated below is obtained by a reduction from the 3-SAT problem.

Theorem 2.2. Given a 2-edge colored graph $G^c = (V, E^c)$ and a specified vertex r of V , it is NP-complete to determine if G^c has a WST rooted at r .

We view the optimization versions of these problems as finding the corresponding trees covering as many vertices as possible. The following results on nonapproximability bounds are obtained by the gap-reduction technique using the MAX-3-SAT problem.

Theorem 2.3. The maximum weighted tree (MWT) problem is nonapproximable within $63/64 + \epsilon$ for $\epsilon > 0$ unless $P = NP$.

Theorem 2.4. The maximum strong tree (MST) problem is nonapproximable within $53/54 + \epsilon$ for $\epsilon > 0$ unless $P = NP$.

3. Colored trees in acyclic edge-colored graphs

In this section, we present results demonstrating that the SST and WST problems can be solved efficiently when restricted to the class of edge colored-acyclic graphs. We present a proof sketch and an algorithm for the SST problem on colored acyclic complete graphs. The case of SST on general colored acyclic graphs is similar, but more involved and appears in a longer version of the paper. We do not provide the details of the WST problem either, due to space constraints.

An important tool we use is a theorem due to Yeo ([7],[4]), which states that every colored acyclic graph has a vertex v , such that the edges between any component \mathcal{C}_i of $G \setminus v$ and v are monochromatic. We call such a vertex a *yeo-vertex*. If in addition, the colors of the edges between v and the various components obtained

by deleting it are all distinct, we call it a *rainbow yeo-vertex*. It is easy to see that a colored acyclic graph with a nonrainbow yeo-vertex has no SST.

For the rest of this section, we assume we are dealing with colored acyclic complete graphs (K_n^c -acyclic). We compute a partial ordering of the vertices and construct the SST by incorporating the vertices in the reverse of this order. The first *block* consists of *all* the yeo-vertices of the graph. They induce a monochromatic clique and the edges between this group of vertices and the rest of the graph are also monochromatic with the same color. We repeat this procedure iteratively, by considering the residual (also) acyclic complete graph obtained by deleting these vertices from the original graph. The second block is also monochromatic but with a *different* color.

We use k to denote the number of blocks in the above partial order and the blocks themselves are denoted $\mathcal{B}_1, \dots, \mathcal{B}_k$. We use c_i to denote the associated color of block \mathcal{B}_i . Recall that the color associated with successive blocks differ. We denote the total number of vertices in the blocks $\mathcal{B}_i, \dots, \mathcal{B}_k$ by t_i and the number of such vertices whose associated color is l by t_i^l . We now state a lemma, which given an acyclic edge colored complete graph determines whether or not it contains an SST.

Lemma 3.1. (SST-Complete Acyclic) An acyclic edge colored complete graph has an SST iff

- (i) Last block \mathcal{B}_k has two vertices, and
- (ii) for each $i < k$,
 - IF block \mathcal{B}_i has the same color as the last block \mathcal{B}_k , THEN
$$t_i^{c_i} - 2 \leq \frac{t_i}{2}.$$
 - ELSE $t_i^{c_i} \leq \frac{t_i}{2}$.

We now describe our algorithm to construct the SST. It is based on the previous lemma. Its running time is $O(n^2)$, as it can be implemented by modifying the basic Breadth-First-Search (BFS) procedure.

Algorithm 1 SST for K_n^c -acyclic

```
1: compute the order described above
2: if last block  $\mathcal{B}_k$  has more than two vertices then return "No SST"
3: if last block  $\mathcal{B}_k$  has two vertices then connect the two vertices of  $\mathcal{B}_k$  to get an
   initial Strong Tree
4: for  $i = k - 1$  to 1 do
5:   if condition 2 of Lemma 3.1 is true then
6:     join the vertices of  $\mathcal{B}_i$  as leaves, to distinct vertices already incorporated
       in the tree which have not used an edge of color  $c_i$  in the partial strong
       tree obtained in the previous iteration.
7:   else
8:     return "NO SST"
9:   end if
10: end for
11: return the SST
```

To conclude this section, we now state our more general result.

Theorem 3.2. The SST and WST problems can be solved efficiently for acyclic colored graphs.

4. Properly edge-colored spanning trees in edge-colored complete graphs

The SST problem remains hard even when stringently restricted, as the following result states. The hardness is proved by a reduction from the SST problem in general graphs.

Theorem 4.1. The SST is NP-complete for complete graphs K_n^c , colored with $|c| \geq 3$ colors.

Observe, that for the case $c = 2$, the SST problem reduces to the Hamiltonian Path problem, which is known to be polynomial [3]. Notice also that the WST problem is trivial in K_n^c as any spanning star is a WST. Concerning SST, we provide below a graph-theoretic characterization for edge-colored complete graphs K_n^c which have SSTs. This characterization is interesting from a mathematical point of view, but the implied conditions cannot be computed in polynomial time, in view of the hardness result above.

Theorem 4.2. Assume that the vertices of K_n^c are covered by a strong tree T and a set of properly edge-colored cycles, say C_1, C_2, \dots, C_k all these components being pairwise vertex-disjoint in K_n^c . Then K_n^c has a strong spanning tree.

References

- [1] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, Network Flows: Theory, Algorithms and Applications, *Prentice Hall*, 1993.
- [2] J. Bang-Jensen, G. Gutin, Digraphs: Theory, Algorithms and Applications, *Springer*, 2002.
- [3] A. Benkour, Y. Manoussakis, V. T. Paschos, R. Saad, On the Complexity of

Some Hamiltonian and Eulerian Problems in Edge-colored Complete Graphs
RAIRO - Operations Research, **30**, 417-438, 1996.

- [4] J. Grossman and R. Häggkvist, properly edge-colored Cycles in Edge-Partitioned Graphs, *Journal of Combinatorial Theory, Series B*, **34** (1983) 77-81.
- [5] Y. Manoussakis, properly edge-colored Paths in Edge-Colored Complete Graphs, *Discrete Applied Mathematics*, **56** (1995) 297-309..
- [6] P. A Pevzner, DNA Physical Mapping and Properly Edge-colored Eulerian Cycles in Colored Graphs, *Algorithmica*, **13** (1995) 77-105 .
- [7] A. Yeo, A Note on Alternating Cycles in Edge-colored Graphs, *Journal of Combinatorial Theory, Series B* **69** (1997) 222-225 .

Intermediate Trees [★]

Kathie Cameron,^a Joanna Fawcett^b

^a*Université Pierre et Marie Curie (Paris VI), Équipe Combinatoire et Optimisation, Paris, France*

kcameron@wlu.ca

^b*Department of Pure Mathematics, University of Waterloo, Waterloo, Canada*

joannafawcett@gmail.com

Key words: spanning tree, vertex degree, pivot algorithm

The intermediate spanning tree problem is: Given two distinct spanning trees, T and T' , of a graph G , is there another spanning tree, T'' , of G such that the degree of each vertex in T'' is between its degree in T and its degree in T' ? More precisely, is there a spanning tree T'' of G such that for each vertex v of G , either $\deg_T(v) \leq \deg_{T''}(v) \leq \deg_{T'}(v)$ or $\deg_{T'}(v) \leq \deg_{T''}(v) \leq \deg_T(v)$, where $\deg_H(v)$ denotes the degree of vertex v in H ? Such a tree T'' is called an *intermediate tree* of T and T' .

The intermediate spanning tree problem is NP-hard in general.

A theorem of Ken Berman [1] implies that if T and T' are edge-disjoint and don't have the same degree at each vertex, then an intermediate tree T'' exists. Cameron and Edmonds [2] gave an algorithm which finds the intermediate tree in this case.

Generally, some pairs of spanning trees in a graph will have an intermediate tree and others won't. For example, in the cycle C_4 on four vertices, v_1, v_2, v_3, v_4, v_1 , there are four spanning trees, and each is a hamiltonian path: T_1 from v_1 to v_4 , T_2 from v_2 to v_1 , T_3 from v_3 to v_2 , and T_4 from v_4 to v_3 . Trees T_1 and T_3 have T_2 and T_4 as intermediate trees, but T_1 and T_2 have no intermediate tree.

We have characterized the graphs in which no pair of spanning trees has an intermediate tree. One such graph is the complete graph on three vertices, K_3 , which has three spanning trees, no two of which have an intermediate tree. However, as

[★] Research supported by the Natural Sciences and Engineering Research Council of Canada.

the following theorem shows, K_3 is essentially the only graph where no pair of spanning trees has an intermediate tree.

Theorem 1. *Let G be a simple connected graph with at least 3 vertices. No pair of spanning trees of G has an intermediate tree if and only if G consists of K_3 together with a tree rooted at each vertex of the K_3 .*

We are interested in characterizing the graphs in which every pair of spanning trees has an intermediate tree.

Theorem 2. *In the following classes of graphs, every pair of spanning trees has an intermediate tree:*

- Complete graphs K_n where $n \geq 5$
- Complete bipartite graphs $K_{m,n}$ where $m, n \geq 4$

Note that each of K_4 and $K_{3,4}$ contains a pair of spanning trees that have no intermediate tree.

In most cases, we can find an intermediate tree of a given pair of spanning trees in graph G by adding at most one edge to G .

References

- [1] Kenneth A. Berman, Parity results on connected f -factors, *Discrete Math.* **59** (1986), 1-8.
- [2] Kathie Cameron and Jack Edmonds, Some graphic uses of an even number of odd nodes, *Ann. Inst. Fourier* **49** (1999), 815-827.
- [3] Douglas B. West, *Introduction to Graph Theory*, Prentice Hall, Upper Saddle River, 2001.

Plenary Session I

Lecture Hall A

Tue 2, 16:45-17:30

Bilevel Programming and Maximally Violated Valid Inequalities

Andrea Lodi,^a Ted K. Ralphs^b

^a*DEIS, University of Bologna
Viale Risorgimento 2, 40136, Bologna
andrea.lodi@unibo.it*

^b*Department of Industrial and Systems Engineering
Lehigh University, Bethlehem, PA 18015
ted@lehigh.edu*

Key words: Bilevel Programming, Cutting Planes, Separation

1. Introduction

In recent years, *branch-and-cut* algorithms have become firmly established as the most effective method for solving generic mixed integer linear programs (MIPs). Methods for automatically generating inequalities valid for the convex hull of solutions to such MIPs are a critical element of branch-and-cut. This paper examines the nature of the so-called *separation problem*, which is that of generating a valid inequality violated by a given real vector, usually arising as the solution to a relaxation of the original problem. We show that the problem of generating a maximally violated valid inequality often has a natural interpretation as a *bilevel program*. In some cases, this bilevel program can be easily reformulated as a single-level mathematical program, yielding a standard mathematical programming formulation for the separation problem. In other cases, no reformulation exists. We illustrate the principle by considering the separation problem for two well-known classes of valid inequalities.

Formally, we consider a MIP of the form

$$\min\{c^\top x \mid Ax \geq b, x \geq 0, x \in \mathbb{Z}^I \times \mathbb{R}^C\}, \quad (1.1)$$

where $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$, I is the set of indices of components that must take integer values in any feasible solution and C consists of the indices of the remaining components. We assume that other bound constraints on the variables (if any) are included among the problem constraints.

The *continuous* or *linear programming* (LP) relaxation of the above MIP is the mathematical program obtained by dropping the integrality requirement on the variables in I , namely

$$\min_{x \in \mathcal{P}} c^\top x, \quad (1.2)$$

where $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \geq b, x \geq 0\}$ is the polyhedron described by the linear constraints of the MIP (1.1). It is not difficult to see that the convex hull of the set of feasible solutions to (1.1) is also a polyhedron. This means that in principle, any MIP is equivalent to a linear program over this implicitly defined polyhedron, which we denote as \mathcal{P}_I .

A bilevel mixed integer linear program (BMIP) is a generalization of a standard MIP used to model hierarchical decision processes. In a BMIP, the variables are split into a set of *upper-level variables*, denoted by x below, and a set of *lower-level variables*, denoted by y below. Conceptually, the values of the upper-level variables are fixed first, subject to the restrictions of a set of *upper-level constraints*, after which the second-stage variables are fixed by solving a MIP parameterized on the fixed values of the upper-level variables. The canonical integer bilevel MIP is given by

$$\min \left\{ c^1 x + d^1 y \mid x \in \mathcal{P}_U \cap (\mathbb{Z}^{I_1} \times \mathbb{R}^{C_1}), \right. \\ \left. y \in \operatorname{argmin} \{ d^2 y \mid y \in \mathcal{P}_L(x) \cap (\mathbb{Z}^{I_2} \times \mathbb{R}^{C_2}) \} \right\},$$

where

$$\mathcal{P}_U = \{x \in \mathbb{R}^{n_1} \mid A^1 x \geq b^1, x \geq 0\}$$

is the polyhedron defining the *upper-level feasible region*;

$$\mathcal{P}_L(x) = \{y \in \mathbb{R}^{n_2} \mid G^2 y \geq b^2 - A^2 x, y \geq 0\}$$

is the polyhedron defining the *lower-level feasible region* with respect to a given $x \in \mathbb{R}^{n_1}$; $A^1 \in \mathbb{Q}^{m_1 \times n_1}$; $b^1 \in \mathbb{Q}^{m_1}$; $A^2 \in \mathbb{Q}^{m_2 \times n_1}$, $G^2 \in \mathbb{Q}^{m_2 \times n_2}$; and $b^2 \in \mathbb{Q}^{m_2}$. The index sets I_1 , I_2 , C_1 , and C_2 are the bilevel counterparts of the sets I and C defined previously. For more detailed information, [5] provide an introduction to and comprehensive survey of the bilevel programming literature, while [14] introduce the discrete case. [9] provides a detailed bibliography.

A *valid inequality* for a set $\mathcal{S} \subseteq \mathbb{R}^n$ is a pair (α, β) , where $\alpha \in \mathbb{R}^n$ is the *coefficient vector* and $\beta \in \mathbb{R}$ is the *right-hand side*, such that $\alpha^\top x \geq \beta$ for all $x \in \mathcal{S}$. Associated with any valid inequality (α, β) is the half-space $\{x \in \mathbb{R}^n \mid \alpha x \geq \beta\}$, which must contain \mathcal{S} . It is easy to see that any inequality valid for \mathcal{S} is also valid for the convex hull of \mathcal{S} .

For a polyhedron $\mathcal{Q} \subseteq \mathbb{R}^n$, the so-called *separation problem* is to generate a valid inequality violated by a given vector. Formally, we define the problem as follows.

Definition 1. The *separation problem* for a polyhedron \mathcal{Q} is to determine for a given $\hat{x} \in \mathbb{R}^n$ whether or not $\hat{x} \in \mathcal{Q}$ and if not, to produce an inequality $(\bar{\alpha}, \bar{\beta}) \in \mathbb{R}^{n+1}$ valid for \mathcal{Q} and for which $\bar{\alpha}^\top \hat{x} < \bar{\beta}$.

A closely associated problem that is more relevant in practice is the *maximally violated valid inequality problem* (MVVIP), which is as follows.

Definition 2. The *maximally violated valid inequality problem* for a polyhedron \mathcal{Q} is to determine for a given $\hat{x} \in \mathbb{R}^n$ whether or not $\hat{x} \in \mathcal{Q}$ and if not, produce an inequality $(\bar{\alpha}, \bar{\beta}) \in \mathbb{R}^{n+1}$ valid for \mathcal{Q} and for which $(\bar{\alpha}, \bar{\beta}) \in \operatorname{argmin}_{(\alpha, \beta) \in \mathbb{R}^{n+1}} \{\alpha^\top \hat{x} - \beta \mid \alpha^\top x \geq \beta \forall x \in \mathcal{Q}\}$.

It is well-known that both the separation problem and the MVVIP for a polyhedron \mathcal{Q} are polynomially equivalent to the associated optimization problem, which is to determine $\min_{x \in \mathcal{Q}} d^\top x$ [12] for a given $d \in \mathbb{R}^n$. In the present context, this means it is unlikely that the MVVIP for \mathcal{P}_I can be solved easily unless the MIP itself can be solved easily.

Because the general MVVIP is usually too difficult to solve, valid inequalities are generated by solving (either exactly or approximately) the MVVIP for one or more relaxations of the original problem. These relaxations often come from considering valid inequalities in a specific *family* or *class*, i.e., inequalities that share a special structure. [1] called this paradigm for generation of valid inequalities the *template paradigm*. Generally speaking, a class of valid inequalities for a given set \mathcal{S} is simply a subset of all valid inequalities for \mathcal{S} . Such subsets can be defined in a number of ways and may be either finite or infinite. Associated with any given class \mathcal{C} is its closure $\mathcal{F}_{\mathcal{C}}$, consisting of the region defined by the intersection of all half-spaces associated with inequalities in the class. If the class is finite, then the closure is a polyhedron. Otherwise, it may or may not be a polyhedron.

Let us consider a given class of valid inequalities \mathcal{C} . Assuming the closure $\mathcal{F}_{\mathcal{C}}$ is a polyhedron, both the separation problem and the MVVIP for \mathcal{C} can be identified with the previously defined separation problem and MVVIP for $\mathcal{F}_{\mathcal{C}}$. A number of authors have noted that the MVVIP for certain classes of valid inequalities can be formulated as structured mathematical programs in their own right and solved using standard optimization techniques (see, e.g., [2], [4] and [10]). We wish to show that the underlying structure of the MVVIP is inherently *bilevel*.

The bilevel nature of the MVVIP for a class \mathcal{C} arises from the fact that for a given coefficient vector $\alpha \in \mathbb{R}^n$, the calculation of the right-hand side β required to ensure (α, β) is a member of the class (if such a β exists) may itself be an optimization problem that we refer to as the *right-hand side generation problem* (RHSGP). The complexity of the separation problem depends strongly on the complexity of the RHSGP. In cases where the RHSGP is in the complexity class NP-hard, it is generally not possible to formulate the separation problem as a tradi-

tional mathematical program. In fact, such separation problems may not even be in the complexity class NP . Roughly speaking, the reason for this is that the problem of determining whether a given inequality is valid is then itself a hard problem.

Putting these question aside for now, however, let us simply define the set $\mathcal{C}_\alpha \subseteq \mathbb{R}^n$ to be the projection of \mathcal{C} into the space of coefficient vectors. In other words, \mathcal{C}_α is the set of all vectors that are coefficients for some valid inequality in \mathcal{C} . Then the MVVIP for \mathcal{C} with respect to a given $\hat{x} \in \mathbb{R}^n$ can in principle be formulated mathematically as

$$\min \alpha^\top \hat{x} - \beta \tag{1.3}$$

$$\alpha \in \mathcal{C}_\alpha \tag{1.4}$$

$$\beta = \min \alpha^\top x \tag{1.5}$$

$$x \in \mathcal{F}_\mathcal{C}. \tag{1.6}$$

The problem (1.3)–(1.6) is a bilevel program in which the *upper-level objective* (1.3) is to find the maximally violated inequality in the class. The upper-level constraints (1.4) require that the inequality is a member of the class. The lower-level problem (1.5)–(1.6) is to generate the strongest possible right-hand side associated with a given coefficient vector.

It is easy to see that the above separation problem may be very difficult to solve in some cases. In fact, the complexity depends strongly on the complexity of the RHSGP and whether the sense of the optimization “agrees” with that of the MVVIP itself. Most of the separation algorithms appearing in the literature define the set $\mathcal{F}_\mathcal{C}$ in such a way that the bilevel program (1.3)–(1.6) collapses into a single-level program, generally linear or mixed integer linear.

In the remainder of the paper, we describe two well-known classes of valid inequalities and give a bilevel interpretation of their associated separation problems. In Section 2, we consider the well-known class of *disjunctive valid inequalities* for general MIPs. For such a class, we show that it is quite straightforward to convert the BMIP (1.3)–(1.6) into a single-level mathematical program, though the MVVIP might nevertheless remain difficult from a practical standpoint. In Section 3, we focus on the so-called *capacity constraints* for the classical *Capacitated Vehicle Routing Problem* (CVRP). There are several closely-related variants of this class of valid inequalities and we show that for the strongest of these, there is no straightforward way to convert the BMIP into a single-level program. That is the main contribution of the present paper, and to the best of our knowledge, it is a new result. Finally, some conclusions are drawn in Section 4.

2. Disjunctive Valid Inequalities for general MIPs

Given a MIP in the form (1.1), [2] showed how to derive a valid inequality by exploiting any disjunction of the form

$$\pi^\top x \leq \pi_0 \quad \text{OR} \quad \pi^\top x \geq \pi_0 + 1 \quad \forall x \in \mathbb{R}^n, \quad (2.7)$$

where $\pi \in \mathbb{Z}^I \times \mathbf{0}^C$ and $\pi_0 \in \mathbb{Z}$. More precisely, the family of disjunctive inequalities (also called *split cuts*) are all those valid for the union of the two polyhedra, denoted by \mathcal{P}_1 and \mathcal{P}_2 , obtained from \mathcal{P} by adding inequalities $(-\pi, -\pi_0)$ and $(\pi, \pi_0 + 1)$, respectively.

For a given disjunction of the form (2.7), the separation problem for the associated family of disjunctive inequalities with respect to a given vector $\hat{x} \in \mathcal{P}$ can be written as a the following bilevel LP:

$$\min \quad \alpha^\top \hat{x} - \beta \quad (2.8)$$

$$\alpha_j \geq u^\top A_j - u_0 \pi_j \quad j \in I \cup U \quad (2.9)$$

$$\alpha_j \geq v^\top A_j + v_0 \pi_j \quad j \in I \cup U \quad (2.10)$$

$$u, v, u_0, v_0 \geq 0 \quad (2.11)$$

$$u_0 + v_0 = 1 \quad (2.12)$$

$$\beta = \min \alpha^\top x \quad (2.13)$$

$$x \in \mathcal{P}_1 \cup \mathcal{P}_2. \quad (2.14)$$

Constraints (2.9) and (2.10) together with the non-negativity requirements on the dual multipliers (2.11) ensure the coefficients constitute those of a disjunctive inequality. (Constraint (2.12) is one of the possible normalizations to make the mathematical program above bounded, see, e.g., [11].) Once the coefficient vector and the corresponding dual multipliers are known, the RHSGP is easy to solve. To obtain a valid inequality, one has only to set β to $\min\{u^\top b - u_0 \pi_0, v^\top b + v_0(\pi_0 + 1)\}$, which is the smallest of the right-hand sides obtained by the sets of multipliers (u, u_0) and (v, v_0) corresponding to the constraints of \mathcal{P}_1 and \mathcal{P}_2 , respectively. It is easy to reformulate the bilevel LP above into the following (single level) linear program by a well-known modeling trick:

$$\min \quad \alpha^\top \hat{x} - \beta \quad (2.15)$$

$$\alpha_j \geq u^\top A_j - u_0 \pi_j \quad j \in I \cup U$$

$$\alpha_j \geq v^\top A_j + v_0 \pi_j \quad j \in I \cup U$$

$$\beta \leq u^\top b - u_0 \pi_0 \quad (2.16)$$

$$\beta \leq v^\top b + v_0(\pi_0 + 1) \quad (2.17)$$

$$u_0 + v_0 = 1$$

$$u, v, u_0, v_0 \geq 0.$$

Indeed, note that for given values of the remaining variables, any value of β satisfying the two inequalities (2.16) and (2.17) above yields a valid disjunctive constraint. Furthermore, these two inequalities ensure that $\beta \leq \min\{u^\top b - u_0\pi_0, v^\top b + v_0(\pi_0 + 1)\}$, while the objective function (2.15) ensures that the largest possible value of β is indeed selected, i.e., $\beta = \min\{u^\top b - u_0\pi_0, v^\top b + v_0(\pi_0 + 1)\}$. In other words, the objective function (2.15) gives for free the best value of the right-hand side, thus finding the strongest cut.

If the disjunction is not given a priori, i.e., one is searching among the set of possible disjunctions for the one yielding the most violated constraint, the above program can still be used, but π and π_0 become integer variables. The same trick can be applied to transform the bilevel separation problem into a single-level one, but the problem remains difficult because (i) some of the constraints contain bilinear terms, and (ii) the program involves the integer variables π and π_0 . The solution of such a formulation has been addressed by [3] and [8].

3. Capacity Constraints for the CVRP

Here, we consider the classical *Capacitated Vehicle Routing Problem (CVRP)*, as introduced by [7], in which a quantity d_i of a single commodity is to be delivered to each customer $i \in N = \{1, \dots, n\}$ from a central depot $\{0\}$ using a homogeneous fleet of k vehicles, each with capacity K . The objective is to minimize total cost, with $c_{ij} \geq 0$ denoting the fixed cost of transportation from location i to location j , for $0 \leq i, j \leq n$. The costs are assumed to be *symmetric*, i.e., $c_{ij} = c_{ji}$ and $c_{ii} = 0$.

This problem is naturally associated with the complete undirected graph consisting of nodes $N \cup \{0\}$, edge set $E = N \times N$, and edge costs c_{ij} , $\{i, j\} \in E$. In this graph, a solution is the union of k cycles whose only intersection is the depot node and whose union covers all customers. By associating an integer variable with each edge in the graph, we obtain the following integer programming formulation:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ & \sum_{e = \{0, j\} \in E} x_e = 2k \end{aligned} \tag{3.18}$$

$$\sum_{e = \{i, j\} \in E} x_e = 2 \quad \forall i \in N \tag{3.19}$$

$$\sum_{\substack{e = \{i, j\} \in E \\ i \in S, j \notin S}} x_e \geq 2b(S) \quad \forall S \subset N, |S| > 1 \tag{3.20}$$

$$0 \leq x_e \leq 1 \quad \forall e = \{i, j\} \in E, i, j \neq 0 \tag{3.21}$$

$$0 \leq x_e \leq 2 \quad \forall e = \{0, j\} \in E \tag{3.22}$$

$$x_e \quad \text{integral} \quad \forall e \in E. \tag{3.23}$$

Constraints (3.18) and (3.19) are the *degree constraints*. In constraints (3.20), referred to as the *capacity constraints*, $b(S)$ is any of several lower bounds on the number of trucks required to service the customers in set S . These constraints can be viewed as a generalization of the subtour elimination constraints from the *Traveling Salesman Problem* and serve both to enforce the connectivity of the solution and to ensure that no route has total demand exceeding the capacity K . The easily calculated lower bound $\sum_{i \in S} d_i / K$ on the number of trucks is enough to ensure the formulation (3.18)–(3.23) is correct, but increasing this bound through the solution of a more sophisticated RHSGP will yield a stronger version of the constraints.

The MVVIP for capacity constraints with a generic lower bound $b(S)$ can be formulated as a BMIP of the form (1.3)–(1.6) as follows. Because we are looking for a set $\bar{S} \subset N$ for which an inequality (3.20) is maximally violated, we define the binary variables

$$y_i = \begin{cases} 1 & \text{if customer } i \text{ belong to } \bar{S} \\ 0 & \text{otherwise} \end{cases} \quad i \in N, \quad (3.24)$$

and

$$z_e = \begin{cases} 1 & \text{if edge } e \text{ belong to } \delta(\bar{S}) \\ 0 & \text{otherwise} \end{cases} \quad e \in E, \quad (3.25)$$

where $\delta(\bar{S})$ denotes the set of edges in E with one endpoint in \bar{S} , to model selection of the members of the set \bar{S} and the coefficients of the corresponding inequality. Thus, the formulation is

$$\min \sum_{e \in E} \hat{x}_e z_e - 2b(\bar{S}) \quad (3.26)$$

$$z_e \geq y_i - y_j \quad \forall e = \{i, j\} \quad (3.27)$$

$$z_e \geq y_j - y_i \quad \forall e = \{i, j\} \quad (3.28)$$

$$\max b(\bar{S}) \quad (3.29)$$

$$b(\bar{S}) \text{ is a valid lower bound.} \quad (3.30)$$

For improved tractability, the RHSGP (3.29)–(3.30) can be replaced by the calculation of a specific bound. One of the strongest possible lower bounds is obtained by solving to optimality the (strongly *NP-hard*) *Bin Packing Problem* (BPP) with the customer demands in set \bar{S} being packed into the minimum number of bins of size K ([6] describe a further strengthening of the right-hand side, but we do not consider this bound here). The RHSGP based on the BPP can be modeled by

using the binary variables

$$w_i^\ell = \begin{cases} 1 & \text{if customer } i \text{ is served by vehicle } \ell \\ 0 & \text{otherwise} \end{cases} \quad (i \in N, \ell = 1, \dots, k), \quad (3.31)$$

and

$$h_\ell = \begin{cases} 1 & \text{if vehicle } \ell \text{ is used} \\ 0 & \text{otherwise} \end{cases} \quad (\ell = 1, \dots, k). \quad (3.32)$$

Then the full separation problem reads as follows:

$$\min \sum_{e \in E} \hat{x}_e z_e - 2b(\bar{S}) \quad (3.33)$$

$$z_e \geq y_i - y_j \quad \forall e = \{i, j\} \quad (3.34)$$

$$z_e \geq y_j - y_i \quad \forall e = \{i, j\} \quad (3.35)$$

$$b(\bar{S}) = \min \sum_{\ell=1}^n h_\ell \quad (3.36)$$

$$\sum_{\ell=1}^n w_i^\ell = y_i \quad \forall i \in N \quad (3.37)$$

$$\sum_{i \in N} d_i w_i^\ell \leq K h_\ell \quad \ell = 1, \dots, n, \quad (3.38)$$

where of course all variables y , z , w and h are binary.

It is clear that the BMIP (3.33)–(3.38) cannot be straightforwardly reduced to a single-level program because the sense of the optimization of the RHSGP is opposed to that of the MVVIP. In other words, because of the upper-level objective function (3.33), the absence of the lower-level objective would result in a BPP solution using the largest number of bins instead of the smallest.

We can simplify the RHSGP by relaxing the integrality requirement on w and h to obtain

$$b(\bar{S}) = \min \sum_{\ell=1}^n h_\ell \quad (3.39)$$

$$\sum_{\ell=1}^n w_i^\ell = y_i \quad \forall i \in N \quad (3.40)$$

$$\sum_{i \in N} d_i w_i^\ell \leq K h_\ell \quad \ell = 1, \dots, n \quad (3.41)$$

$$w_i^\ell \in [0, 1], \quad h_\ell \in [0, 1] \quad i \in N, \ell = 1, \dots, n, \quad (3.42)$$

which is also a lower bound for the BPP. In this case, the RHSGP can be solved in

closed form, with an optimal solution being

$$b(\bar{S}) = \frac{\sum_{i \in \bar{S}} d_i}{K} = \frac{\sum_{i \in N} d_i y_i}{K}. \quad (3.43)$$

Hence, the MVVIP reduces to a single-level MIP that can in turn be solved in polynomial time by transforming it into a network flow problem as proven by [13].

An intermediate valid lower bound is obtained by rounding the bound (3.43), i.e., using $b(S) = \left\lceil \frac{\sum_{i \in N} d_i y_i}{K} \right\rceil$. Although such rounding can be done after the fact, relaxing the integrality on w , but not h , i.e., replacing conditions (3.42) by

$$w_i^\ell \in [0, 1], \quad h_\ell \in \{0, 1\} \quad i \in N, \ell = 1, \dots, n,$$

results in reduction of the MVVIP to the single-level MIP

$$\begin{aligned} \min \quad & \sum_{e \in E} \hat{x}_e z_e - 2b \\ & z_e \geq y_i - y_j & \forall e = \{i, j\} \\ & z_e \geq y_j - y_i & \forall e = \{i, j\} \\ & b \geq \frac{\sum_{i \in N} d_i y_i}{K} \\ & b \text{ integral} \\ & y_i \in \{0, 1\}, z_e \in \{0, 1\} & \forall i \in N, \forall e \in E, \end{aligned}$$

which was shown by [6] to be *NP*-hard.

4. Conclusions

We have presented a conceptual framework for the formulation of general separation problems as bilevel programs. This framework reflects the inherent bilevel nature of the separation problem arising from the fact that calculation of a valid right-hand side for a given coefficient vector is itself an optimization problem. In cases where this optimization problem is difficult in a complexity sense, it is generally not possible to formulate the separation problem as a traditional mathematical program. We conjecture that the MVVIP for most classes of valid inequalities can be thought of as having this hierarchical structure, but that certain of them can nonetheless be reformulated effectively. This is either because the RHSGP is easy to solve or because it goes “in the right direction” with respect to the MVVIP itself. We believe that the paradigm presented here may be useful for the analysis of other intractable classes of valid inequalities. In a future study, we plan to further formalize the conceptual framework presented here with a further investigation of the complexity issues, additional examples of this phenomena, and an assessment whether these ideas may be useful from a computational perspective.

Acknowledgements

We warmly thank Leo Liberti for useful comments about the paper.

References

- [1] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2007.
- [2] E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.
- [3] E. Balas and A. Saxena. Optimizing over the split closure. *Mathematical Programming*, 113:219–240, 2008.
- [4] A. Caprara and A. Letchford. On the separation of split cuts and related inequalities. *Mathematical Programming*, 94:279–294, 2003.
- [5] B. Colson, P. Marcotte, and G. Savard. Bilevel programming: A survey. *4OR: A Quarterly Journal of Operations Research*, 3:87–107, 2005.
- [6] G. Cornuéjols and F. Harche. Polyhedral Study of the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 60:21–52, 1993.
- [7] G. B. Dantzig and R. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6:80–91, 1959.
- [8] S. Dash, O. Günlük, and A. Lodi. On the MIR closure of polyhedra. In M. Fischetti and D. P. Williamson, editors, *Integer Programming and Combinatorial Optimization - IPCO 2007*, volume 4513 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, 2007.
- [9] S. Dempe. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints. *Optimization*, 52:333–359, 2003.
- [10] M. Fischetti and A. Lodi. Optimizing over the first Chvátal closure. *Mathematical Programming*, 110:3–20, 2007.
- [11] M. Fischetti, A. Lodi, and A. Tramontani. On the separation of disjunctive cuts. Technical Report OR-08-2, DEIS, University of Bologna, 2008.
- [12] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.
- [13] S. T. McCormick, M. R. Rao, and G. Rinaldi. Easy and difficult objective functions for max cut. *Mathematical Programming*, 94:459–466, 2003.
- [14] J. Moore and J. Bard. The mixed integer linear bilevel programming problem. *Operations Research*, 38:911–921, 1990.

Integer Programming

Lecture Hall A

Wed 3, 08:45–10:15

Decomposition Methods for Stochastic Integer Programs with Dominance Constraints

Rüdiger Schultz

*Department of Mathematics, University of Duisburg-Essen
Lotharstr. 65, D-47048 Duisburg, Germany
schultz@math.uni-duisburg.de*

Key words: stochastic integer programming, stochastic dominance, decomposition methods

Stochastic programming models are derived from random optimization problems with information constraints. For instance, we may start out from the random mixed-integer linear program

$$\min\{c^\top x + q^\top y + q'^\top y' : Tx + Wy + W'y' = z(\omega), \\ x \in X, y \in \mathbb{Z}_+^{\bar{m}}, y' \in \mathbb{R}_+^{m'}\}, \quad (0.1)$$

together with the information constraint that x must be selected without anticipation of $z(\omega)$. This leads to a two-stage scheme of alternating decision and observation: The decision on x is followed by observing $z(\omega)$ and then (y, y') is taken, thus depending on x and $z(\omega)$. Accordingly, x and (y, y') are called first- and second-stage decisions, respectively.

Assume that the ingredients of (0.1) have conformable dimensions, that W, W' are rational matrices, and that $X \subseteq \mathbb{R}^m$ is a nonempty polyhedron, possibly involving integer requirements to components of x .

The mentioned two-stage dynamics becomes explicit by the following reformulation of (0.1)

$$\min_x \{c^\top x + \Phi(z(\omega) - Tx) : x \in X\}$$

where

$$\Phi(t) := \min\{q^\top y + q'^\top y' : Wy + W'y' = t, y \in \mathbb{Z}_+^{\bar{m}}, y' \in \mathbb{R}_+^{m'}\}. \quad (0.2)$$

In this way, the random optimization problem (0.1) gives rise to the family of random variables

$$\left(c^\top x + \Phi(z(\omega) - Tx) \right)_{x \in X}. \quad (0.3)$$

Two principal alternatives arise at this point. Either the aim is to find “a best” element in this family of random variables or the aim is to single out “acceptable” elements and optimize some objective function over the feasible set arising.

Traditional stochastic programming, see for instance [4], followed the first alternative by judging the quality of the random variables in (0.3) according to their expectations

$$Q_{\mathbb{E}}(x) := \mathbb{E}_{\omega} [c^\top x + \Phi(z(\omega) - Tx)].$$

leading to the optimization problem

$$\min\{Q_{\mathbb{E}}(x) : x \in X\}.$$

This model, however, is risk neutral. Introducing risk aversion into the first alternative of handling (0.3) leads to mean-risk models

$$\min\{Q_{MR}(x) : x \in X\}$$

where

$$Q_{MR}(x) := (\mathbb{E} + \rho \cdot \mathcal{R})[c^\top x + \Phi(z - Tx)] = Q_{\mathbb{E}}(x) + \rho \cdot Q_{\mathcal{R}}(x)$$

with some fixed $\rho > 0$. Here \mathcal{R} denotes a statistical parameter reflecting risk (risk measure). For a possible specification see for instance [5].

Partial orders of random variables provide a possibility to formalize the second alternative of handling (0.3). A (real-valued) random variable X is said to be stochastically smaller in first order than a random variable Y ($X \preceq_1 Y$) iff $\mathbb{E}h(X) \leq \mathbb{E}h(Y)$ for all nondecreasing functions h for which both expectations exist. X is said to be stochastically smaller than Y in increasing convex order ($X \preceq_{icx} Y$) iff $\mathbb{E}h(X) \leq \mathbb{E}h(Y)$ for all nondecreasing convex functions h for which both expectations exist.

Equivalent formulations read as follows (see, e.g., [3]):

$$X \preceq_1 Y \quad \text{iff} \quad \mathbb{P}[\{\omega : X(\omega) \leq \eta\}] \geq \mathbb{P}[\{\omega : Y(\omega) \leq \eta\}] \quad \forall \eta \in \mathbb{R} \quad (0.4)$$

and

$$X \preceq_{icx} Y \quad \text{iff} \quad \mathbb{E}_{\omega}[X(\omega) - \eta]_+ \leq \mathbb{E}_{\omega}[Y(\omega) - \eta]_+ \quad \forall \eta \in \mathbb{R} \quad (0.5)$$

with $[\cdot]_+$ denoting the non-negative part.

With a prescribed reference random variable $d(\omega)$, “acceptance” of a random variable $f(x, \omega) := c^\top x + \Phi(z(\omega) - Tx)$ can be formalized as

$$f(x, \omega) \preceq_\iota d(\omega), \quad \text{with either } \iota = 1 \text{ or } \iota = icx.$$

This means that only those $x \in X$ are acceptable whose associated cost profile $f(x, \omega)$, in a stochastic sense, is not worse than the prescribed (critical) random profile $d(\omega)$. With an objective function $g : \mathbb{R}^m \rightarrow \mathbb{R}$ this leads to the following stochastic optimization problem with dominance constraints

$$\min\{g(x) : f(x, \omega) \preceq_\iota d(\omega), x \in X\}, \quad \iota = 1, icx. \quad (0.6)$$

Stochastic optimization problems with dominance constraints involving general random variables were pioneered in [1; 2], with first results on structure, stability, and algorithms for (0.6) if general random variables replace $f(x, \omega)$. The random variables $f(x, \omega)$ are more specific, since essentially given by the mixed-integer value function in (0.2). Nevertheless, the results from [1; 2] are not applicable here since Φ in (0.2) fails to be smooth, let alone linear, and often even turns out discontinuous.

The talk will address the following:

- (departing from (0.4) and (0.5)) equivalent mixed-integer linear programming formulations for (0.6) if the probability distributions of z and d are discrete with finitely many realizations,
- branch-and-bound based decomposition algorithms for solving these mixed-integer linear programs,
- cutting plane based decomposition algorithms if there are no integer variables in the second stage.

References

- [1] Dentcheva, D.; Ruszczyński, A.: Optimization with stochastic dominance constraints, *SIAM Journal on Optimization* 14 (2003), 548 - 566.
- [2] Dentcheva, D.; Ruszczyński, A.: Optimality and duality theory for stochastic optimization with nonlinear dominance constraints, *Mathematical Programming* 99 (2004), 329 - 350.
- [3] Müller, A.; Stoyan, D.: *Comparison Methods for Stochastic Models and Risks*, Wiley, Chichester, 2002.
- [4] Ruszczyński, A.; Shapiro, A. (eds.): *Handbooks in Operations Research and Management Science*, 10: *Stochastic Programming*, Elsevier, 2003.
- [5] Schultz, R., Tiedemann, S.: Conditional value-at-risk in stochastic programs with mixed-integer recourse, *Math. Progr.* 105 (2006), 365–386.

On a Two-Stage Stochastic Knapsack Problem with Probabilistic Constraint

Stefanie Kosuch and Abdel Lisser

LRI, Université Paris XI, Orsay, France

Key words: stochastic programming, two-stage knapsack problem, probabilistic constraint

1. Introduction

The knapsack problem is a widely studied combinatorial optimization problem. Special interest arises from the numerous real life applications for example in logistics and scheduling. The basic problem consists in choosing a subset out of a given set of items such that the total weight (or size) of the subset does not exceed a given limit (the capacity of the knapsack) and the total benefit of the subset is maximized.

However, most real life problems are non-deterministic in the sense that some of the parameters are not known in the moment when the decision has to be made. We will study the case where the item weights are random. This case entail the problem that we cannot be sure that the total weight of the items chosen in advance (i.e. before the revealing of the actual weights) will respect the capacity. Depending on the situation given, the resulting stochastic problem can be modeled in two different ways: Either using a single stage problem which means that the final decision has to be made before the random parameters are revealed ([3], [2]); or by a two- or multi-stage problem that allows later corrections of the decision made in the first stage ([2]).

In this paper, we discuss a two-stage stochastic knapsack problem (see section 2) and we assume the item weights to be independently distributed following a (known) normal distribution. The first aim of this paper is to show how to obtain upper bounds on the overall problem or on subproblems (i.e. with some of the first stage variables already fixed). The second aim is to compute high probable lower bounds on the overall problem, given a first stage decision. These upper and lower bounds could afterwards be used in a branch-and-bound framework such as presented in [1] or [3] in order to search the solution space of the first stage variables for good lower bounds on the overall problem.

2. Mathematical formulation

We consider a stochastic knapsack problem of the following form: Given a knapsack with fixed weight capacity $c > 0$ as well as a set of n items. Each item has a weight that is not known in the first stage and that comes to be known before the second stage decision has to be made. We handle the weights as random variables and assume that weight χ_i of item i is independently normally distributed with mean $\mu_i > 0$ and standard deviation σ_i . Furthermore, each item has a fix reward per weight unit $r_i > 0$.

In the first stage, items can be put in the knapsack (first stage items) and we assume that, in case of an overload, these items can be removed in the second stage. However, removing items entails a penalty that is proportional to the total weight of the removed items. We restrict the percentage of cases where the first stage items lead to an overload by introducing a probabilistic constraint in the first stage. In the second stage, the weights of all items are revealed and the aim is to minimize the total penalty.

$$(TSKP) \quad \max_{x \in \{0,1\}^n} \quad \mathbf{E} [\sum_{i=1}^n r_i \chi_i x_i] - d \cdot \mathbf{E} [\mathcal{Q}(x, \chi)]$$

$$\text{s.t.} \quad \mathbf{P}\{\sum_{i=1}^n \chi_i x_i \leq c\} \geq p \quad (2.1)$$

$$\mathcal{Q}(x, \chi) = \min_{y \in \{0,1\}^n} \sum_{i=1}^n \chi_i y_i, \quad (2.2)$$

$$\text{s.t.} \quad y_k \leq x_k, \quad k = 1, \dots, n. \quad (2.3)$$

$$\sum_{i=1}^n (x_i - y_i) \chi_i \leq c, \quad (2.4)$$

where $\mathcal{P}\{A\}$ denotes the probability of an event A , $\mathbf{E}[\cdot]$ the expectation, $d > 1$ and $p \in (0.5, 1]$.

3. Computing upper and lower bounds

3.0.0.1 Upper bounds: Given a first stage solution \tilde{x} , the expectation of the second stage solution can be bounded (from below) by

$$\mathbf{E}[\mathcal{Q}(\tilde{x}, \chi)] \geq \mathbf{E} \left[\left[\sum_{i=1}^n \tilde{x}_i \chi_i - c \right]^+ \right]$$

The right hand side of the inequality equals the expectation of the optimal solution of the second stage problem in case of continuous second stage variables. For normally distributed weights, it has a deterministic equivalent closed form ([1],[3]). An upper bound on the optimal solution of the *TSKP* (2.1) is thus given by the optimal solution of the following simple recourse problem:

$$(SRKP) \quad \max_{x \in \{0,1\}^n} \quad \mathbf{E} [\sum_{i=1}^n r_i \chi_i x_i] - d \cdot \mathbf{E} [[\sum x_i \chi_i - c]^+]$$

$$\text{s.t.} \quad \mathbf{P}\left\{\sum_{i=1}^n \chi_i x_i \leq c\right\} \geq p \quad (3.5)$$

$$(3.6)$$

This problem can be solved by the method presented in [3].

3.0.0.2 Lower bounds: If we are not able to solve the second stage problem to optimality (given a first stage decision), we need good lower bounds on the overall problem (i.e. good upper bounds on the second stage problem) to be able to exclude subtrees in a branch-and-bound framework.

Given a first stage solution \tilde{x} , the expectation of the optimal second stage solution $\mathbf{E}[Q(\tilde{x}, \chi)]$ can be written

$$\mathbf{E}[Q(\tilde{x}, \chi)] = \mathbf{E}\left[\sum_{i=1}^n \tilde{x}_i \chi_i - c\right]^+ + \mathbf{E}\left[\sum_{i \in S} y_i^* \chi_i - \left[\sum_{i=1}^n \tilde{x}_i \chi_i - c\right]^+\right]$$

where $y^* = y^*(\tilde{x})$ is a corresponding optimal second stage decision and $S \subseteq \{1, \dots, n\}$ such that $i \in S$ if and only if $\tilde{x}_i = 1$. $\sum_{i \in S} y_i^* \chi_i - [\sum_{i=1}^n \tilde{x}_i \chi_i - c]^+$ is the amount of weight we remove from the knapsack in addition to the overweight. This amount can be bounded independently of the second stage solution y^* as in the worst case the knapsack weight might fall $\max_{i \in S} \hat{\chi}_i - \epsilon$ under the capacity due to the removal of items in the second stage (where $\epsilon > 0$ and, for all $i = 1, \dots, n$, $\hat{\chi}_i$ is the actual outcome of random variable χ_i). As items have to be removed in at most $\frac{1-p}{100}$ percent of all cases, we get the following lemma:

Lemma 3.1.
$$\mathbf{E}\left[\sum_{i \in S} y_i^* \chi_i - \left[\sum_{i=1}^n \tilde{x}_i \chi_i - c\right]^+\right] < (1-p) \cdot \mathbf{E}[\max_{i \in S} \chi_i]$$

Lemma 3.2. If the probability for any item to have twice the size of another item is at most π , it follows

$$\mathbf{E}\left[\sum_{i \in S} y_i^* \chi_i - \left[\sum_{i=1}^n \tilde{x}_i \chi_i - c\right]^+\right] < (1-p) \left((1-\pi) \cdot \mathbf{E}\left[\min_{i \in S} \hat{\chi}_i\right] + \pi \cdot \mathbf{E}\left[\max_{i \in S} \chi_i\right] \right)$$

In the case of normally distributed weights, neither $\mathbf{E}[\max_{i \in S} \chi_i]$ nor $\mathbf{E}[\min_{i \in S} \chi_i]$ are easily computable. Let us define the random variables $\chi_{max}^S := \max_{i \in S} \chi_i$ and $\chi_{min}^S := \min_{i \in S} \chi_i$. Let Φ_i be the cumulative distribution function of χ_i , and Φ_{max}^S and Φ_{min}^S the cumulative distribution functions of χ_{max}^S and χ_{min}^S , respectively. Then one can easily show that $\Phi_{max}^S = \prod_{i \in S} \Phi_i$ and $\Phi_{min}^S = 1 - \prod_{i \in S} (1 - \Phi_i)$. Furthermore, if there exists a $\beta < \infty$ such that $\mathcal{P}\{\max_{i \in S} \chi_i \in (-\infty, \beta]\} = 1$ (resp. $\mathcal{P}\{\min_{i \in S} \chi_i \in (-\infty, \beta]\} = 1$), we can bound $\mathbf{E}[\max_{i \in S} \chi_i]$ (resp. $\mathbf{E}[\min_{i \in S} \chi_i]$) by splitting the interval $(-\infty, \beta]$ in K disjunct intervals (K scenarios) $(\alpha_k, \beta_k]$, $k = 1, \dots, K$, and it follows

$$\mathbf{E}[\max_{i \in S} \chi_i] \leq \sum_{k=1}^K \beta_k \mathcal{P}\{\max_{i \in S} \chi_i \in (\alpha_k, \beta_k]\} = \sum_{k=1}^K \beta_k (\Phi_{max}^S(\beta_k) - \Phi_{max}^S(\alpha_k))$$

$$\mathbf{E} [\min_{i \in S} \chi_i] \leq \sum_{k=1}^K \beta_k \mathcal{P}\{\min_{i \in S} \chi_i \in (\alpha_k, \beta_k]\} = \sum_{k=1}^K \beta_k (\Phi_{min}^S(\beta_k) - \Phi_{min}^S(\alpha_k))$$

Of course, in the case of normally distributed weights no such β exists. However, as for every ϵ there exists a β such that $\mathcal{P}\{\exists i \in S | \chi_i \geq \beta\} < \epsilon$, we can approximate the upper bound by defining β in such a way that $\mathcal{P}\{\exists i \in S | \chi_i \geq \beta\}$ is (sufficient) small.

Proposition 3.3. Let β such that $\mathcal{P}\{\exists i \in S | \chi_i > \beta\} = 0$ and define $(\alpha_k, \beta_k]$ ($k = 1, \dots, K$) such that $(-\infty, \beta] = \cup_{k=1}^K (\alpha_k, \beta_k]$. Let the probability for any item to have twice the size of another item be at most π . Then, given a first stage solution \tilde{x} , the following lower bound on the *TSKP* (2.1) hold:

$$\begin{aligned} \mathbf{E} \left[\sum_{i=1}^n r_i \chi_i \tilde{x}_i \right] - d \cdot \mathbf{E} [\mathcal{Q}(\tilde{x}, \chi)] &> \sum_{i=1}^n r_i \mu_i \tilde{x}_i - d \cdot \mathbf{E} \left[\left[\sum_{i=1}^n \tilde{x}_i \chi_i - c \right]^+ \right] \\ &+ (1-p) \left((1-\pi) \cdot \left(\sum_{k=1}^K \beta_k \left(\prod_{i \in S} (1 - \Phi_i(\alpha_k)) - \prod_{i \in S} (1 - \Phi_i(\beta_k)) \right) \right) \right. \\ &\quad \left. + \pi \cdot \beta_k \left(\prod_{i \in S} \Phi_i(\beta_k) - \prod_{i \in S} \Phi_i(\alpha_k) \right) \right) \end{aligned}$$

References

- [1] A. Cohn, C. Barnhart, *The Stochastic Knapsack Problem with Random Weights: A Heuristic Approach to Robust Transportation Planning*, Proceedings of the Triennial Symposium on Transportation Analysis (TRISTAN III), 1998.
- [2] A. Gaivoronski, A. Lisser, R. Lopez, *Knapsack problem with probability constraints*, Technical Report No. 1498 of the Laboratoire de recherche en informatique, Orsay, France, 2008.
- [3] S. Kosuch, A. Lisser, *Stochastic Knapsack Problems*, Technical Report No. 1505 of the Laboratoire de recherche en informatique, Orsay, France, 2008.

Improved strategies for branching on general disjunctions

Gerard Cornuéjols,^a Leo Liberti,^b Giacomo Nannicini^b

^a*LIF, Faculté de Sciences de Luminy, Marseille, France*
and Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA
gc0v@andrew.cmu.edu

^b*LIX, École Polytechnique, 91128 Palaiseau, France*
{liberti,giacomon}@lix.polytechnique.fr

Key words: Integer programming, branch and bound, split disjunctions

1. Extended abstract

In this paper we consider the Mixed Integer Linear Program in standard form:

$$\left. \begin{array}{l} \min c^\top x \\ Ax = b \\ x \geq 0 \\ \forall j \in N_I \quad x_j \in \mathbb{Z}, \end{array} \right\} \mathcal{P} \quad (1.1)$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ and $N_I \subset N = \{1, \dots, n\}$. The LP relaxation of (1.1) is the linear program obtained by dropping the integrality constraints, and is denoted by $\bar{\mathcal{P}}$. The Branch-and-Bound algorithm makes an implicit use of the concept of disjunctions [1]: whenever the solution of the current relaxation is fractional, we divide the current problem \mathcal{P} into two subproblems \mathcal{P}_1 and \mathcal{P}_2 such that the union of the feasible regions of \mathcal{P}_1 and \mathcal{P}_2 contains all feasible solutions to \mathcal{P} . Usually, this is done by choosing a fractional component \bar{x}_i (for some $i \in N_I$) of the optimal solution \bar{x} to the relaxation $\bar{\mathcal{P}}$, and adding the constraints $x_i \leq \lfloor \bar{x}_i \rfloor$ and $x_i \geq \lceil \bar{x}_i \rceil$ to \mathcal{P}_1 and \mathcal{P}_2 respectively.

Within this paper, we take the more general approach whereby branching can occur with respect to a direction $\pi \in \mathbb{R}^n$ by adding the constraints $\pi x \leq \beta_0$, $\pi x \geq \beta_1$ with $\beta_0 < \beta_1$ to \mathcal{P}_1 and \mathcal{P}_2 respectively, as long as no integer feasible point is cut off. Karamanov and Cornuéjols [2] proposed using disjunctions arising

from Gomory Mixed-Integer Cuts generated directly from the rows of the optimal tableau. We consider split disjunctions arising from Gomory Mixed-Integer Cuts generated from linear combinations of the rows of the simplex tableau; by computing combinations that yield a stronger intersection cut, we generate split disjunctions that cut deeply into the feasible region, thereby reducing the total number of nodes in the enumeration tree. By combining branching on simple disjunctions and on general disjunctions, we obtain an improvement over traditional branching rules on the majority of the test instances.

References

- [1] E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.
- [2] M. Karamanov and G. Cornuéjols. Branching on general disjunctions. Technical report, Carnegie Mellon University, 2005.

Graph Theory II

Lecture Hall B

Wed 3, 08:45–10:15

Extremal Stable Graphs

Gyula Y. Katona,^a Illés Horváth^b

^a*Department of Computer Science and Information Theory
Budapest University of Technology and Economics
1521, P. O. B.: 91, Hungary
kiskat@cs.bme.hu*

^b*Department of Stochastics
Budapest University of Technology and Economics
1521, P. O. B.: 91, Hungary
pollux@math.bme.hu*

Key words: extremal properties, stability, stable graphs

1. Introduction

There is a wide range of graph theoretical questions that is a special case of the following very general question: *Let Π be a graph property so that if $H \in \Pi$ and H is a subgraph of G then $G \in \Pi$. (i. e. being non- Π is a hereditary graph property.) What is the minimum number of edges in a graph $G \in \Pi$ on n vertices if removing any k edges or vertices from the graph still preserves Π ?*

A few examples:

- What is the minimum number of edges in a k -connected or k -edge connected graph?
- What is the minimum number of edges in hypo-hamiltonian graph?
- What is the minimum number of edges in graph that is still Hamiltonian after removing k edges (or vertices)? [2]

In the present paper we will concentrate on the problem where Π is the property that G contains a given fixed subgraph H . We only consider simple, undirected graphs.

¹ Research partially supported by the Hungarian National Research Fund and by the National Office for Research and Technology (Grant Number OTKA 67651)

Definition 1. (Stability) Let H be a fixed graph. If the graph G has the property that removing any k edges of G , the resulting graph still contains (not necessarily spans!) a subgraph isomorphic with H , then we say that G is k -stable.

By $S(n, k)$ we denote the minimum number of edges in a k -stable graph on n vertices, and by $S(k)$ the minimum number of edges in any k -stable graph (that is, $S(k) = \min_n S(n, k)$).

For clarity, we do not include H in the notation, instead just keep in mind that a graph H is always fixed. We regard calculating the value of $S(k)$ a separate question for each H .

Note that if n is fixed, then $S(n, k)$ is decreasing in k , because if there is a k -stable graph on n vertices, one can add isolated vertices to get k -stable graphs on $n + 1, n + 2, \dots$ vertices. This implies that for any fixed k , $S(n, k) = S(k)$ if n is large enough. In the present paper, we settle this question for several H graphs. Our main concern is $H = P_4$ (the path of 3 edges on 4 distinct vertices), but other graphs are of interest in their own right.

If $H = P_2$ (that is, a single edge containing 2 vertices), then naturally $S(k) = k + 1$ and any graph with $k + 1$ -edges is k -stable. We can state a general remark.

Remark 1. If H is fixed, then $S(k) \geq k + |V(H)|$.

Proposition 1.

- (a) If $H = P_3$, then $S(k) = k + |V(H)| = k + 3$.
- (b) If H is the graph on 4 vertices with two nonadjacent edges, then $S(k) = k + |V(H)| = k + 4$.

A general upper bound for the value of $S(k)$ holds too.

Proposition 2. For any fixed H , $S(k) \leq (|V(H)| + 3)k$ if k is large enough.

The main morale of these propositions is that for any choice of H , $S(k)$ is of a linear order. Of course, identifying the exact $S(k)$ functions is a completely different matter.

From now on, we fix $H = P_4$.

The question of P_4 was raised in [1] during the examination of Hamiltonian chains in hypergraphs. The authors calculated the value of $S(k)$ for $k \leq 8$ and posed a conjecture for larger k 's ([1], Conjecture 13). In the present paper we prove this conjecture.

2. Main result

Our main result is the following:

Theorem 1. $S(1) = 4$, and if $k \geq 2$, then $S(k) = k + \lceil \sqrt{2k + \frac{9}{4} + \frac{3}{2}} \rceil$.

Although it is written in an explicit form above, the following alternative definition may be easier to understand and just as useful.

Proposition 3. The above formula for $S(k)$ is equivalent with the following: $S(1) = 4$, $S(2) = 6$, and if $k \geq 3$, then

$$S(k) = \begin{cases} S(k-1) + 2 & \text{if } k = \binom{l}{2} \text{ for some } l \\ S(k-1) + 1 & \text{otherwise.} \end{cases}$$

Now let us take a look at the graphs containing no P_4 .

Proposition 4. If the graph G contains no P_4 as a subgraph, then all of its components are triangles and stars.

Proposition 5. If G has e edges on n vertices, then G is k -stable $\iff G$ cannot be covered by $k + n - e$ stars and any number of triangles.

In order to prove Theorem 1, we use the following theorem.

Theorem 2. Let G be a graph with $e \geq 5$ edges. If $e \geq \binom{l-1}{2} + 1$, then there are $l - 1$ edges of the graph which contain no P_4 as a subgraph.

The extremal graphs are shown for $k = 1, \dots, 12$ on Figure 1.

References

- [1] P. FRANKL, G. Y. KATONA, Extremal k -edge-hamiltonian hypergraphs, *Discrete Mathematics* (2008) **308**, pp. 1415–1424
- [2] M. PAOLI, W. W. WONG, C. K. WONG, Minimum k -Hamiltonian graphs. II., *J. Graph Theory* (1986) **10**, no. 1, pp. 79–95
- [3] D. RAUTENBACH, A linear Vizing-like relation between the size and the domination number of a graph, *J. Graph Theory* (1999) **31**, no. 4, pp. 297–302
- [4] V. G. VIZING, An estimate of the external stability number of a graph, *Dokl. Akad. Nauk SSSR* (1965) **52**, pp. 729–731

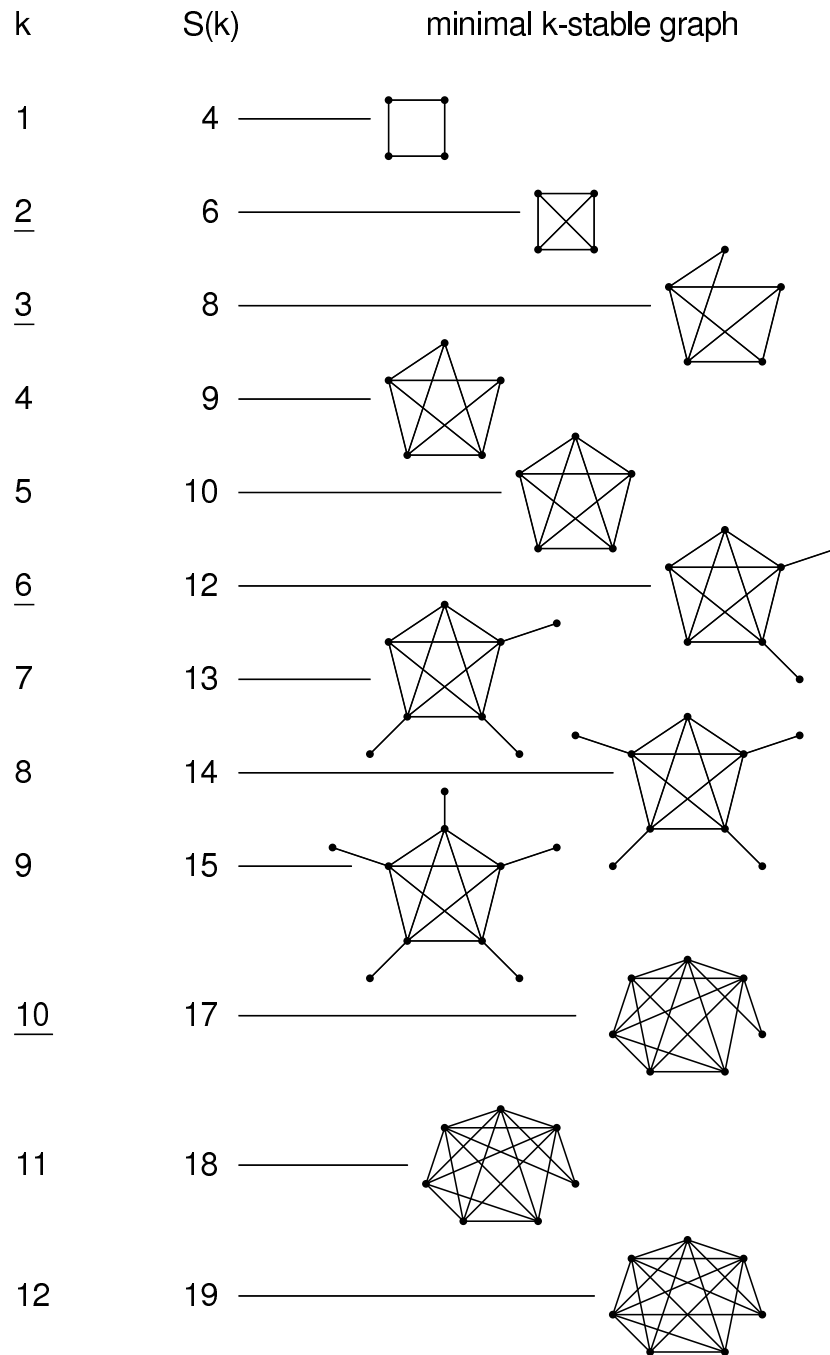


Fig. 1. A display of the values of $S(k)$ and the minimal k -stable graphs for smaller k 's. The values of k where $S(k)$ jumps 2 are marked. Note that the examples are almost- K_n graphs except where $3|n$.

Recognizing edge-perfect graphs: some polynomial instances¹

M. P. Dobson,^a V. A. Leoni,^{a,b,c} G. L. Nasini^{a,b}

^a *Depto. de Matemática. F.C.E.I.A. Universidad Nacional de Rosario, Av. Pellegrini 250, 2000 Rosario, Argentina.*

^b *Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina.*

^c *Corresponding author. E-mail: valeoni@fceia.unr.edu.ar*

Key words: stability number, edge-covering number, odd chordless cycle, polynomial instances

1. Preliminaries and notation

Packing and *covering* games defined by 0, 1 matrices were introduced in [1] as particular classes of combinatorial optimization games. The authors left open for both cases the problem of characterizing matrices defining *totally balanced* games, that is, games for which every induced subgame is balanced.

Van Velzen [4] showed that the only matrices defining totally balanced covering games are clique-node matrices of perfect graphs, proving that they may be recognized in polynomial time. In contrast, a complete characterization of matrices defining totally balanced packing games remains open.

Given a 0, 1 matrix A with column set M , $G(A)$ denotes the *associated graph* of A , that is, the graph with vertex set M and where two vertices are adjacent if there is a row in A with two 1's in their corresponding positions.

In Escalante et al. [2] it is proved that given a matrix A , the *edge-perfection* of $G(A)$ gives a sufficient condition for A to define a totally balanced packing game. In case A has exactly two ones per row, this condition is also sufficient. The authors left open the problem of characterizing edge-perfect graphs.

In this work we give two characterizations of edge-perfect graphs, present some known classes of graphs in which the edge-perfection recognition is poly-

¹ Partially supported by grants of ANPCyT-PICT2005 38036 and CONICET PIP 5810.

nomial and derive sufficient conditions for a graph in order to be polynomially edge-perfect recognized.

Throughout this work, graphs are simple and connected. Most notation and conventions are similar to those in [5].

Let $G = (V(G), E(G))$ be a graph. For $v \in V(G)$, $N_G(v)$ denotes the neighborhood of v in G and $d_G(v) = |N_G(v)|$, the degree of v . A vertex $v \in V(G)$ is a *pendant* if $d_G(v) = 1$. Vertices v and w are *twins*, if $N_G(v) = N_G(w)$. We call *two-twin pair*, a pair of twins in G of degree exactly two.

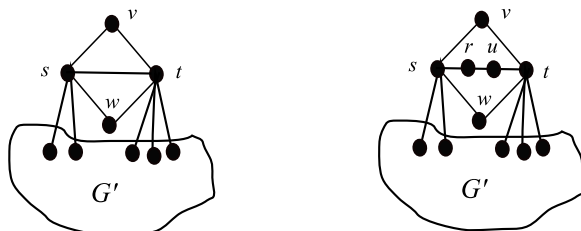


Fig. 1. A scheme of graphs with a two-twin pair: $\{v, w\}$.

Given $T \subseteq V(G)$, $G \setminus T$ denotes the *induced* subgraph of G by the vertices in $V(G) \setminus T$, that is, the subgraph obtained by the deletion of the elements in T . Following the terminology introduced in [2], $G' = G \setminus T$ with $T \subseteq V(G)$ is an *edge-subgraph* if T is the union of the endpoints of the edges in some subset of $E(G)$.

Denoting by $\alpha(G)$ and $\rho(G)$ the *stability* and *edge-covering* numbers of G , respectively, it is known that $\alpha(G) \leq \rho(G)$. When the equality holds, G is called *edge-good*. Besides, G is *edge-perfect* if G' is edge-good, for every edge-subgraph of G . From Knig's edge cover theorem [3], bipartite graphs are edge-good. Hence, we have that bipartite graphs are edge-perfect.

2. Characterizations of edge-perfect graphs

Let us first characterize those induced subgraphs of G which are edge-subgraphs.

Proposition 2.1. Let G be a graph and $G' = G \setminus T$ with $T \subseteq V(G)$. Then, G' is an edge-subgraph of G if and only if for all $v \in T$, $N_G(v) \cap T \neq \emptyset$.

Given an induced subgraph $G' = G \setminus T$ of G , we will say that a vertex v is a *savior* of G' if $v \in T$ and $N_G(v) \cap T = \emptyset$, or equivalently, $N_G(v) \subseteq V(G')$.

The following simple result will become a fundamental property on the way of finding a characterization of edge-perfect graphs.

Lemma 2.2. Let G be an edge-perfect graph and G' a edge-subgraph of G which is not edge-good. Then there exists a savior of G' .

We state a first characterization for edge-perfect graphs, whose proof is based on lemma 2.2 together with some technical results, the most relevant listed below.

Theorem 2.3. A graph G is edge-perfect if and only if for every odd chordless cycle C of G , C has a pendant savior, or there exists a two-twin pair $\{v, w\}$ with $N_G(v) = N_G(w) \subseteq V(C)$.

The proof makes use of the following results:

- (i) Let $v \in V(G)$ be a pendant vertex. If $N_G(v) = \{w\}$ and $G' = G \setminus \{v, w\}$, then G is edge-good if and only if G' is edge-good.
- (ii) If $\{v, w\}$ is a two-twin pair of G with $N_G(v) = N_G(w) = \{s, t\}$, it holds:
 - G is edge-good if and only if $G' = G \setminus \{v, w, s, t\}$ is edge-good.
 - Let $u \in V(G)$ with $u \neq v$ such that $\{u, w\}$ is a two-twin pair of G . Then, G is edge-perfect if and only if $G \setminus \{w\}$ is edge-perfect.
- (iii) If G is an edge-perfect graph, then every triangle induced subgraph T of G has a pendant savior, or one of its edges is the diagonal of a kite like the one induced by $\{v, s, w, t\}$ in the first picture of figure 1.

The condition given by the theorem above does not seem to be easy to check for an arbitrary non bipartite graph. This motivates us to analyze further the structure of certain edge-subgraphs of a given graph.

Given a graph G , let us denote by $\{w_j^1, w_j^2\}$ for $j = 1, \dots, k$, all pairs of two-twins and by $\{z_j\}$ for $j = 1, \dots, h$, all pendant vertices of G . Also, denote by $P_j = N_G(w_j^1) = N_G(w_j^2)$, for $j = 1, \dots, k$, $\{r_j\} = N_G(z_j)$, for $j = 1, \dots, h$, $W = \bigcup_{j=1}^k \{w_j^1, w_j^2\}$, $Z = \bigcup_{j=1}^h \{z_j\}$, $P = \bigcup_{j=1}^k P_j$ and $R = \bigcup_{j=1}^h \{r_j\}$.

Consider the edge-subgraph $G^\emptyset = G \setminus (W \cup P \cup Z \cup R)$ and, for $K \subseteq P$ denote by G^K , the subgraph induced by the vertices in $V(G^\emptyset) \cup K$.

We may state another characterization for edge-perfect graphs:

Theorem 2.4. G is edge-perfect if and only if G^K is bipartite, for all $K \subseteq P$ with $|K \cap P_j| \leq 1$, for all $j = 1, \dots, k$.

3. Polynomial instances

Since the edge-perfection of a graph with at most four vertices is easily verified, we consider graphs with at least five vertices.

Although checking the condition in each of the characterizations given by theorems 2.3 and 2.4 could be exponential, we study some families of graphs for which this task becomes polynomial. Clearly we have the following:

Lemma 3.1. If G has no two-twin pair, then G is edge-perfect if and only if G^\emptyset is bipartite.

From this result it is clear that, for every graph with minimum degree at least three, we can decide in polynomial time if it is edge-perfect or not. Moreover, we can derive the polynomiality of the edge-perfection recognition problem on some known classes of graphs. For example, *quasi-line* graphs and *outerplanar* graphs.

On the other hand and based on the result in theorem 2.3, families of graphs with a polynomial number of odd chordless cycles also define instances where the edge-perfection recognition problem is polynomial. For example, the well-known class of perfect graphs.

Finally, some other polynomial instances may be derived from certain connectivity conditions.

Theorem 3.2. Let G be a graph such that for every $v \in P$ with $d_G(v) \geq 4$, $N_{G\{v\}}(v) \cap V(G^\emptyset) \neq \emptyset$. If there exists $K \subseteq P$ with $|K \cap P_j| \leq 1$ for all $j = 1, \dots, k$ and G^K bipartite and connected, then recognizing if G is edge-perfect is polynomial.

We have presented a wide family of instances where the edge-perfection recognition problem is polynomial. Even though, its computational complexity remains open.

References

- [1] X. Deng, T. Ibaraki and H. Nagamochi, *Algorithms aspects of the core of combinatorial optimization games*. Mathematics of Operations Research **24** (3) (1999) 751-766.
- [2] M. Escalante, V. Leoni and G. Nasini, *A graph theoretical model for total balancedness of combinatorial games*, submitted to Graphs and Combinatorics, 2009.
- [3] D. Knig, *Graphen und Matrizen*. Math. Lapok, **38** (1931) 116-119.
- [4] B. van Velzen, *Discussion Paper: Simple Combinatorial Optimisation Cost Games*. Tilburg University (The Netherlands). ISSN 0924-7815. (2005)
- [5] D. West, *Introduction to Graph Theory*, Prentice Hall, 2001, 2nd. ed.

Some infinite families of Q -integral graphs

Maria Aguieiras A. de Freitas,^a Nair M. Maia de Abreu,^a
Renata R. Del-Vecchio^b

^a*Federal University of Rio de Janeiro, Brazil*
magueiras@im.ufrj.br , nair@pep.ufrj.br

^b*Fluminense Federal University, Brazil*
renata@vm.uff.br

Key words: signless Laplacian, Q -integral graph, double graph, hierarchical product

1. Introduction

Let $G = (V, E)$ be a simple graph on n vertices. The *signless Laplacian matrix* of G is $Q(G) = A(G) + D(G)$, where $A(G)$ is its adjacency matrix and $D(G) = \text{diag}(d_1, \dots, d_n)$ is the diagonal matrix of the vertex degrees in G [1]. The characteristic polynomial of $Q(G)$, $P_Q(G, x)$, is called the *Q -polynomial of G* and its roots are the *Q -eigenvalues of G* . The *spectrum of $Q(G)$* , $Sp_Q(G) = (q_1, \dots, q_{n-1}, q_n)$, is the sequence of the eigenvalues $q_i, i = 1, \dots, n$, of $Q(G)$ displayed in non-increasing order. Recently, studies about the signless Laplacian matrix have been appeared in the literature and some results related to $Q(G)$ and its spectrum can be found in [1], [2] and [3].

If all Q -eigenvalues of G are integer numbers, G is called a *Q -integral graph* and we can find some few classes of these graphs in [2; 3]. Our aim in this paper is to build new families of Q -integral graphs, obtained by three different operations: double graph, inserting edges between two copies of complete graphs and hierarchial products of graphs.

2. Infinite families of Q -integral graphs

For $i = 1, 2$, let $G_i = (V_i, E_i)$ be graphs on n_i vertices. The *union* of G_1 and G_2 is the graph $G_1 \cup G_2$ such that the vertex set is $V_1 \cup V_2$ and the edge set is $E_1 \cup E_2$. The *cartesian product* of G_1 and G_2 is the graph $G_1 \times G_2$, such that the vertex set is $V = V_1 \times V_2$ and where two vertices (u_1, u_2) and (v_1, v_2) are adjacent if and only

if u_1 is adjacent to v_1 in G_1 and $u_2 = v_2$ or $u_1 = v_1$ and u_2 is adjacent to v_2 in G_2 . It is well known that the operations of union and cartesian product of graphs preserve the property of integrality and Laplacian integrality of graphs. As consequence of our result below, we have that these operations also preserve the Q -integrality of graphs.

Theorem 4. For $i = 1, 2$, let G_i be graphs on n_i vertices, with Q -eigenvalues $q_{i,1}, \dots, q_{i,n_i}$. Then the Q -eigenvalues of $G_1 \cup G_2$ and $G_1 \times G_2$ are $q_{1,1}, \dots, q_{1,n_1}, q_{2,1}, \dots, q_{2,n_2}$ and $q_{1,j} + q_{2,k}, j = 1, \dots, n_1, k = 1, \dots, n_2$, respectively.

It is clear that one can generate infinitely many examples of Q -integral graphs by successive applications of the operations above between Q -integral graphs. On the next results we show other ways of building Q -integral graphs.

If $G = (V, E)$ is a graph on n vertices, the *double graph* of G , $\mathcal{D}[G]$, is the graph whose vertex set is $V(\mathcal{D}[G]) = V \times \{0, 1\}$ and where two vertices (i_1, j_1) and (i_2, j_2) are adjacent if and only if the vertices i_1 and i_2 are adjacent in G [5]. Figure 1 shows the graph $\mathcal{D}[K_3]$.

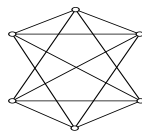


Fig. 1. $\mathcal{D}[K_3]$

The *Kronecker product* of matrices A and B , $A \otimes B$, is the block matrix obtained by replacing each enter a_{ij} of A by the matrix $a_{ij}B$ for all i and j . The adjacency matrix of $\mathcal{D}[G]$ can be represented as $A(\mathcal{D}[G]) = \mathbb{J}_2 \otimes A(G)$, where \mathbb{J}_2 is the all ones 2×2 matrix. Then the signless Laplacian matrix of $\mathcal{D}[G]$ is given by $Q(\mathcal{D}[G]) = \mathbb{I}_2 \otimes (Q(G) + D(G)) + (\mathbb{J}_2 - \mathbb{I}_2) \otimes A(G)$, where $Q(G)$ the signless Laplacian matrix of G and $D(G) = \text{diag}(d_1, \dots, d_n)$ is the diagonal matrix of vertex degrees in G .

In [5], the spectra of $A(\mathcal{D}[G])$ and $L(\mathcal{D}[G]) = A(\mathcal{D}[G]) - D(\mathcal{D}[G])$ were determined in terms of the vertex degrees of G and $A(G)$ and $L(G)$ -spectra. It was shown that $\mathcal{D}[G]$ is an integral (Laplacian integral) graph if and only if G is an integral (Laplacian integral) graph. We prove that this property can also be extended to Q -integral graphs.

Theorem 5. The Q -spectrum of $\mathcal{D}[G]$ is given by $2d_1, \dots, 2d_n, 2q_1, \dots, 2q_n$, where d_1, \dots, d_n are the vertex degrees of G and q_1, \dots, q_n are the Q -eigenvalues of G . Consequently, $\mathcal{D}[G]$ is a Q -integral graph if and only if G is Q -integral.

Let $\mathcal{D}^1[G] = \mathcal{D}[G]$ and, for $i \in \mathbb{N}$, $\mathcal{D}^{i+1}[G] = \mathcal{D}[\mathcal{D}^i[G]]$. Based on the theorem above, for each Q -integral graph G , $\{\mathcal{D}^i[G], i \in \mathbb{N}\}$ is an infinite family of Q -integral graphs.

Let KK_n^j be the graph obtained from two copies of the complete graph K_n by adding j edges between one vertex of a copy of K_n and j vertices of the other copy. For $j = 2$, we obtain the graph KK_n , which was introduced in [4]. Figure 2 displays the graph KK_6^2 .

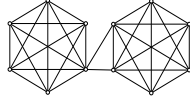


Fig. 2. KK_6^2

Theorem 6. If $j \leq n \in \mathbb{N}$, $n \geq 3$, the characteristic polynomial of $Q(KK_n^j)$ is $P_Q(KK_n^j, x) = (x - 2n + 2)(x - n + 1)^{j-1}(x - n + 2)^{2n-j-2}(x^2 - (3n + j - 3)x + 2(n^2 + n(j - 2) - 2j + 1))$.

As an immediate consequence of the above theorem, we have the following characterization:

Corollary 1. The graph KK_n^j is Q -integral if and only if $(n - j - 1)^2 + 8j$ is a perfect square.

The next result provides new infinite families of Q -integral graphs.

Corollary 2. For $n, k, j \in \mathbb{N}$, if one of the conditions below is satisfied, the KK_n^j graph is Q -integral:

- (i) $n = 3j$;
- (ii) $n = 2j - 1$;
- (iii) $n = 5k - 2$ and $j = 3k$;
- (iv) $n = 3k + 6$ and $j = 2k + 6$;
- (v) $n = j = \frac{k(k+1)}{2}$.

In [6], the *hierarchical product* $H = G_2 \square G_1$ of two graphs G_1 and G_2 having root vertices, labeled 0, is defined as the graph whose the vertex set is $V = V_2 \times V_1$ such that (u_2, u_1) is adjacent to (v_2, v_1) if and only if u_1 is adjacent to v_1 in G_1 or $u_1 = v_1 = 0$ and u_2 is adjacent to v_2 in G_2 . Note that this definition depends on the specified root of G_1 . Figure 2 shows the graph $K_3 \square K_4$.

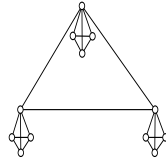


Fig. 3. $K_3 \square K_4$

Under some appropriated labelling of its vertices, the adjacency matrix of $G_2 \square G_1$ can be given by $D_1 \otimes A(G_2) + A(G_1) \otimes \mathbb{I}_{n_2}$, where n_2 is the order of G_2 and $D_1 = \text{diag}(1, 0, \dots, 0)$ has size $n_2 \times n_2$. Then the signless Laplacian matrix of $G_2 \square G_1$

is represented by $Q(G_2 \sqcap G_1) = D_1 \otimes Q(G_2) + Q(G_1) \otimes \mathbb{I}_{n_2}$. In the special case where $G_1 = K_2$, we obtain the following result.

Theorem 7. Let G be a graph on n vertices whose Q -eigenvalues are q_1, \dots, q_n . Then the Q -eigenvalues of $G \sqcap K_2$ are $\frac{q_i + 2 \pm \sqrt{q_i^2 + 4}}{2}$, for $i = 1, \dots, n$.

As an immediate consequence of this, we have that $G \sqcap K_2$ is not a Q -integral graph, for every connected graph G on $n \geq 2$ vertices.

Theorem 8. Let $j, n \in \mathbb{N}$, $n \geq 3$. The characteristic polynomial of $Q(K_j \sqcap K_n)$ is $P_Q(K_j \sqcap K_n, x) = (x - n + 2)^{(n-2)j} (x^2 - (3n + j - 6)x + 2n^2 - 10(n - 1) + j(2n - 3))^{j-1} (x^2 - (3n + 2j - 6)x + 2n^2 - 10(n - 1) + 2j(2n - 3))$.

From the theorem above, we obtain the following characterization:

Corollary 3. Let $j, n \in \mathbb{N}$, $n \geq 3$. The graph $K_j \sqcap K_n$ is Q -integral if and only if $(n - j + 2)^2 + 4(j - 2)$ and $(n - 2j + 2)^2 + 8(j - 1)$ are perfect squares.

Based on the last corollary, we can see that hierarchical product of complete graphs can be Q -integral graph or not. For example, $\{K_{i(i+1)} \sqcap K_{i(i+1)+1}, i \in \mathbb{N}\}$ is an infinite family of Q -integral graphs, while, for every $n \geq 2$, the graph $K_n \sqcap K_n$ is not Q -integral.

References

- [1] D. Cvetković, P. Rowlinson, S. Simić, “Signless Laplacian of finite graphs”, *Linear Algebra and its Applications* 423 (2007) 155-171.
- [2] S. Simić, Z. Stanić, “ Q -integral graphs with edge-degrees at most five”, *Discrete Math.* 308 (2008) 4625-4634.
- [3] Z. Stanić, “There are exactly 172 connected Q -integral graphs up to 10 vertices”, *Novi Sad J. Math.* 37 n. 2 (2007) 193-205.
- [4] D. Stevanović, I. Stanković, M. Milosević, “More on the relation between energy and Laplacian energy of graphs”, *MATCH Commun. Math. Comput. Chem.* 61 n. 2 (2009) 395-401.
- [5] E. Munarini, C.P. Cippo, A. Scagliola, N.Z. Salvi, “Double graphs”, *Discrete Math.* 308 (2008) 242-254.
- [6] L. Barrière, F. Comellas, C. Dalfó, M.A. Fiol, “The hierarchical product of graphs”, *Discrete Appl. Math* 157 (2009) 36-48.

Applications

Lecture Hall A

Wed 3, 10:30–12:30

Balanced clustering for efficient detection of scientific plagiarism

Alberto Ceselli,^a Roberto Cordone,^a Marco Cremonini^a

^a*DTI - Università degli Studi di Milano, Italy*

{alberto.ceselli, roberto.cordone, marco.cremonini}@unimi.it

Key words: Clustering, Dynamic programming, Tabu Search, Branch-and-bound

1. Introduction

Duplication, co-submission and plagiarism are rising phenomena in modern scientific publishing, as the number of peer-reviewed journals and the perceived chances of escaping detection are increasing. On the other side, electronic indexes and new text-searching tools such as the search engine eTBLAST might provide an effective deterrent of unethical publications. Though manual inspection is unavoidable in the end, automatic detection might strongly reduce the work required. However, the size of online databases makes a full search impractical even by algorithmic tools. In this paper, we consider the problem of structuring a textual database so as to optimize queries for potential duplicates.

2. Formulations and properties

The database is modelled as a set N of n elements, with a distance function $d : N \times N \rightarrow \mathbb{R}^+$ enjoying symmetry and triangle inequality ($d_{ij} = d_{ji}$ and $d_{ij} \leq d_{ik} + d_{kj}$ for all $i, j, k \in N$). A trivial duplication check would compare the new item i to each element $j \in N$, to ascertain whether their distance d_{ij} exceeds or not a suitable threshold δ . To reduce the effort one can select s “centres” j_1, \dots, j_s , partition N into clusters C_{j_1}, \dots, C_{j_s} associated to the centres and assign a “radius” $r_j = \max_{k \in C_j} d_{kj}$ to each cluster. Then, i can be compared to each centre j : if $d_{ij} > r_j + \delta$, the triangle inequality guarantees that i is not a duplicate of any element in C_j . Otherwise, one should compare i to all elements in C_j .

Definition 3. *Ordering constraint:* if an element i is assigned to a centre j , all elements k such that $d_{kj} \leq d_{ij} + \delta$ (that is, closer or slightly farther from the centre)

must also be assigned to j . Therefore, all elements k such that $d_{kj} \leq r_j + \delta$ are assigned to j .

For the sake of simplicity, in the following we assume $\delta = 0$, thus searching only for exact duplicates under the given metric.

Proposition 1. Under the ordering constraint with $\delta = 0$, a duplicate item i satisfies condition $d_{ij} \leq r_j + \delta$ for a single centre j .

In this case, then, at most one cluster must be fully explored and the total worst-case computational effort is given by the comparisons to the centres (proportional to the number of clusters s), plus the comparison to the only candidate cluster (proportional to the largest cardinality).

The resulting problem consists in partitioning set N into clusters under the ordering constraint and minimizing the sum of the number of clusters plus the maximum cardinality of a cluster. One can formulate the problem as follows: let $z_{ij} = 1$ if element i is the farthest element of a cluster centred in element j ; $z_{ij} = 0$ otherwise; let η be the cardinality of the largest cluster.

$$\min f = \sum_{i \in N} \sum_{j \in N} z_{ij} + \eta \quad (2.1a)$$

$$\sum_{j \in N} \sum_{k \in E_{ij}} z_{kj} = 1 \quad i \in N \quad (2.1b)$$

$$\sum_{i \in N} |I_{ij}| z_{ij} \leq \eta \quad j \in N \quad (2.1c)$$

$$z_{ij} \in \{0, 1\} \quad i, j \in N \quad (2.1d)$$

where $E_{ij} = \{k \in N : d_{kj} \geq d_{ij}\}$ and $I_{ij} = \{k \in N : d_{kj} \leq d_{ij}\}$.

But one can also set $x_{ij} = 1$ if element i belongs to a cluster centred in element j ; $x_{ij} = 0$ otherwise, and set η as the cardinality of the largest cluster.

$$\min f = \sum_{i \in N} x_{ii} + \eta \quad (2.2a)$$

$$\sum_{j \in N} x_{ij} = 1 \quad i \in N \quad (2.2b)$$

$$\sum_{i \in N} x_{ij} \leq \eta \quad j \in N \quad (2.2c)$$

$$x_{ij} \leq x_{kj} \quad i \in N, j \in N, k \in I_{ij} \quad (2.2d)$$

$$x_{ij} \in \{0, 1\} \quad i, j \in N \quad (2.2e)$$

Proposition 2. Formulations (2.1) and (2.2) are equivalent.

Formulation (2.1) is much faster to solve because it has $O(n^2)$ constraints instead of $O(n^3)$. Both formulations are, most of the time, very weak: the lower bound is just slightly larger than 2. However, they can easily be strengthened by

additional linear constraints and by fixing to zero a number of variables.

Proposition 3. The continuous relaxation of Formulation (2.1) can be strengthened by including the linearization of constraint $\eta s \geq n$, where $s = \sum_{i,j \in N} z_{ij}$.

Proposition 4. For all feasible solutions whose cost is strictly lower than f_H

$$z_{ij} = 0 \quad \text{for all } i, j \in N : |I_{ij}| > \frac{f_H - 1 + \sqrt{(f_H - 1)^2 - 4n}}{2}$$

Proposition 5. The problem is \sqrt{n} -approximable by two trivial algorithms: a) build a single cluster of n elements, b) build n singleton clusters.

Proposition 6. If N is a set of points on a line, i. e. $\exists \pi_i : N \rightarrow \mathbb{R}$ such that $d_{ij} = |\pi_j - \pi_i|$ for all $i, j \in N$, then the problem can be solved in $O(n^3)$ time.

3. Algorithms

The exact solution of the problem is strongly supported by the availability of good heuristic solutions, which allow to prune branching nodes and force to zero some decision variables (Proposition 4). We implemented a greedy algorithm and a Tabu Search to obtain such solutions and applied the commercial *MIP* solver CPLEX 11.0 to Formulation (2.1), strengthened as described.

Greedy algorithm Each step of the greedy algorithm builds a feasible cluster by selecting a pair (i, j) and setting i as the farthest element of a new cluster centred in j (i.e., $z_{ij} = 1$). The chosen pair maximises the cardinality $|I_{ij}|$ of the resulting cluster. In case of ties, the algorithm selects the pair which leaves the highest number of feasible pairs after the fixing. To avoid building excessively large clusters, and to reduce the computational complexity, it is forbidden to select pairs with $|I_{ij}| > \alpha\sqrt{n}$ where $\alpha \geq 1$ is a suitable coefficient. The purpose is to build as many clusters as possible with a size as close as possible to the ideal value \sqrt{n} .

Tabu Search algorithm The Tabu Search algorithm is based on the following neighbourhood: each pair (i, j) corresponds either to shrinking a current cluster (if j is a centre and i is already assigned to it) or to inflating it (possibly, to creating a new cluster). In the first case, after shrinking the selected cluster, we apply the greedy heuristic to obtain a complete solution. In the second case, we first shrink all cluster which include the elements assigned to the inflated (or newly created) cluster and then complete the solution with the greedy heuristic. All pairs (i, j) with $|I_{ij}| > \beta\sqrt{n}$ (where $\beta \geq \alpha$) are forbidden, to avoid building excessively

large clusters and to reduce the computational complexity. Thus, the neighbourhood includes $O(n\sqrt{n})$ solutions. The tabu mechanism saves for each pair (i, j) the last iteration in which the current solution included cluster (i, j) . For a specified number of iterations L (*tabu tenure*), the move based on pair (i, j) is forbidden, unless the resulting solution improves the best known one (*aspiration criterion*). At each step, the whole neighbourhood is visited and the best non tabu solution (or the best tabu solution respecting the aspiration criterion) is accepted as the incumbent one. The algorithm terminates after a specified total number of iterations I .

4. Results

We have generated 25 Euclidean 3D instances, of five different sizes (from 50 to 250 elements by steps of 50) and 25 instances (with the same sizes) in which random distances are generated and the triangle inequality is enforced by replacing direct distances by the lengths of shortest paths.

Parameter α proves relevant for the greedy algorithm: the gap with respect to the best known result decreases from 23.8% for $\alpha = 0.8$ to 19.5% for $\alpha = 1.1$, then increasing up to 25.5% for $\alpha = 1.2$ (it is better to start with clusters slightly larger than the ideal value). The computational time on a 1.59GHz Intel Core 2 laptop with 2 GB of RAM is always lower than 0.1 seconds.

Parameter β has a similar influence on the Tabu Search: the gap decreases from 2.0% ($\beta = 1.1$) to 1.4% ($\beta = 1.2$), then increasing up to 1.7% ($\beta \geq 1.4$). The total number of iterations I has been fixed to 200 and the tabu tenure L to 10 (lower values induced cyclic behaviours on some instances). The computational time for the largest instances ranges from about 10 minutes ($\beta = 1.1$) to about 25 ($\beta = 1.5$). Presently, the role of Tabu Search is not fully clarified: the objective decreases in the first steps, usually stabilizing on the final value with few (if any) intermediate increases. This suggests that the problem might be characterized by large plateaus, in which, besides avoiding cycles, some further guiding mechanism might be relevant.

The formulation strongly gains from the additional cutting planes and variable fixings: all Euclidean instances up to $n = 100$ and all but two of the random instances with $n \leq 200$ could be solved exactly in a limit time of 10 minutes. The average final gap is about 12%. The truncated branch-and-bound improved the result produced by Tabu Search on 13 of the 50 instances.

As a final test, we have created an instance from real-world data by selecting five papers from the combinatorial optimization literature, including a paper by J.B. Shearer plagiarized by D. Marcu. We have split these papers in subsections, keeping only alpha-numeric characters, obtaining an instance of 55 blocks. We have

defined the distance between two blocks i and j , respectively consisting of I and J characters, as follows:

$$d_{ij} = \max\{I, J\} - \text{LCS}(i, j)$$

where $\text{LCS}(i, j)$ denotes the length of the Longest Common Subsequence of characters between i and j . This distance function showed some predictive power, being able to correctly identify the plagiarism by Marcu.

We have been able to find the optimal solution of value 18 to this instance in few seconds by using model (2.1a) – (2.1d) and CPLEX 11. Our Greedy and Tabu Search algorithms found respectively a value of 20 and 18 in fractions of a second, confirming our former computational experience.

Appendix (Proofs of the propositions)

Proposition 1 *Under the ordering constraint with $\delta = 0$, a duplicate item i satisfies condition $d_{ij} \leq r_j + \delta$ for a single centre j .*

proof 1. Let $\delta = 0$ and i be a duplicate of element $k \in N$. By contradiction, let i be close to two centres ($d_{ij_1} \leq r_{j_1} + \delta = r_{j_1}$ and $d_{ij_2} \leq r_{j_2} + \delta = r_{j_2}$). Then $d_{kj_1} \leq d_{ki} + d_{ij_1} \leq r_{j_1}$ and $d_{kj_2} \leq d_{ki} + d_{ij_2} \leq r_{j_2}$. The ordering constraint requires to assign k to both centres, which is unfeasible.

Proposition 2 *Formulations (2.1) and (2.2) are equivalent.*

proof 2. Start from any feasible solution to Formulation (2.2) and define $x_{ij} = \sum_{k \in E_{ij}} z_{kj}$.

$$\begin{aligned} \min f &= \sum_{i \in N} \sum_{k \in E_{ii}} z_{ki} + \eta \\ &\sum_{j \in N} \sum_{k \in E_{ij}} z_{kj} = 1 \quad i \in N \\ &\sum_{i \in N} \sum_{k \in E_{ij}} z_{kj} \leq \eta \quad j \in N \\ &\sum_{k \in E_{ij}} z_{kj} \leq \sum_{k \in E_{lj}} z_{kj} \quad i \in N, j \in N, l \in I_{ij} \\ &\sum_{k \in E_{ij}} z_{kj} \in \{0, 1\} \quad i, j \in N \end{aligned}$$

Since $E_{ii} = N$ and $\sum_{i \in N} \sum_{k \in E_{ij}} z_{kj} = \sum_{k \in N} \sum_{i \in I_{kj}} z_{kj}$

$$\begin{aligned} \min f &= \sum_{i \in N} \sum_{k \in N} z_{ki} + \eta \\ &\sum_{j \in N} \sum_{k \in E_{ij}} z_{kj} = 1 \quad i \in N \\ &\sum_{k \in N} \sum_{i \in I_{kj}} z_{kj} \leq \eta \quad j \in N \\ &\sum_{k \in E_{ij}} z_{kj} \leq \sum_{k \in E_{lj}} z_{kj} \quad i \in N, j \in N, l \in I_{ij} \\ &\sum_{k \in E_{ij}} z_{kj} \in \{0, 1\} \quad i, j \in N \end{aligned}$$

The third constraint is redundant ($E_{lj} \supseteq E_{ij}$ for $l \in I_{ij}$) and the fourth can be reduced to $z_{kj} \in \{0, 1\}$. Suitably changing the names of some indexes, one obtains Formulation (2.1). Therefore, any feasible solution to Formulation (2.2) corresponds to a feasible solution of identical cost to Formulation (2.1).

The converse is also true. Start from any feasible solution to Formulation (2.1) and define $z_{ij} = x_{ij} - x_{\sigma_i j}$ where σ_i is the successor of i in the list of the elements sorted by increasing distances from j (let $z_{ij} = x_{ij}$ when i is the farthest element from j).

$$\begin{aligned} \min f &= \sum_{i \in N} \sum_{j \in N} (x_{ij} - x_{\sigma_i j}) + \eta \\ &\sum_{j \in N} \sum_{k \in E_{ij}} (x_{kj} - x_{\sigma_k j}) = 1 && i \in N \\ &\sum_{i \in N} |I_{ij}| (x_{ij} - x_{\sigma_i j}) \leq \eta && j \in N \\ &(x_{ij} - x_{\sigma_i j}) \in \{0, 1\} && i, j \in N \end{aligned}$$

Since $\sum_{i \in N} (x_{ij} - x_{\sigma_i j}) = x_{jj}$ and $\sum_{k \in E_{ij}} (x_{kj} - x_{\sigma_k j}) = x_{ij}$

$$\begin{aligned} \min f &= \sum_{j \in N} x_{jj} + \eta \\ &\sum_{j \in N} x_{ij} = 1 && i \in N \\ &\sum_{i \in N} |I_{ij}| (x_{ij} - x_{\sigma_i j}) \leq \eta && j \in N \\ &(x_{ij} - x_{\sigma_i j}) \in \{0, 1\} && i, j \in N \end{aligned}$$

Since $|I_{\sigma_i j}| = |I_{ij}| + 1$

$$\begin{aligned} \min f &= \sum_{j \in N} x_{jj} + \eta \\ &\sum_{j \in N} x_{ij} = 1 && i \in N \\ &\sum_{i \in N} [|I_{ij}| x_{ij} - (|I_{\sigma_i j}| - 1) x_{\sigma_i j}] \leq \eta && j \in N \\ &x_{ij} \geq x_{\sigma_i j} && i, j \in N \\ &x_{ij} \in \{0, 1\} && i, j \in N \end{aligned}$$

which yields Formulation (2.1).

Proposition 3 *The continuous relaxation of Formulation (2.1) can be strengthened by including the linearization of constraint $\eta s \geq n$, where $s = \sum_{i,j \in N} z_{ij}$.*

proof 3. Since s is the number of cluster and η their maximum cardinality, constraint $\eta s \geq n$ states that the number of elements does not exceed their product. The convex hull of the integer points respecting this condition is a politope providing valid inequalities for the problem.

Proposition 4 For all feasible solutions whose cost is strictly lower than f_H

$$z_{ij} = 0 \quad \text{for all } i, j \in N : |I_{ij}| > \frac{f_H - 1 + \sqrt{(f_H - 1)^2 - 4n}}{2}$$

proof 4. If the solution is better than f_H , then it satisfies both $s + \eta \leq f_H - 1$ and $\eta s \geq n$. This implies that $\eta \leq \frac{f_H - 1 + \sqrt{(f_H - 1)^2 - 4n}}{2}$. As no cluster includes more than η elements, i cannot be assigned to centre j when $|I_{ij}| > \eta$.

Proposition 5 The problem is \sqrt{n} -approximable by two trivial algorithms: a) build a single cluster of n elements, b) build n singleton clusters.

proof 5. Since $\eta s \geq n$, the objective is $f = s + \eta \geq s + n/s$. The minimum of $s + n/s$ for $s \geq 0$ is $2\sqrt{n}$. Since both trivial algorithms provide a solution costing $f_{\text{apx}} = n + 1 \leq 2n$, the optimum is at least $f^* \geq 2\sqrt{n}$ and $f_{\text{apx}} \leq \sqrt{n}f^*$.

Proposition 6 If N is a set of points on a line, i. e. $\exists \pi_i : N \rightarrow \mathbb{R}$ such that $d_{ij} = |\pi_j - \pi_i|$ for all $i, j \in N$, then the problem can be solved in $O(n^3)$ time.

proof 6. Due to the ordering constraint, each cluster corresponds to an interval $[\pi_i, \pi_j]$ along the line. Let $\tilde{\pi}_i$ and $\tilde{\pi}_j$ be the coordinates of the elements preceding i and following j , respectively (with $\tilde{\pi}_i = -\infty$ if i is the first element and $\tilde{\pi}_j = +\infty$ if j is the last). Not all such intervals are feasible: to be feasible, an interval must admit a central element $k \in N$ such that $(\tilde{\pi}_i + \pi_j)/2 \leq \pi_k \leq (\pi_i + \tilde{\pi}_j)/2$. All the unfeasible intervals can be filtered out in $O(n^2 \log n)$ time (actually $O(n^2)$ if they are scanned lexicographically).

Now, build an auxiliary graph with a vertex i for each element and an arc (i, j) for each pair of elements such that the interval between i and the element preceding j is feasible. There is a one-to-one correspondence between the paths from the first to the last vertex in this graph and the partitions of N into feasible clusters. For each fixed value of $\eta \in \{1, \dots, n\}$, one can remove the arcs corresponding to the clusters of size exceeding η and determine in $O(n^2)$ time, by a simple breadth-first visit, the shortest path from the first to the last vertex with respect to the number of arcs. By repeating this procedure for all values of η , one can solve the problem to optimality in $O(n^3)$ time.

Robustness in Train Timetabling

V. Cacchiani,^a A. Caprara,^a M. Fischetti^b

^aDEIS, University of Bologna, Italy
{valentina.cacchiani, alberto.caprara}@unibo.it

^bDEI, University of Padova, Italy
fisch@dei.unipd.it

Key words: Train Timetabling, Robustness, Lagrangian relaxation

1. Abstract

We propose a Lagrangian-based heuristic approach to obtain *robust solutions* to the Train Timetabling Problem (TTP) on a corridor (i.e. a single one-way line connecting two major stations). Roughly speaking, we define a solution to be robust if it allows to avoid delay propagation as much as possible. In particular, in the planning phase that we are considering the aim is to build timetables characterized by *buffer times* that can be used to absorb possible delays occurring at an operational level. In the TTP, we are given a set of stations S along the corridor, a set of trains T , and for each train an ideal timetable (i.e. the timetable suggested by the Train Operator). In the nominal TTP the aim is to change the ideal timetables for the trains *as little as possible*, while satisfying the *track capacity constraints* consisting of:

- departure constraints (imposing a minimum headway between two consecutive departures from a station);
- arrival constraints (imposing a minimum headway between two consecutive arrivals at a station);
- overtaking constraints (avoiding overtaking between consecutive stations, since we are considering a single one-way line).

In order to obtain feasible timetables we are allowed to change the departure of any train from its first station (shift) and/or to increase the minimum stopping time in one or more of the visited stations (stretch). Each train is assigned an ideal profit which is gained if it is scheduled according to its ideal timetable. The profit is decreased (according to a linear function) if shift and/or stretch are applied; if the profit becomes null or negative the train is cancelled. A common approach to deal

with the nominal TTP is to discretize the time horizon and to formulate the problem on a space-time graph $G = (V, A)$. Each node of the set V corresponds to a possible time instant at which a train can depart from or arrive at a station. Each arc in A represents the travel of a train from a station to the next one (segment arcs) or the stop of a train at a station (station arcs). Furthermore, each timetable for a train corresponds to a suitable path in G . We refer the reader to [1] for further details on the space-time graph representation. Given the graph G , a common Integer Linear Programming (ILP) formulation based on arc binary variables (see [1]) is the following:

$$\max \sum_{t \in T} \sum_{a \in A^t} p_a x_a \quad (1.1)$$

$$\sum_{a \in \delta^+(\sigma) \cap A^t} x_a \leq 1, \quad t \in T \quad (1.2)$$

$$\sum_{a \in \delta^-(v) \cap A^t} x_a = \sum_{a \in \delta^+(v) \cap A^t} x_a, \quad t \in T, v \in V \setminus \{\sigma, \tau\} \quad (1.3)$$

$$\sum_{a \in C} x_a \leq 1, \quad C \in \mathcal{C}, \quad (1.4)$$

$$x_a \in \{0, 1\}, \quad a \in A \quad (1.5)$$

where:

- A^t is the set of arcs in A that may be used by the path for train t ;
- x_a is a binary variable which assumes value 1 if arc $a \in A^t$ is selected in the solution for train $t \in T$, and 0 otherwise;
- p_a is the profit gained if arc $a \in A^t$ is in the solution
- \mathcal{C} is the collection of all the cliques of incompatible arcs, with respect to track capacity constraints.

The drawback of the nominal problem is that it does not take into account possible delays that can occur in the operational phase and that can affect the feasibility and the quality of the solution. We describe how to change the model so as to move towards robust solutions, that take into account the possible presence of delays. Namely, we want to introduce *buffer times* for absorbing delays: this is obtained by favoring longer stops at the stations with respect to the minimum stopping time, so that, if a “short” delay occurs, it will not be propagated to the following trains. On the other hand, allowing for buffer times that can absorb any reasonable delay would be too conservative and produce solutions which are not acceptable from a practical point of view. The aim of the robust problem is then bi-objective: to maximize the profits of the scheduled trains (*efficiency*) while maximizing the buffer times (*robustness*). The new objective function reads:

$$\max \sum_{t \in T} \sum_{a \in A^t} p_a x_a + \sum_{t \in T} \sum_{a \in A_B^t} b_a x_a \quad (1.6)$$

where $A_B^t \subseteq A^t$ is the set of arcs corresponding to buffer times for train $t \in T$ and b_a is the profit achieved if we select buffer arc $a \in A_B^t$. We consider the Lagrangian relaxation of the constraints (1.4) of the model (1.6), (1.2)-(1.5), and solve it within a subgradient optimization framework. In particular, the Lagrangian objective is:

$$\max \sum_{t \in T} \sum_{a \in A^t} p_a x_a + \sum_{t \in T} \sum_{a \in A_B^t} b_a x_a + \sum_{C \in \mathcal{C}} \lambda_C (1 - \sum_{a \in C} x_a) = \quad (1.7)$$

$$= \sum_{t \in T} \sum_{a \in A_B^t} \bar{p}_a x_a + \sum_{C \in \mathcal{C}} \lambda_C \quad (1.8)$$

Here, for $t \in T$ and $a \in A^t$, $\bar{p}_a := p_a - \sum_{C \in \mathcal{C}_a} \lambda_C + b_a$ (we assume $b_a = 0$ for $a \in A^t \setminus A_B^t$), where $\mathcal{C}_a \subseteq \mathcal{C}$ denotes the subfamily of cliques containing arc a . The Lagrangian relaxation thus calls for finding the maximum profit path in the graph G for each train, in an analogous way as in [1].

We present some preliminary computational results on real-life instances of the corridor Modane-Milan with 100 and 200 trains, respectively, used for a comparison between the described approach and the method proposed in [3]. In [3], Fischetti et al. develop different methods for improving the robustness of a given TTP solution. In particular, they propose an event-based model (by adapting the Periodic Event Scheduling Problem for the periodic case), and investigate different approaches to get robust solutions: stochastic models and a light robustness approach (see [2]). In order to evaluate the robustness of a given solution in a more realistic way, rather than simply considering the second objective above, an external validation method method is used in [3]. Given a TTP solution, the validation method considers different realistic external delay scenarios and, assuming that all the trains in the solution have to be scheduled and all train precedences are fixed, adapts the solution to make it feasible with the given external delays, evaluating the overall resulting delay.

In Table 8, we perform a comparison between our method and the method of [3], by considering the solutions obtained by [3] when the precedences are left unfixed, as is the case in our approach. The comparison is done on the efficiency *Eff.* of the solutions found (the first objective above) and the outcome of the validation method, which provides the cumulative delay *Delay* in the scenarios considered. Our heuristic is run for 1000 iterations, which take less than 2000 seconds, and the method [3] is run with a time limit of 2 hours. The best nominal solutions obtained with the Lagrangian heuristic algorithm of [1] have efficiencies equal to 9297 and 18542, respectively. We mention that the Lagrangian approach obtains one solution per iteration and we report in Table 8 only the values of three selected solutions.

As it can be observed, the Lagrangian-based approach finds solutions of comparable efficiency producing smaller cumulative delays than in [3] (in slightly shorter computing times). Note that the cumulative delay is not necessarily increasing when efficiency increases (e.g., this can be due to a better distribution of the buffer

Instance	#trains	Eff. ([3])	Delay ([3])	Eff.	Delay
MdMI1	100	9209	16683	9220	14331
MdMI1	100	8837	14070	9020	14728
MdMI1	100	8372	12675	8889	14508
MdMI2	200	18437	36376	18041	32962
MdMI2	200	17692	32355	17848	31653
MdMI2	200	16761	29716	16911	24723

Table 8. Comparison between the solutions in [3] and our solutions.

times in a solution with higher efficiency value). Ongoing work consists of testing other instances and different parameter settings in order to produce more robust solutions.

References

- [1] A. Caprara, M. Fischetti and P. Toth, “Modeling and Solving the Train Timetabling Problem”, *Operations Research* 50, 851–861 (2002).
- [2] M. Fischetti and M. Monaci, “Light Robustness”, *Technical Report DEI*, (2008).
- [3] M. Fischetti, D. Salvagnin and A. Zanette, “Fast Approaches to Improve the Robustness of a Railway Timetable”, to appear in *Transportation Science*, (2009).

Combinatorial optimization based recommender systems

Fabio Roda,^a Leo Liberti,^b Franco Raimondi^c

^a*DisMoiOu, 33 rue des Jeuneurs, 75002 Paris, France, and
LIX, École Polytechnique, 91128 Palaiseau, France
fronline@libero.it*

^b*LIX, École Polytechnique, 91128 Palaiseau, France
liberti@lix.polytechnique.fr*

^c*Department of Computer Science, UCL, London, UK
f.raimondi@cs.ucl.ac.uk*

Key words: collaborative filtering, maximum capacity path

1. Introduction

Recommender systems exploit a set of established user preferences to predict topics or products that a new user might like [2]. Recommender systems have become an important research area in the field of information retrieval. Many approaches have been developed in recent years and the interest is very high. However, despite all the efforts, recommender systems are still in need of further development and more advanced recommendation modelling methods, as these systems must take into account additional requirements on user preferences, such as geographic search and social networking. This fact, in particular, implies that the recommendation must be much more “personalized” than it used to be.

In this paper, we describe the recommender system used in the “DisMoiOu” (“TellMeWhere” in French) on-line service (<http://dismoiou.fr>), which provides the user with advice on places that may be of interest to him/her; the definition of “interest” in this context is personalized taking into account the geographical position of the user (for example when the service is used with portable phones such as the Apple iPhone), his/her past ratings, and the ratings of his/her neighbourhood in a known social network.

Using the accepted terminology [6], DisMoiOu is mainly a Collaborative Filtering System (CFS): it employs opinions collected from similar users to suggest likely places. By contrast with existing recommender systems, ours puts together

the use of a graph theoretical model [4] and that of combinatorial optimization methods [1]. Broadly speaking, we encode known relations between users and places and users and other users by means of weighted graphs. We then define essential components of the system by means of combinatorial optimization problems on a reformulation of these graphs, which are finally used to derive a ranking on the recommendations associated to pairs (user,place).

We remark that this is work in progress relating the first few months of work in an industrial Ph.D. thesis. Preliminary computational results on the three classical evaluation parameters for recommender systems (accuracy, recall, precision [3]) show that our system performs well with respect to accuracy and recall, but precision results need to be improved.

2. Formalization of the problem

We employ the usual graph-theoretical notation, e.g. for a vertex v of a graph G , $\delta_G^+(v)$, $\delta_G^-(v)$ are the set of vertices adjacent to incoming and respectively outgoing arcs. For vertices u, v of G we also let $\Delta_G(u, v) = \delta_G^+(u) \cap \delta_G^-(v)$.

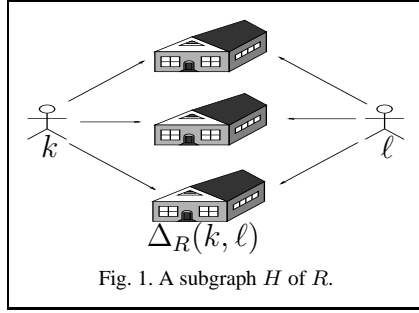
We are given two finite sets U (the users) and P (the places), and a vertex set $V = U \cup P$. We are also given two directed graphs as follows.

- A ratings bipartite digraph $R = (V, A)$ where $A \subseteq U \times P$ is weighted by a function $\rho : A \rightarrow [-1, 1]$, which expresses the ratings of users with respect to the places.
- A social network $S = (U, B)$ weighted by a function $\gamma : B \rightarrow [0, 1]$ which encodes a confidence coefficient between users.

The union of the two graphs $G = R \cup S$ is a mixed ratings/social network which is used to establish new arcs in $U \times U$ or to change the values that γ takes on existing arcs: a missing relation of confidence between two users can be established if both like (almost) the same places in (almost) the same way. Moreover, even when a confidence relation is already part of B , its strength can change according to similar shared preferences situations. This is encoded by the reformulated graph G' described below.

We define a graph G' with vertex set $V' = U \cup P$ and arc set B' (weighted by a function $\gamma' : B' \rightarrow [0, 1]$) defined in the following way.

- For every $k, \ell \in U$ such that $(k, \ell) \notin B$ and subgraph $H = (V_H, A_H)$ of R induced by the vertex set $V_H = \{k, \ell\} \cup \Delta_R(k, \ell)$ (see Fig. 1) such that $A_H \neq \emptyset$, B' contains the arc (k, ℓ) weighted by $\gamma'_{k\ell} = f(\vartheta)$, where



$$\vartheta = \frac{1}{|\Delta_R(k, \ell)|} \sum_{i \in \Delta_R(k, \ell)} |\rho_{ki} - \rho_{\ell i}|. \quad (2.1)$$

ϑ represents the difference between users. The bigger it is, the lower the confidence $\gamma'_{k\ell} \cdot \gamma'_{\ell k}$ is obtained as a function f of ϑ .

- (ii) For every $k, \ell \in U$ such that $(k, \ell) \in B$ and subgraph $H = (V_H, A_H)$ of R induced by the vertex set $V_H = \{k, \ell\} \cup \Delta_R(k, \ell)$ such that $A_H \neq \emptyset$, B' contains the arc (k, ℓ) weighted by $\gamma'_{k\ell} = g(\gamma_{k\ell}, \vartheta)$.

We let $X = (U \times P) \setminus A$ be the set of all recommendations that the system is supposed to be able to make.

2.1 Identification of maximum confidence paths

Given $(k^*, i^*) \in X$, we consider the graph $Z = (W, C)$ where $W = U \cup \{i^*\}$ and $C = B' \cup \{(k, i^*) \mid k \in \delta_R^-(i^*)\}$. Our aim is to compute a ranking for the known ratings $\{\rho_{ki^*} \mid k \in \delta_R^-(i^*)\}$ by means of the confidence relations encoded in the network Z , using paths (or sets thereof) ensuring maximum confidence. By convention, we extend the confidence function γ to arcs in C adjacent to i^* as follows: $\forall k \in \delta_R^-(i^*)$ ($\gamma_{ki^*} = 1$).

We make the assumption that for a path $p \subseteq C$ in Z , $\gamma(p) = \min_{(k, \ell) \in p} \gamma_{k\ell}$, i.e. that the confidence on a path is defined by the lowest confidence arc in the path. This implies that finding the maximum confidence path between k^* and i^* is the same as finding a path whose arc of minimum weight γ is maximum (among all paths $k^* \rightarrow i^*$). Considering Z as a network where γ are capacities on the arcs, a maximum confidence path is the same as a *maximum capacity path* between k^* and i^* , for which there exists an algorithm linear in the number of arcs [5]. The mathematical

programming formulation for the MAXIMUM CAPACITY PATH (MCP) problem is:

$$\left. \begin{array}{l}
 \max_{x,t} \quad t \\
 \text{s.t.} \quad \sum_{\ell \in \delta_R^+(k^*)} x_{k^*\ell} = 1 \\
 \forall \ell \in W \setminus \{k^*, i^*\} \quad \sum_{h \in \delta_R^-(\ell)} x_{h\ell} = \sum_{h \in \delta_R^+(\ell)} x_{\ell h} \\
 \forall (k, \ell) \in C \quad t \leq \gamma_{k\ell} x_{k\ell} + M(1 - x_{k\ell}) \\
 x \in \{0, 1\}, t \geq 0,
 \end{array} \right\} \quad (2.2)$$

where $M \geq \max_{(k,\ell) \in C} \gamma_{k\ell}$. Let $\bar{p} \subseteq C$ be the maximum confidence path (i.e. the set of arcs (k, ℓ) such that $x_{k\ell} = 1$), and $\alpha(\bar{p}) = \operatorname{argmin}\{\gamma_{k\ell} \mid (k, \ell) \in \bar{p}\}$. Removing $\alpha(\bar{p})$ from $C^1 = C$ yields a different set of arcs C^2 with associated network $Z^2 = (W, C^2)$, in which we can re-solve (2.2) to obtain a path \bar{p}^2 as long as Z^2 is connected (otherwise, define $\bar{p}^2 = \emptyset$): this defines an iterative process for obtaining a sequence of triplets (Z^r, \bar{p}^r) . Given a confidence threshold $\Gamma \in [0, 1]$ and an integer $q > 0$, we define the set $\Omega = \{\bar{p}^r \mid \bar{p}^r \neq \emptyset \wedge r \leq q \wedge \gamma_{\alpha(\bar{p}^r)} \geq \Gamma\}$ of all high confidence paths from k^* to i^* .

2.2 Ranking the ratings

Recall each $p \in \Omega$ ends in i^* , so we can define $\lambda : \Omega \rightarrow \delta_R^-(i^*)$ such that $\lambda(p)$ is the last arc of p . Thus, we extend ρ to Ω as follows:

$$\rho(p) = \rho(\lambda(p)).$$

Let $\Theta = \{\sigma \in [-1, 1] \mid \exists p \in \Omega (\sigma = \rho(p))\}$ be the set of ratings for i^* available to k^* . We evaluate each rating by assigning it the sum of the confidences along the corresponding paths. Let $v : \Theta \rightarrow \mathbb{R}_+$ be given by

$$\forall \sigma \in \Theta \quad v(\sigma) = \sum_{\substack{p \in \Omega \\ \rho(p) = \sigma}} \gamma(p).$$

We use v to define a ranking on Θ (i.e. an order $<$ on Θ): for all $\sigma, \tau \in \Theta$ ($\sigma < \tau \leftrightarrow v(\sigma) < v(\tau)$). Naturally, this set-up rests on the fact that $|\Theta| < |\Omega|$, which is exactly what happens in DisMoiOu's implementation. The recommender system then picks the greatest σ in Θ (i.e. the rating with highest associated cumulative confidence) as the recommendations to user k^* concerning the place i^* . Finally, the output of the recommender system is a set of high confidence recommendations for user k^* as i^* ranges in P .

3. Extensions

One of the troubles with the recommender system described in Sect. 2 is that paths in Ω might be too long: although in our formalization paths are only weighted by the value of the arc of minimum confidence, in practice it also makes sense to require that the paths should either be shortest or at least of constrained cardinality, for confidence usually wanes with distance in social networks. Enforcement of the first idea yields a bi-criterion path problem as (2.2) with an additional objective function:

$$\min_{x,t} \sum_{(k,\ell) \in C} x_{k\ell}. \quad (3.3)$$

Enforcement of the second idea (say with paths having cardinality at most K) yields the corresponding constraint:

$$\sum_{(k,\ell) \in C} x_{k\ell} \leq K. \quad (3.4)$$

References

- [1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.
- [2] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, San Francisco, 1998. Morgan Kaufmann.
- [3] C. Cleverdon, J. Mills, and M. Keen. *Factors Determining the Performance of Indexing Systems: ASLIB Cranfield Research Project. Volume 1: Design*. ASLIB Cranfield Research Project, Cranfield, 1966.
- [4] Z. Huang, W. Chung, and H. Chen. A graph model for e-commerce recommender systems. *Journal of the American Society for Information Science and Technology*, 55(3):259–274, 2004.
- [5] A. Punnen. A linear time algorithm for the maximum capacity path problem. *European Journal of Operational Research*, 53:402–404, 1991.
- [6] E. Vozalis and K. Margaritis. Analysis of recommender systems algorithms. In E. Lipitakis, editor, *The 6th Hellenic European Conference on Computer Mathematics & its Applications*, pages 732–745, Athens, 2003. Athens University of Economics and Business.

Bounds and solutions for strategic, tactical and operational ambulance location

Roberto Cordone,^a Federico Ficarelli,^a Giovanni Righini^a

^a*Dipartimento di Tecnologie dell'Informazione*

Università degli Studi di Milano

Via Bramante 65, 26013 Crema, Italy

{roberto.cordone, federico.ficarelli, giovanni.righini}@unimi.it

Key words: Location, Emergency services, Integer Linear Programming

1. Introduction

The organization and management of an emergency health care system requires decisions at different levels and involves several stake-holders and decision-makers, with different and possibly conflicting objectives. The service is typically provided by a fleet of ambulances which are dispatched to the patients' homes upon arrival of phone calls to an operating center. The available ambulances are parked in specially equipped areas, scattered in the territory so as to reach the patient within a limit time (in urban environments, a few minutes). The parking areas must be built in advance, and this is a strategic decision taken by the municipality. Part of the areas are selected to host the available ambulances, and this is a tactical decision taken by the managers at the operating center; the problem is known as Maximum Covering Location Problem [2]. This problem is made harder by the variability of the resources (number of available ambulances) and of the demand, both in time and space: the demand is typically stronger at daytime and during working days than by night or during week-ends; it is more concentrated in residential areas during the night, in areas with working places during the working days.

Furthermore, the number of available ambulances changes along the day as some get busy servicing new calls and others become available again after terminating their service. A compelling operational problem is to optimally cover the territory when some ambulances are busy by re-locating the available ones. In large cities, the calls and the consequent operational decisions are very frequent, and it is advisable to pre-compute optimal solutions for a number of possible scenarios as a useful guideline. This problem has been considered for instance in [3].

In this paper we consider an integrated approach to three decision problems, usually tackled at different levels: the choice of parking areas (strategic level), the location of ambulances in each time slot (tactical level) and the relocation of the available ambulances according to their number (operational level). We present an integer linear programming formulation and some preliminary computational results with general-purpose solvers.

2. Mathematical models

2.1 Model n.1: Parking areas construction and ambulance location

Model 1 integrates the strategic and tactical levels. Let \mathcal{T} denote the set of time slots and \mathcal{I} the set of demand points, where service requests are concentrated; d_{it} is the amount of demand for each point $i \in \mathcal{I}$ and time slot $t \in \mathcal{T}$, A_t ambulances are available in time slot t . Let \mathcal{C} denote the set of candidate parking areas and C the number of areas which can be built. The ambulances can only be assigned to parking places that have been built. Each demand point i can be covered within a prescribed maximum intervention time from a known subset \mathcal{C}_i of parking areas. The problem of selecting the parking areas to build and allocating to them the ambulances available in each time slot, in order to provide maximum coverage to the population, can be modelled by the following binary variables: x_c indicates whether a parking area is built or not in candidate location $c \in \mathcal{C}$; y_{ct} whether an ambulance is assigned or not to parking area $c \in \mathcal{C}$ during time slot t ; z_{it} whether demand point i is covered or not during time slot t .

$$\text{maximize } v = \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} d_{it} z_{it} \quad (2.1a)$$

$$\text{s.t. } z_{it} \leq \sum_{c \in \mathcal{C}_i} y_{ct} \quad \forall t \in \mathcal{T} \quad (2.1b)$$

$$y_{ct} \leq x_c \quad \forall c \in \mathcal{C} \forall t \in \mathcal{T} \quad (2.1c)$$

$$\sum_{c \in \mathcal{C}} y_{ct} \leq A_t \quad \forall t \in \mathcal{T} \quad (2.1d)$$

$$\sum_{c \in \mathcal{C}} x_c \leq C \quad (2.1e)$$

$$x_c \in \{0, 1\} \quad \forall c \in \mathcal{C} \quad (2.1f)$$

$$y_{ct} \in \{0, 1\} \quad \forall c \in \mathcal{C} \forall t \in \mathcal{T} \quad (2.1g)$$

$$z_{it} \in \{0, 1\} \quad \forall i \in \mathcal{I} \forall t \in \mathcal{T}. \quad (2.1h)$$

2.2 Model n.2: Ambulance location and relocation

Model 2 integrates the tactical and operational levels. In this problem, there is a single time slot and the parking areas have already been chosen. On the other hand, the number of available ambulances a varies between 1 and A and they must be relocated, depending on the value of a , to maximize coverage in all scenarios, under the constraint that only one ambulance can be relocated each time a varies. So, the locations of a and $a + 1$ ambulances must have a parking areas in common. The model can be easily generalized to allow for more relocations, but here we neglect this extension since it is unrealistic. The binary variables y_{ac} indicate whether an ambulance is assigned or not to parking area c when a ambulances are available; variables z_{ia} indicate whether demand point i is covered or not when a ambulances are available. The datum w_a represents the weight attributed to the configuration with a available ambulances and it depends on the fraction of time for which exactly a ambulances are available. This is estimated by a queuing theory model.

$$\text{maximize } v_2 = \sum_{i \in \mathcal{I}} \sum_{a=1}^A d_i w_a z_{ia} \quad (2.2a)$$

$$\text{s.t. } z_{ia} \leq \sum_{c \in \mathcal{C}_i} y_{ac} \quad \forall a = 1, \dots, A \quad (2.2b)$$

$$\sum_{c \in \mathcal{C}} y_{ac} = a \quad \forall a = 1, \dots, A \quad (2.2c)$$

$$y_{ac} \leq y_{a+1 c} \quad \forall c \in \mathcal{C} \quad \forall a = 1, \dots, A \quad (2.2d)$$

$$y_{ac} \in \{0, 1\} \quad \forall a = 0, \dots, A \quad \forall c \in \mathcal{C} \quad (2.2e)$$

$$z_{ia} \in \{0, 1\} \quad \forall i \in \mathcal{I} \quad \forall a = 1, \dots, A. \quad (2.2f)$$

We also considered a third model, resulting from the fusion of the two models above, where all three decision levels are integrated. In such a model we have variables y_{cta} , indicating whether an ambulance is assigned to parking area c in time slot t when there are a ambulances available, and z_{ita} , indicating whether demand point i is covered in time slot t when a ambulances are available.

3. Computational results

We have performed some preliminary experiments with the models introduced above on real-world instances referring to the city of Milan. Five time slots have been identified, based on the profiles of the phone calls and the ambulance speed along the day, as recorded in the database of the emergency health care system. The demand points ($|\mathcal{I}| \approx 12\,000$) have been located on the street graph of Milan and each one is associated with a demand d_{it} in each time slot, derived from the historical data. Smaller instances have been produced by selecting only 5000 or 8000

demand points from the larger instances. The number of ambulances ranges from 20 to 30 to 40, corresponding approximately to the current availability of public and private ambulances offering the service in Milan. Correspondingly, the number of parking areas to be built has been fixed to twice the number of ambulances (hence, 40, 60, or 80), chosen among 100, 150 or 200 candidate sites, according to the number of demand points. These sites have been located by maximizing the total distance between each other with a Maximum Diversity algorithm [1]. We consider instances with a single time slot, 3 time slots or 5 time slots. This produces $3 \cdot 3 \cdot 3 = 27$ different instances.

All instances have been submitted to CPLEX 11.0 with a time limit of one hour. It could solve most problem instances with 5000 demand points and some with 8000 demand points, sometimes at the root node, sometimes after branching. For the instances not solved to optimality it could achieve a very small gap between the upper and the lower bounds (from about 5% to less than 1%). When confronted with instances with 12442 demand points CPLEX could solve the linear relaxation at the root node only with the barrier method and it could not solve any instance to proven optimality. This suggests the development of alternative ad hoc approaches, such as Lagrangian relaxation, which is the subject of ongoing research.

References

- [1] Aringhieri R., Cordone R., Melzani Y., *Tabu Search vs. GRASP for the Maximum Diversity Problem*, 4OR: A Quarterly Journal of Operations Research 6:1 (2008) 45–60
- [2] Church R.L., ReVelle C.S., *The maximal covering location problem*, Papers of the Regional Sciences Association 32 (1974) 101-118
- [3] Gendreau M., Laporte G., Semet F., *The maximal expected coverage relocation problem for emergency vehicles* Journal of the Operational Research Society 57 (2006) 22-28

Coloring II

Lecture Hall B

Wed 3, 10:30–12:30

A branch-and-price approach for the Partition Coloring Problem

E. A. Hoshino,^a Y. Frota,^b C.C. de Souza^b

^aUniversity of Mato Grosso do Sul, Department of Computing and Statistics, Campo Grande MS, Brazil

^bUniversity of Campinas, Institute of Computing, Campinas SP, Brazil

Key words: partition coloring, formulation by representatives, branch-and-price

Introduction. Let $G = (V, E)$ be a undirected graph, where V is the set of vertices, E is the set of edges, and $Q = (V_1, \dots, V_q)$ is a partition of V into q disjoint sets. The **Partition Coloring Problem** (PCP) consists of finding a subset V' of V with exactly one vertex from each set $V_k \in Q$ and such that the chromatic number of the graph induced in G by V' is minimum.

This problem was first introduced in [5], motivated by the routing and wave-length assignment problem in optical networks. The authors proved that PCP is NP-hard and proposed heuristics extending classical methods for the vertex coloring problem (VCP). Different integer programming formulations were proposed to model the VCP. Mehrotra and Trick developed a column generation algorithm based on the classical independent set (IS) formulation [6]. Campêlo et. al. [2] proposed an alternative formulation in which a *representative* vertex is chosen to identify each color. Later, Campêlo et. al. [1] unified both formulations into a cut and price method to handle the VCP.

In this work, we explore this same idea of [1] to introduce a new integer programming (IP) formulation for the PCP. To the best of our knowledge, the only exact algorithm available for the PCP to date is the *branch-and-cut* approach based on the *representatives* formulation presented in [3]. An experimental comparison between these algorithms is also carried out here.

IP formulation. Let P be the set of all independent sets of G . Each independent set p in P can be represented by the characteristic vector a^p , where $a_i^p = 1$ if and

¹ First and second authors are supported by scholarships from Capes (Brazilian Ministry of Education). Third author is partially supported by CNPq – Conselho Nacional de Desenvolvimento Científico e Tecnológico – Grants # 301732/2007-8 and # 472504/2007-0.

only if the vertex i belongs to p . We now associate a binary variable λ_p to each independent set p in P . The formulation of the PCP that combines the IS and the *representatives* formulations is described below:

$$\text{(PCPr)} \quad \min \sum_{p \in P} \lambda_p \quad (0.1)$$

$$\text{subject to} \quad \sum_{p \in P} r_i^p \lambda_p \leq 1, \forall i \in V \quad (0.2)$$

$$\sum_{p \in P} a_i^p \lambda_p \leq 1, \forall i \in V \quad (0.3)$$

$$\sum_{p \in P} \left(\sum_{i \in V_k} a_i^p \right) \lambda_p = 1, \forall V_k \in Q \quad (0.4)$$

$$\lambda_p \in \{0, 1\}, \forall p \in P \quad (0.5)$$

where $r_i^p = 1$ if, and only if i is the representative vertex of the independent set related to p . The objective function (0.1) counts the number of independent sets, i.e., the number of colors. Constraints (0.2) state that a vertex can represent at most one independent set. A constraint in (0.3) forbids a vertex to be in two or more independent sets simultaneously. Finally, constraints (0.4) enforce that, in each set $V_k \in Q$, precisely one vertex is colored.

Let π , μ and α be the dual variables related to constraints (0.2), (0.3) and (0.4), respectively, in the linear relaxation of the (PCPr). Thus, the reduced cost of an independent set p is $\bar{c}_p = 1 - \sum_{i \in V} (\mu_i + \alpha_{Q[i]}) a_i^p - \sum_{i \in V} \pi_i r_i^p$, where $Q[i]$ is the set of the partition that contains the vertex i , i.e., $i \in V_{Q[i]}$. Now, for every vertex i in V , let $P(i) = \{p \in P : i \text{ is the representative vertex of } p\}$. Clearly, these sets form a partition of P . Therefore, a solution to the pricing problem can be computed by solving independently $|V|$ subproblems of the form:

$$\text{(IS}(i)) \quad \pi_i + \max \sum_{j \in N(i)} (\mu_j + \alpha_{Q[j]}) x_j$$

$$\text{subject to} \quad x_u + x_v \leq 1, \quad \forall (u, v) \in \tilde{E} \text{ and } u, v \in N(i), \\ x \in \{0, 1\}^{|N(i)|},$$

where $\tilde{E} = \bigcup_{k=1}^q \{(u, v) \in V_k \times V_k\} \cup E$ and $N(i) = \{j \in V \setminus V_{Q[i]} : (i, j) \notin E\}$.

It is not difficult to prove that the relaxation of (PCPr) provides dual bounds which are at least as good as those given by the *representatives* model described in [3], even when the latter is amended with all the *external cuts* introduced in that paper. Because these inequalities were claimed to be the most effective ones in cutting plane approaches, we were not compelled to implement a *branch-cut-and-price* algorithm. Indeed, our results showed that a simpler *branch-and-price* is already quite efficient.

Empirical analysis. We developed a *branch-and-price* algorithm using the model presented above. The code was implemented in C++ and XPRESS (version 2008)

was used as the linear programming solver. We applied the same branching strategy and primal heuristic described in [3]. The dual bound of Lasdon [4] was computed at each node of the branch-and-bound tree. To this end, we included the constraint $\sum_{p \in P} \lambda_p \leq UB$ in the formulation (PCPr), where UB is the best known primal bound at that current node.

We experimented our algorithm with the same instance classes reported in [3]. The class `RAND` consists of 80 randomized instances while the class `NSF` contains 49 instances related to the routing and wavelength assignment problem. The tests were ran on a Pentium Core2 Quad 2.83 GHz with 8Gb of RAM. We established a limit of 1800 seconds on the running time for all experiments.

First, we analyzed four algorithms to solve the pricing problem: (1) `TRICK`, (2) `SOLVER`, (3) `ADAPTIVE`, and (4) `ADAPTIVE*`. The first one consists of the algorithm proposed in [6] by Mehrotra and Trick to compute the maximum weighted independent set. The second algorithm uses an ILP solver. The method `ADAPTIVE` decides between `TRICK` and `SOLVER` depending on the density of the input graph. Finally, `ADAPTIVE*` uses pricing heuristics to solve the pricing first and, if it fails, the `ADAPTIVE` algorithm is called. Table 9 shows the average performance in each case to solve the linear relaxation at the root node for the 40 instances in class `RAND`. Line `opt` exhibits the total number of instances solved at optimality. Consider now just the instances solved at optimality by the four algorithms. Line `fast` shows the number of instances that were solved faster for each algorithm while line `speed-up` presents the average speed-up rate defined as the ratio between the running times of the slower and the current method, respectively.

	SOLVER	TRICK	ADAPTIVE	ADAPTIVE*
<code>opt</code>	40	30	40	40
<code>fast</code>	0	25	0	5
<code>speed-up</code>	1.01	15.29	12.38	12.18

Table 9. Comparison among different algorithms to solve the pricing problem.

We can see from Table 9 that `TRICK`'s algorithm solved more instances faster than the others, but it could not solve all instances to optimality. The difficulty for this method is related to graphs with low density ($\leq 30\%$). Overall, the best performance was achieved by the `ADAPTIVE*` algorithm.

The performance of our *branch-and-price* algorithm with `ADAPTIVE*` pricing (BP) is now compared to that of the *branch-and-cut* (BC) from [3], whose code was made available to us. The results are summarized in Table 10. Columns `opt`, `fast`, and `speed-up` have the same meaning as in Table 9. Column `dual` represents the total of instances with better dual bound at the root node. It came as no surprise that the dual bounds from BP surpassed those from BC since, as mentioned earlier, the linear relaxation of the former is at least as good as the one used in the latter.

Inspecting column `opt`, we can see that BP solved more instances to optimality than BC. Moreover, the improvement in the speed-up rate using the BP code was far more impressive. However, BC presented a slight better performance for the NSF class.

instance	BC				BP			
	opt	fast	dual	speed-up	opt	fast	dual	speed-up
RAND	47	15	6	3.22	64	29	73	36.9
NSF	37	28	3	6.17	49	9	23	4.17

Table 10. Comparative between BC and BP.

Conclusions. We proposed a new formulation for the Partition Coloring Problem which combines the IS and the representatives models. A *branch-and-price* algorithm was developed to compute this model exactly. Experiments showed that our approach is highly competitive with a *branch-and-cut* algorithm proposed earlier, outperforming the latter for random graphs. The development of the *branch-and-price* algorithm is still on going and we plan to test the code on other classes which include practical instances.

References

- [1] M. Campêlo, V. Campos, R. Corrêa, and C. Rodrigues. On fractional and integral chromatic numbers of a graph via cutting and pricing. In *Proceedings of the Fifth ALIO/EURO Conference on Combinatorial Optimization* (2005) pp. 42–42.
- [2] M. Campêlo, R. Corrêa, and Y. Frota. Cliques, holes and the vertex coloring polytope. *Information Processing Letters* **89** (2004) pp. 159–164.
- [3] Y. Frota, N. Maculan, T. Noronha, and C.C. Ribeiro. A branch-and-cut algorithm for partition coloring. In *Proceedings of the International Network Optimization Conference* (2007).
- [4] L. Lasdon. *Optimization Theory for Large Systems*. MacMillan (1970).
- [5] G. Li and R. Simha. The partition coloring problem and its application to wavelength routing and assignment. in *Proceedings of the First Workshop on Optical Networks* (2000).
- [6] A. Mehrotra and M. A. Trick. A column generation approach for graph coloring. *INFORMS Journal on Computing* **8** (1996) pp. 344–354.

Two upper bounds on the Chromatic Number

María Soto,¹ André Rossi, Marc Sevaux

*Université de Bretagne-Sud
Lab-STICC, CNRS UMR 3192
Centre de Recherche B.P. 92116
F-56321 Lorient Cedex FRANCE*

Key words: Graph coloring, Chromatic number, Upper bounding scheme

1. Introduction

Processor cache memory management is a challenging issue as it deeply impact performances and power consumption of electronic devices. It has been shown that allocating data structures to memory for a given application (as MPEG encoding, filtering or any other signal processing application) can be modeled as a minimum k -weighted graph coloring problem, on the so-called conflict graph. The graph coloring problem plays an important role as a particular case of the minimum weighted graph coloring problem, and providing upper bounds on the minimum number of colors to be used is an important issue for addressing these memory allocation problems.

A coloring of graph $G = (X, U)$ is a function $F : X \rightarrow \mathbb{N}^*$; where each node in X is allocated an integer value that is called a color. A proper coloring satisfies $F(u) \neq F(v)$ for all $(u, v) \in U$ [2]. The *chromatic number* of G , denoted by $\chi(G)$, is the smallest number of colors involved in any proper coloring. Determining $\chi(G)$ for any graph G is a \mathcal{NP} -hard problem [1] however there are some well known particular cases: $\chi(G) = 1$ if and only if G is a totally disconnected graph, $\chi(G) = 2$ for any exactly bipartite graphs (including trees and forests) and $\chi(G) = |X|$ if G is complete.

In this paper, we focus on upper bounds for $\chi(G)$ for any simple undirected graph G . The following bounds on the chromatic number can be found in [1].

- $\chi(G) \leq \delta(G) + 1 = d$, where $\delta(G)$ is the highest degree on G .
- $\chi(G) \leq \lfloor \frac{1 + \sqrt{8m + 1}}{2} \rfloor = l$, where $n = |X|$ is the number of vertexes and $m = |U|$

¹ Corresponding author: maria.soto@univ-ubs.fr

is the number of edges.

This paper is organized as follows. The next section introduces two new upper bounds for the chromatic number, without making any assumption on the graph structure. The first bound ξ is based on the number of the edges and nodes, and is to be applied to any connected component of the graph. The second bound ζ is based on the degree of the nodes in the graph. Section three briefly sketches the results obtained on a large set of instances used for assessing the quality of these bounds. This section will be widely extended in the full paper version of this abstract. Section four provides some conclusions and directions for future work.

2. Two new upper bounds on the chromatic number

Lemma 2.1. The following inequality holds for any connected, simple, undirected graph.

$$\chi(G) \leq \left\lfloor \frac{3 + \sqrt{9 + 8(m - n)}}{2} \right\rfloor = \xi \quad (2.1)$$

Proof of Lemma 2.1. There exists at least one edge between any pair of colors [1]. Such an edge joins node u and node v with $F(u) \neq F(v)$. Since G is not directed, $(u, v) = (v, u)$, there are at least $\chi(G)(\chi(G) - 1)/2$ such edges. The minimum number of nodes connected by $\chi(G)(\chi(G) - 1)/2$ edges are the $\chi(G)$ nodes of a clique. Then, these $\chi(G)(\chi(G) - 1)/2$ edges must connect at least $\chi(G)$ nodes in X (because the structure that involves k nodes and $k(k - 1)/2$ edges is a k -clique, that is a complete partial graph of G). As G is connected, at least $n - \chi(G)$ other edges are required for connecting the other $n - \chi(G)$ nodes to the $\chi(G)$ nodes previously considered. Therefore m can be lower bounded as follows:

$$m \geq \frac{\chi(G)(\chi(G) - 1)}{2} + n - \chi(G)$$

This inequality leads to a second degree polynomial in the variable $\chi(G)$, and solving it leads to (2.1). \square

Lemma 2.2. The chromatic number of any non directed, simple graph has the following property:

$\chi(G) \leq \zeta$ where ζ is the greatest integer such that there exist at least ζ nodes in X which degree is greater than or equal to $\zeta - 1$.

The second bound is indirectly based on the degree of saturation[†] of nodes, $DS(v)$ and its proof relies on Theorem 2.3. The following notation are used in this paper.

- $C = \{1..χ(G)\}$ is the minimum set of colors used in any valid coloring.
- A valid (or proper) coloring using exactly $χ(G)$ colors is said to be a minimal coloring.
- The neighborhood of node v denoted $N(v)$ is the set of all nodes u such that (u, v) belongs to U .

Theorem 2.3. Let F be a minimal coloring of G . For all color k in C , there exists a node v colored with k , (i.e. $F(v) = k$), such that its degree of saturation is $χ(G) - 1$, i.e. $DS(v) = χ(G) - 1$.

Proof of Theorem 2.3. The theorem is shown by contradiction.

It is shown that for all k in C there exists a node v , colored with k , such that $DS(v) = χ(G) - 1$. To do so, it is assumed that there exists a color k such that all node v colored with k have a degree of saturation being strictly less than $χ(G) - 1$. Then, for all $v \in X$ such that $F(v) = k$, $\exists c \in C \setminus \{k\}$ such that there does not exist $u \in N(v) | F(u) = c$. Consequently, a new valid coloring can be by setting $F(v) = c$. This operation can be performed for any node colored with k , leading to a coloring that involves $χ(G) - 1$ colors, which is impossible by definition of the chromatic number. \square

Proof of Lemma 2.2. It can be deduced from Theorem 2.3 that there exist at least $χ(G)$ nodes in G which degree is at least $χ(G) - 1$. So, $\zeta = χ(G)$ is the smallest integer such that there exist ζ nodes which degree is at least $\zeta - 1$ and such that $χ(G) \leq \zeta$. Consequently $χ(G)$ is less than or equal to the greatest integer satisfying this condition. \square

3. Assessing the new bounds quality

The new bounds introduced in this abstract were tested with existing bounds on a large set of instances from literature. As a result, ζ and ξ have the best performances on chromatic number bounds because ζ reaches the best value on 95 % of the instances and so does ξ on the remaining 5 %. Furthermore, ζ is significantly better than the others bounds as its value is in average 48% lesser than the others ones. It can also be observed that there is no dominance relationship between ξ and d as d is better than ξ on 45 % of the instances where ξ is on average 44% lesser than d , and d is better than ξ on 55 % of the instances where d is on average only 34% lesser than ξ . Whereas, it has been proved that $\xi \leq l$, these bounds have per-

[†] The degree of saturation of a node $v \in X$ denoted $DS(v)$ is the number of different colors at the nodes adjacent to v [3], [2].

performances very closer in practice.

Due to a lack of space, extended results will be presented at the conference.

4. Conclusion

The two upper bounds on the chromatic number introduced in this paper appear to be significantly better than the previously known ones. They are of particular interest for microprocessor cache memory management as they enable to reduce the search space for non conflicting memory allocations. These new bounds are easily computable even for large graphs as ξ complexity is $\mathcal{O}(1)$ and ζ is $\mathcal{O}(m)$. However, there is room for improvement as the gap with the chromatic number remains quite large. Using more information on graphs topology appears to be a promising direction for future work.

References

- [1] Reinhard Diestel. *Graph Theory*, volume 98 of *Graduate Texts in Mathematics*. Springer-Verlag, Heidelberg, July 2005.
- [2] Klotz Walter, *Graph Coloring Algorithms*, Mathematik-Bericht 5 (2002),1-9, Tu Clausthal.
- [3] Brélaz, D.,*New methods to color the vertices of a graph*, Communications of the Assoc. of Comput. Machinery 22 (1979), 251-256

Minimum sum set coloring on some subclasses of block graphs¹

Flavia Bonomo,^a Guillermo Durán,^b Javier Marengo,^c
Mario Valencia-Pabon^d

^a*CONICET, Argentina and Departamento de Computación, FCEN, Universidad de Buenos Aires, Argentina*
fbonomo@dc.uba.ar

^b*CONICET, Argentina and Departamento de Matemática, FCEN, Universidad de Buenos Aires, Argentina and Departamento de Ingeniería Industrial, FCFyM, Universidad de Chile, Chile*
gduran@dm.uba.ar

^c*Departamento de Computación, FCEN, Universidad de Buenos Aires, Argentina and Instituto de Ciencias, Universidad Nacional de General Sarmiento, Argentina*
jmarengo@ungs.edu.ar

^d*LIPN, Université Paris-Nord, France*
valencia@lipn.univ-paris13.fr

Abstract

In this work we study the Minimum Sum Set Coloring Problem (MSSCP) which consists in assign a set of $\omega(v)$ positive integers to each vertex v of a graph so that the intersection of sets assigned to adjacent vertices is empty and the sum of the assigned set of numbers to each vertex of the graph is minimum. This problem generalizes the well known Minimum Sum Coloring Problem, which is solvable in polynomial time on block graphs. We study two versions of the MSSCP (preemptive and non-preemptive) on two subclasses of block graphs: trees and line graphs of trees. This allows us to show that both versions of the problem are NP-complete on block graphs. We also find polynomial-time algorithms for the MSSCP under certain conditions.

Key words: graph coloring, minimum sum coloring, set-coloring, block graphs

A *vertex coloring* of a graph is an assignment of colors (positive integers) to its vertices such that adjacent vertices receive different colors. The *sum* of the

¹ Partially supported by ANPCyT PICT-2007-00518 and 00533, UBACyT Grants X069 and X606 (Arg.), FONDECyT Grant 1080286 and Millennium Science Institute “Complex Engineering Systems” (Chile), and BQR-UPN-2008 Grant (France).

coloring is the sum of the colors assigned to the vertices. The *chromatic sum* $\Sigma(G)$ of a graph G is the smallest sum that can be achieved by any vertex coloring of G . In the *Minimum Sum Coloring Problem* (MSCP) we have to find a coloring of G with sum $\Sigma(G)$.

The MSCP was introduced by Kubicka [11]. The problem is motivated by applications in scheduling [1; 2; 8; 9] and VLSI design [15; 17]. The computational complexity of determining the vertex chromatic sum of a simple graph has been studied extensively since then. In [12] it is shown that the problem is NP-hard in general but solvable in polynomial-time for trees. The dynamic programming algorithm for trees can be extended to partial k -trees and block graphs [10]. Furthermore, the MSCP is NP-hard even when restricted to some classes of graphs for which finding the chromatic number is easy, such as bipartite or interval graphs [2; 17]. A number of approximability results for various classes of graphs were obtained in the last ten years [1; 6; 8; 9; 4].

In an analogous way, it has been defined the edge coloring version of the MSCP: the *Minimum Sum Edge Coloring Problem* (MSECP). The MSECP is NP-hard for bipartite graphs [7], even if the graph is also planar and has maximum degree 3 [13]. Furthermore, in [13] is also shown that the MSECP is NP-hard for 3-regular planar graphs and for partial 2-trees. For trees, the MSECP can be solved in polynomial time [7; 16; 18] by a dynamic programming algorithm that uses weighted bipartite matching as a subroutine (see also [10]). In [3] it has been shown that this problem is also polynomial-time solvable for multicycles (i.e., cycles with parallel edges). For general multigraphs, a 1.829-approximation algorithm for the MSECP is presented in [9]. For bipartite graphs there exist better approximation ratios: a 1.796-approximation algorithm is given in [8], and a 1.414-approximation algorithm is proposed recently in [5].

An interesting application of the MSECP is to model dedicated scheduling of biprocessor jobs. The vertices correspond to the processors and each edge $e = uv$ corresponds to a job that requires a time unit of simultaneous work on the two preassigned processors u and v . The colors correspond to the available time slots. A processor cannot work on two jobs at the same time, this corresponds to the requirement that a color can appear at most once on the edges incident to a vertex. The objective is to minimize the average time before a job is completed. When there can be $\omega(e)$ instances of the same job, it arises the notion of *set-coloring* of the corresponding conflict graph. Formally, given a simple graph $G = (V, E)$ and a demand function $\omega : V \rightarrow Z^+$, a *vertex set-coloring* of (G, ω) consists in assigning to each vertex $v \in V$ a set of $\omega(v)$ colors in such a way that adjacent vertices will be assigned disjoint sets of colors. Given a vertex set-coloring of a graph G with demand function ω , the *sum* of the set-coloring is the sum of the colors in the set assigned to each one of the vertices. The *chromatic set-sum* $\Sigma(G, \omega)$ of (G, ω) is the smallest sum that can be achieved by any proper set-coloring of (G, ω) . In the *Minimum Sum Set Coloring Problem* (MSSCP) we have to find a set-coloring of

(G, ω) with sum $\Sigma(G, \omega)$. Clearly, when $\omega(v) = 1$ for each vertex v of the graph, the MSSCP becomes the MSCP. The dedicated scheduling of biprocessor jobs with multiple instances can be modeled as a MSSCP on the line graph of the conflict graph. A similar problem where each job e requires $\omega(e)$ time units of dedicated biprocessors, thus leading to a different objective function, was studied in [14]. In this case, sometimes it is allowed that a job is interrupted and continue later : the set of colors assigned to a vertex does not have to be consecutive. This type of scheduling is called *preemptive* (assuming that preemptions can happen only at integer times). Otherwise, if the set of colors assigned to each vertex needs to be consecutive, then the scheduling is called *non-preemptive*. In our case, the non-preemptive case arises when each job requires a high cost setup on the processors, and thus the objective is to minimize the average time before a job is completed, within the solutions minimizing the setup costs. Therefore, we have two variants of the MSSCP : the preemptive (pMSSCP) and the non-preemptive (npMSSCP) one.

Let $G = (V, E)$ be a graph with a demand function $\omega : V \rightarrow Z^+$. Denote $n = |V|$, Δ the maximum degree of G , and $\omega_{\max} = \max_{v \in V} \omega(v)$. The family of block graphs includes as special cases trees and line graphs of trees. We found dynamic programming algorithms for pMSSCP and npMSSCP on trees and line graphs of trees, that run in polynomial time under certain assumptions, like bounded degree or demand, or considering $n\omega_{\max}$ as the size of the input. This last assumption makes sense specially in the preemptive case, where the output of the algorithm are the lists of $\omega(v)$ colors assigned to each vertex v , and they can be formed by non-consecutive numbers.

Theorem 0.1. The npMSSCP on trees can be solved in $O(n\Delta^2\omega_{\max}^2)$ time.

Theorem 0.2. The pMSSCP on trees can be solved in $O(n(\Delta\omega_{\max})^{2\omega_{\max}})$ time.

Theorem 0.3. The npMSSCP for line graphs of trees can be solved in $O(\Delta^3 n^{\Delta+2} \omega_{\max}^{\Delta+1})$ time.

As a counterpart, we showed the NP-completeness of the preemptive and non-preemptive MSSCP on trees and their line graphs, respectively.

Theorem 0.4. The pMSSCP on trees and the npMSSCP on line graphs of trees are NP-complete, even considering the sum of the demands as the size of the input graph.

These results show that the MSSCP is NP-hard on block graphs, both in the preemptive and non-preemptive case, and that the computational complexity of the MSCP and the MSSCP (resp. the pMSSCP and the npMSSCP) can be different for the same family of graphs.

References

- [1] A. Bar-Noy, M. Bellare, M. M. Halldórsson, H. Shachnai, and T. Tamir. On chromatic sums and distributed resource allocation. *Inf. Comput.*, 140(2):183–202, 1998.
- [2] A. Bar-Noy and G. Kortsarz. Minimum color sum of bipartite graphs. *J. Algorithms*, 28(2):339–365, 1998.
- [3] J. Cardinal, V. Ravelomanana, and M. Valencia-Pabon. Minimum Sum Edge Coloring of Multicycles. Manuscript (ext. abstract in *ENDM*, 30:39–44, 2008).
- [4] U. Feige, L. Lovász, and P. Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.
- [5] R. Gandhi and J. Mestre. Combinatorial algorithms for data migration to minimize average completion time. To appear in *Algorithmica*.
- [6] K. Giaro, R. Janczewski, M. Kubale, and M. Malafiejski. Approximation algorithm for the chromatic sum coloring of bipartite graphs. *Lect. Notes Comput. Sci.* 2462:135–145, 2002.
- [7] K. Giaro and M. Kubale. Edge-chromatic sum of trees and bounded cyclicity graphs. *Inf. Process. Lett.*, 75(1–2):65–69, 2000.
- [8] M. M. Halldórsson, G. Kortsarz, and H. Shachnai. Sum coloring interval and k -claw free graphs with application to scheduling dependent jobs. *Algorithmica*, 37(3):187–209, 2003.
- [9] M. M. Halldórsson, G. Kortsarz, and M. Sviridenko. Min Sum Edge Coloring in Multigraphs Via Configuration LP. *Lect. Notes Comput. Sci.* 5035:359–373, 2008.
- [10] K. Jansen. Complexity results for the optimum cost chromatic partition problem. *Lect. Notes Comput. Sci.* 1256:727–737, 1997.
- [11] E. Kubicka. The Chromatic Sum of a Graph. PhD thesis, Western Michigan University, 1989.
- [12] E. Kubicka and A. J. Schwenk. An introduction to chromatic sums. In *Proc. of ACM Computer Science Conf.*, p. 15–21. Springer, 1989.
- [13] D. Marx. Complexity results for minimum sum edge coloring. To appear in *Discrete Appl. Math.*
- [14] D. Marx. Minimum sum multicoloring on the edges of trees. *Theoret. Comput. Sci.*, 361(2-3):133–149, 2006.
- [15] S. Nicoloso, M. Sarrafzadeh, and X. Song. On the sum coloring problem on interval graphs. *Algorithmica*, 23(2):109–126, 1999.
- [16] M. Salavatipour. On sum coloring of graphs. *Discrete Appl. Math.*, 127(3):477–488, 2003.
- [17] T. Szkaliczki. Routing with minimum wire length in the dogleg-free manhattan model is NP-complete. *SIAM J. Comput.*, 29(1):274–287, 1999.
- [18] X. Zhou and T. Nishizeki. Algorithm for the cost edge-coloring of trees. *J. Comb. Optim.*, 8(1):97–108, 2004.

Acyclic and Star Colorings of Joins of Graphs and an Algorithm for Cographs

Andrew Lyons

*Computation Institute, University of Chicago and
Mathematics and Computer Science Division, Argonne National Laboratory*
lyonsam@gmail.com

Key words: acyclic coloring, star coloring, cographs

An *acyclic coloring* of a graph is a proper vertex coloring such that the subgraph induced by the union of any two color classes is a disjoint collection of trees. The more restricted notion of *star coloring* requires that the union of any two color classes induces a disjoint collection of stars. The acyclic and star chromatic numbers of a graph G are defined analogously to the chromatic number $\chi(G)$ and are denoted by $\chi_a(G)$ and $\chi_s(G)$, respectively. In this paper, we consider acyclic and star colorings of graphs that are decomposable with respect to the join operation, which builds a new graph from a collection of two or more disjoint graphs by adding all possible edges between them. In particular, we present a recursive formula for the acyclic chromatic number of joins of graphs and show that a similar formula holds for the star chromatic number. We also demonstrate the algorithmic implications of our results for the cographs, which have the unique property that they are recursively decomposable with respect to the join and disjoint union operations.

1. Introduction

Both acyclic and star colorings have applications in the field of combinatorial scientific computing, where they model two different schemes for the evaluation of sparse Hessian matrices. The general idea behind the use of coloring in computing derivative matrices is the identification of entities that are essentially independent and thus may be computed concurrently; see [5] for a survey.

A number of results exist for acyclic and star colorings of graphs formed by certain graph operations. Results have been obtained for Cartesian products of paths [4], trees [9], cycles [7], and complete graphs [8]. In Section 2, we describe

the acyclic and star chromatic numbers of graphs formed by the join operation. The *join* of a collection $\{G_i = (V_i, E_i)\}_{i \in \mathcal{I}}$ of pairwise disjoint graphs, denoted \oplus , is the graph $G = (V, E)$, where $V = \bigcup_{i \in \mathcal{I}} V_i$ and $E = \{ab \mid ab \in E_i, i \in \mathcal{I}\} \cup \{ab \mid a \in V_i, b \in V_j, i, j \in \mathcal{I}, i \neq j\}$. Here and throughout this paper, \mathcal{I} denotes a finite index set.

The problems of finding optimal acyclic and star colorings are both \mathcal{NP} -hard and remain so even for bipartite graphs [2; 1]. It was shown recently [6] that every coloring of a chordal graph is also an acyclic coloring. Since recognizing and optimally coloring chordal graphs can be done in linear time, this result immediately implies a linear time algorithm for the acyclic coloring problem on chordal graphs. A generalization of this result and other related results can be found in [10], where it is shown that the graphs for which every acyclic coloring is also a star coloring are exactly the cographs. In Section 3, we show that our results imply a linear time algorithm for finding optimal acyclic and star colorings of cographs.

2. Joins of graphs

In this section, we outline a proof of the following theorem.

Theorem 9. Let $\{G_i = (V_i, E_i)\}_{i \in \mathcal{I}}$ be a finite collection of graphs. Then

$$(i) \quad \chi_a \left(\bigoplus_{i \in \mathcal{I}} G_i \right) = \sum_{i \in \mathcal{I}} \chi_a(G_i) + \min_{j \in \mathcal{I}} \left\{ \sum_{i \in \mathcal{I}, i \neq j} (|V_i| - \chi_a(G_i)) \right\};$$

$$(ii) \quad \chi_s \left(\bigoplus_{i \in \mathcal{I}} G_i \right) = \sum_{i \in \mathcal{I}} \chi_s(G_i) + \min_{j \in \mathcal{I}} \left\{ \sum_{i \in \mathcal{I}, i \neq j} (|V_i| - \chi_s(G_i)) \right\}.$$

For ease of exposition, we will focus on the case where G is the join of exactly two graphs as in the following lemma. To see that these results generalize to joins of arbitrarily large collections of graphs, first observe that the join operation is commutative and associative; the result is then obtained by using induction on $|\mathcal{I}|$.

Lemma 4. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be graphs. Then

$$(i) \quad \chi_a(G_1 \oplus G_2) = \chi_a(G_1) + \chi_a(G_2) + \min \{|V_1| - \chi_a(G_1), |V_2| - \chi_a(G_2)\};$$

$$(ii) \quad \chi_s(G_1 \oplus G_2) = \chi_s(G_1) + \chi_s(G_2) + \min \{|V_1| - \chi_s(G_1), |V_2| - \chi_s(G_2)\}.$$

We now sketch the idea behind the proof of this lemma. Suppose we are given graphs G_1 and G_2 and we wish to find an optimal acyclic or star coloring of their join. Since every vertex in V_1 is adjacent to every vertex in V_2 , no color can occur in V_1 and V_2 simultaneously. Moreover, the desired coloring must also be valid for the subgraphs induced by each $V_i, i \in \{1, 2\}$, where the lower bound will be $\chi_a(G_i)$ or $\chi_s(G_i)$ depending on the type of coloring that is sought. The key observation is that

at least one V_i must be *saturated*, meaning that each vertex receives a unique color. It can be shown that G will otherwise contain a bichromatic cycle – a violation of the conditions of acyclic coloring. Furthermore, such a bichromatic cycle implies a bichromatic path on four vertices, which cannot occur in a star coloring. Thus, given disjoint optimal acyclic colorings of G_1 and G_2 , an optimal acyclic coloring of their join can be constructed by saturating the graph G_i that minimizes $|V_i| - \chi_a(G_i)$. It is easy to see that the same procedure can be used in the context of star coloring.

3. Cographs

In this section, we outline a linear time algorithm for finding optimal acyclic and star colorings of cographs. The algorithm works on the cotree — defined below — in a way that is typical for algorithms on cographs. We begin with some definitions. The *disjoint union* of a collection $\{G_i = (V_i, E_i)\}_{i \in \mathcal{I}}$ of pairwise disjoint graphs, denoted \cup , is the graph $G = (V, E)$, where $V = \cup_{i \in \mathcal{I}} V_i$ and $E = \cup_{i \in \mathcal{I}} E_i$. A graph $G = (V, E)$ is a *cograph* if and only if one of the following is true:

- (i) $|V| = 1$;
- (ii) there exists a collection $\{G_i\}_{i \in \mathcal{I}}$ of cographs such that $G = \bigcup_{i \in \mathcal{I}} G_i$;
- (iii) there exists a collection $\{G_i\}_{i \in \mathcal{I}}$ of cographs such that $G = \bigoplus_{i \in \mathcal{I}} G_i$.

Cographs can be recognized in linear time [3], where most recognition algorithms also produce a special decomposition structure when the input graph G is a cograph. We associate with a cograph G a tree T_G called a *cotree*, whose leaves correspond to the vertices of G and whose internal nodes are labeled either 0 or 1. The 0-nodes correspond to the disjoint union of their children, and the 1-nodes correspond to the join of their children.

As in Section 2, we describe the binary case, which can be appropriately generalized. The algorithm proceeds by traversing the cotree starting with the leaves, such that no node is visited before both of its children have been visited. We do the following when we visit a node $t \in T_G$ with children t_1 and t_2 . If t is a 0-node, we construct a coloring that uses $\chi_a(t) = \max\{\chi_a(t_1), \chi_a(t_2)\}$ colors in the obvious way. If t is a 1-node, we use the process described in Section 2 to construct a coloring that is optimal by Theorem 9. Since the algorithm produces an optimal acyclic coloring for every node in the cotree, the last step will produce an optimal acyclic coloring of G itself. Our final theorem follows from the fact that every acyclic coloring of a cograph is also a star coloring and vice versa.

Theorem 10. An optimal acyclic coloring of a cograph can be found in linear time. Furthermore, the obtained coloring is also an optimal star coloring.

Acknowledgments. This work was supported in part by U.S. Department of Energy, under Contract DE-AC02-06CH11357.

References

- [1] Thomas F. Coleman and Jin-Yi Cai. The cyclic coloring problem and estimation of sparse Hessian matrices. *SIAM J. Alg. Disc. Meth.*, 7:221–235, 1986.
- [2] Thomas F. Coleman and Jorge J. Moré. Estimation of sparse Hessian matrices and graph coloring problems. *Mathematical Programming*, 28(3):243–270, 1984.
- [3] Derek G. Corneil, Yehoshua Perl, and Lorna K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.
- [4] Guillaume Fertin, Emmanuel Godard, and André Raspaud. Acyclic and k -distance coloring of the grid. *Information Processing Letters*, 87(1):51–58, 2003.
- [5] Assefaw Hadish Gebremedhin, Fredrik Manne, and Alex Pothen. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review*, 47(4):629–705, 2005.
- [6] Assefaw Hadish Gebremedhin, Alex Pothen, Arijit Tarafdar, and Andrea Walther. Efficient computation of sparse Hessians using coloring and automatic differentiation. *INFORMS J. Computing*, 21(2), 2009.
- [7] Robert E. Jamison and Gretchen L. Matthews. Acyclic colorings of products of cycles. *Bulletin of the Institute of Combinatorics and its Applications*, 54:59–76, 2008.
- [8] Robert E. Jamison and Gretchen L. Matthews. On the acyclic chromatic number of hamming graphs. *Graph. Comb.*, 24(4):349–360, 2008.
- [9] Robert E. Jamison, Gretchen L. Matthews, and John Villalpando. Acyclic colorings of products of trees. *Information Processing Letters*, 99(1):7–12, 2006.
- [10] Andrew Lyons. Restricted coloring problems and forbidden induced subgraphs. Preprint ANL/MCS-P1611-0409, Mathematics and Computer Science Division, Argonne National Laboratory, April 2009.

Exact Algorithms

Lecture Hall A

Wed 3, 14:00–15:30

Exact exponential-time algorithms for finding bicliques in a graph

Henning Fernau,^a Serge Gaspers,^b Dieter Kratsch,^c
Mathieu Liedloff,^d Daniel Raible^a

^a*Universität Trier, FB 4—Abteilung Informatik, D-54286 Trier, Germany*
{fernau, raible}@uni-trier.de

^b*LIRMM – University of Montpellier 2, CNRS, 34392 Montpellier, France*
gaspers@lirmm.fr

^c*LITA, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France*
kratsch@univ-metz.fr

^d*LIFO, Université d'Orléans, 45067 Orléans Cedex 2, France*
liedloff@univ-orleans.fr

Key words: graph, exact exponential-time algorithm, NP-hard problem, biclique

1. Introduction

Throughout the paper all graphs $G = (V, E)$ are undirected and simple. An *induced biclique* of G is a complete bipartite induced subgraph of G . A *non-induced biclique* is a complete bipartite (not necessarily induced) subgraph of G . Equivalently, the pair (X, Y) of disjoint vertex subsets $X \subseteq V$ and $Y \subseteq V$ is a non-induced biclique of G if $\{x, y\} \in E$ for all $x \in X$ and $y \in Y$. If, additionally, X and Y are independent sets, then (X, Y) is also an induced biclique of G . Let the pair (X, Y) be an induced or non-induced biclique. We call it a (k_1, k_2) biclique if $|X| = k_1$ and $|Y| = k_2$. Its cardinality is $|X| + |Y|$.

The literature dealing with bicliques is rich and diverse. There are applications of bicliques (induced or non-induced on bipartite graphs or general graphs) in various different areas such as data mining, automata and language theory, artificial intelligence and biology, see e.g. [1]. Therefore bicliques and algorithmic problems about bicliques have been studied extensively.

Known results. Already in [3], the complexity of finding certain bicliques has been considered. For example, deciding whether a bipartite graph has a balanced biclique of size (at least) k is NP-complete ([GT24] in [3]). A maximum cardinality induced biclique can be computed in polynomial time on bipartite graphs [2],

whereas this problem is NP-complete for general graphs [7]. Another related problem that asks to compute a non-induced biclique with a maximum number of edges is also known to be NP-hard [6].

The above-mentioned NP-completeness of the balanced biclique problem on bipartite graphs implies the NP-completeness of the following two problems about the existence of induced and non-induced bicliques, respectively.

Induced (k_1, k_2) Biclique

Input: An undirected graph $G = (V, E)$, positive integers k_1 and k_2 .

Question: Does G have an induced (k_1, k_2) biclique (X, Y) ?

Non-Induced (k_1, k_2) Biclique

Input: An undirected graph $G = (V, E)$, positive integers k_1 and k_2 .

Question: Does G have a non-induced (k_1, k_2) biclique (X, Y) ?

There is a trivial $O^*(3^n)$ algorithm for finding and also for enumerating all induced and non-induced (k_1, k_2) bicliques, respectively. * It considers all partitions of the vertex set into X , Y and $V \setminus (X \cup Y)$ and verifies for each whether (X, Y) fulfils all conditions.

Our results. For generating all non-induced (k_1, k_2) bicliques, note that there is no hope in obtaining a faster algorithm than the above-described $O^*(3^n)$ algorithm, as a complete graph on n vertices has $3^n \text{poly}(n)$ non-induced $(\lfloor n/3 \rfloor, \lfloor n/3 \rfloor)$ bicliques. For solving the **Non-Induced (k_1, k_2) Biclique** problem, however, we give a polynomial-space $O(1.8899^n)$ algorithm and an exponential-space $O(1.8458^n)$ algorithm.

There is also an $O^*(3^{n/3})$ time algorithm to solve **Induced (k_1, k_2) Biclique**. This algorithm is based on enumerating all maximal induced bicliques of the graph with a polynomial delay algorithm and on the fact that an n -vertex graph has $O^*(3^{n/3})$ maximal induced bicliques [4].

2. Polynomial-space algorithms for finding a non-induced biclique

We start by describing two simple $O^*(2^n)$ time algorithms for the **Non-Induced (k_1, k_2) Biclique** problem. We will use these algorithms as subroutines in our third algorithm with running-time $O(1.8899^n)$ and polynomial space usage.

The first algorithm, NIB1, verifies all sets $X_c \subseteq V$ with $|X_c| = k_1$ as candidates for being the set X in the pair (X, Y) . It computes for each X_c the set

* Throughout the paper we write $f(n) = O^*(g(n))$ if $f(n) \leq p(n) \cdot g(n)$ for some polynomial $p(n)$.

$B(X_c) := \{v \in V \setminus X_c \mid \forall x \in X_c : v \in N(x)\}$. If $|B(X_c)| < k_2$ then the algorithm rejects the candidate X_c . Otherwise it picks an arbitrary set $Y_c \subseteq B(X_c)$ with $|Y_c| = k_2$, and clearly (X_c, Y_c) is a non-induced (k_1, k_2) biclique. The only exponential part is the enumeration step and thus the running-time is $O^*(2^n)$.

The second algorithm, NIB2, verifies all sets $U \subseteq V$ with $|U| = k_1 + k_2$ and checks for each whether there is a non-induced biclique (X, Y) such that $|X| = k_1$ and $|Y| = k_2$. This can be done in polynomial time by computing the connected components of the complement of $G[U]$. If s_1, s_2, \dots, s_t are the sizes of those components, then there is a non-induced biclique as described above iff there is an $I \subseteq \{1, 2, \dots, t\}$ such that $\sum_{i \in I} s_i = k_1$. Such a SUBSET SUM problem can be solved in time $O(nW)$ by dynamic programming, where $W = \max_i s_i$. The only exponential part is the enumeration step and thus the running-time is $O^*(2^n)$.

For the third algorithm, NIB3, suppose w.l.o.g. that $k_1 \leq k_2$. If $k_1 \leq \lfloor n/3 \rfloor$, then run NIB1, otherwise run NIB2. Thus, the running-time of NIB3 is at most $O^*\left(\binom{n}{n/3}\right) = O(1.8899^n)$ if NIB1 is executed and at most $O^*\left(\binom{n}{2n/3}\right) = O(1.8899^n)$ if NIB2 is executed.

Theorem 2.1. Algorithm NIB3 solves the **Non-Induced (k_1, k_2) Biclique** problem in time $O(1.8899^n)$ and polynomial space.

3. Exponential-space algorithm for finding a non-induced biclique

In this section we provide an exponential-space algorithm for the the **Non-Induced (k_1, k_2) Biclique** problem in time $O(1.8458^n)$. The algorithm relies on a preprocessing involving a dynamic programming approach. It is described by the forthcoming three steps called *Partitioning*, *Preprocessing* and *Computing*.

3.1 Description of the algorithm

Partitioning Step. Let α be a constant to be determined. Given the graph $G = (V, E)$, compute an arbitrary partition of the vertex set into two subsets L and R such that $|R| = \lceil \alpha n \rceil$ and $|L| = \lfloor (1 - \alpha)n \rfloor$.

Preprocessing Step. This step focuses on the vertices of R . For any two (not necessarily disjoint) subsets $X, Y \subseteq R$ and any two integers i and j , $0 \leq i, j \leq |R|$, we compute the value of the boolean **R-biclique** $[X, i, Y, j]$ which is true iff there exist two subsets $X' \subseteq X$ and $Y' \subseteq Y$ such that (X', Y') is a non-induced (i, j) biclique. To compute the values of **R-biclique**, the sets X, Y and the integers i, j are considered by increasing cardinality and order.

For any $X, Y \subseteq R$ and i, j , such that $0 \leq i, j \leq |R|$, $\mathbf{R}\text{-biclique}[X, 0, Y, 0]$ is clearly true. Obviously $\mathbf{R}\text{-biclique}[\emptyset, i, Y, j]$ is true iff $i = 0$ and $\mathbf{R}\text{-biclique}[X, i, \emptyset, j]$ is true iff $j = 0$. For any other value, $\mathbf{R}\text{-biclique}[X, i, Y, j]$ is true iff

$$\bigvee_{v \in X} \left(\mathbf{R}\text{-biclique}[X \setminus \{v\}, i, Y, j] \vee \mathbf{R}\text{-biclique}[X \setminus \{v\}, i - 1, N(v) \cap Y, j] \right) \vee \bigvee_{v \in Y} \left(\mathbf{R}\text{-biclique}[X, i, Y \setminus \{v\}, j] \vee \mathbf{R}\text{-biclique}[N(v) \cap X, i, Y \setminus \{v\}, j - 1] \right).$$

Computing step. If the graph admits a non-induced (k_1, k_2) biclique, then it is found during this final step. For every two disjoint subsets $X_L, Y_L \subseteq L$ for which (X_L, Y_L) is a non-induced biclique with $|X_L| \leq k_1, |Y_L| \leq k_2$, let $X'_R = \{v \in R : v \text{ is adjacent to every vertex of } Y_L\}$ and $Y'_R = \{v \in R : v \text{ is adjacent to every vertex of } X_L\}$; if $\mathbf{R}\text{-biclique}[X'_R, k_1 - |X_L|, Y'_R, k_2 - |Y_L|]$ is true then the graph has a non-induced (k_1, k_2) biclique and “Yes” is returned.

If the algorithm was not able to find any X_L, Y_L such that $\mathbf{R}\text{-biclique}[X'_R, k_1 - |X_L|, Y'_R, k_2 - |Y_L|]$ is true, then the graph has no non-induced (k_1, k_2) biclique and it returns “No”. The correctness of the algorithm is shown in the next section. Note that instead of returning “Yes” or “No”, our algorithm can easily be modified (by standard backtracking techniques) to indeed return a non-induced (k_1, k_2) biclique if one exists.

3.2 Correctness of the algorithm

Assume that G has a non-induced (k_1, k_2) biclique and let (X, Y) be such a biclique. Since (L, R) is a partition of V , it holds that $X = X_L \cup X_R$ and $Y = Y_L \cup Y_R$ where $X_L = X \cap L, X_R = X \cap R, Y_L = Y \cap L$ and $Y_R = Y \cap R$. Since (X, Y) is a biclique, note that $X_R \subseteq X'_R$ and $Y_R \subseteq Y'_R$ where $X'_R = \{v \in R : v \text{ is adjacent to every vertex of } Y_L\}$ and $Y'_R = \{v \in R : v \text{ is adjacent to every vertex of } X_L\}$. Moreover $|X_R| = k_1 - |X_L|$ and $|Y_R| = k_2 - |Y_L|$.

Thus, assuming that X_L and Y_L are given, by definition of $\mathbf{R}\text{-biclique}$ it is sufficient to know whether $\mathbf{R}\text{-biclique}[X'_R, k_1 - |X_L|, Y'_R, k_2 - |Y_L|]$ is true. Since the Computing step goes through all possible choices for X_L and Y_L , it remains to show that the formula of the Preprocessing step is correct. Clearly, the base cases are correct. Let us consider the inductive step. The value $\mathbf{R}\text{-biclique}[X, i, Y, j]$ is true iff there exists a vertex $v \in X$ (the same argument can be used if v is a vertex of Y) such that $\mathbf{R}\text{-biclique}[X \setminus \{v\}, i, Y, j]$ is true (i.e. v does not belong to the (i, j) -biclique and thus it is removed from X) or $\mathbf{R}\text{-biclique}[X \setminus \{v\}, i - 1, N(v) \cap Y, j]$ is true (i.e. v is a vertex of the (i, j) -biclique and thus it remains to find a $(i - 1, j)$ -biclique from $X \setminus \{v\}$ and $\{u \in Y \setminus \{v\} : \{u, v\} \in E\}$).

3.3 Analysis of the running-time

The Partitioning step can clearly be done in polynomial time. During the Pre-processing step we need to compute $\text{R-biclique}[X, i, Y, j]$ for any (not necessarily disjoint) subsets $X, Y \subseteq R$ and any integers i and j , $0 \leq i, j \leq |R|$. For each such 4-tuple, R-biclique can be evaluated in polynomial time: go through all vertices of $X \cup Y$ and use the previously computed values of R-biclique – recall that X and Y are considered by increasing cardinality. Thus, to enumerate all X and Y , the algorithm needs $O^*(4^{\alpha n})$ time. The Computing step needs to consider all disjoint subsets $X_L, Y_L \subseteq L$ and then to look for already computed values of R-biclique . Consequently, it needs $O^*(3^{(1-\alpha)n})$ time. Finally the value of α is chosen to balance the running-time of the last two steps. By setting $\alpha = \log(3)/(2 + \log(3)) \approx 0.44211\dots$, our main theorem follows.

Theorem 3.1. The described algorithm solves the **Non-Induced (k_1, k_2) Biclique** problem in time $O(1.8458^n)$ and exponential space.

References

- [1] J. Amilhastre, M.C. Vilarem, P. Janssen, Complexity of minimum biclique cover and minimum biclique decomposition for bipartite dominofree graphs. *Discrete Appl. Math.* 86, pp. 125–144 (1998).
- [2] M. Dawande, J. Swaminathan, P. Keskinocak, S. Tayur, On bipartite and multipartite clique problems. *J. Algorithms* 41, pp. 388–403 (2001).
- [3] M.R. Garey, D.S. Johnson, *Computers and Intractability: A guide to the Theory of NP-completeness*. Freeman, New York, 1979.
- [4] S. Gaspers, D. Kratsch, M. Liedloff, On Independent Sets and Bicliques in Graphs, *Proceedings of WG 2008*, Springer, LNCS 5344, pp. 171-182.
- [5] D.S. Hochbaum, Approximating clique and biclique problems, *J. Algorithms* 29, pp. 174–200 (1998).
- [6] R. Peeters, The maximum edge biclique problem is NP-complete, *Discrete Appl. Math.* 131, pp. 651–654 (2003).
- [7] M. Yannakakis, Node and edge deletion NP-complete problems, *Proceedings of STOC 78*, ACM, pp. 253-264 (1978).

An Exact Algorithm to Minimize the Makespan in Project Scheduling with Scarce Resources and Feeding Precedence Relations

Lucio Bianco,^a Massimiliano Caramia^a

^a*Dipartimento di Ingegneria dell'Impresa, Università di Roma "Tor Vergata", Via del Politecnico, 1 - 00133 Roma, Italy*
{bianco,caramia}@disp.uniroma2.it

Key words: Branch and bound, Feeding precedences, Lagrangian relaxation

1. Introduction

In this paper we study an extension of the classical Resource-Constrained Project Scheduling Problem (RCPSP) with minimum makespan objective by introducing a further type of precedence constraints denoted as “Feeding Precedences” (FP). This problem happens in that production planning environment, like make-to-order manufacturing, which commonly requires the so-called project-oriented approach. In this approach a project consists of tasks, each one representing a manufacturing process, that is an aggregate activity. Due to the physical characteristics of these processes the effort associated with a certain activity for its execution can vary over time. An example is that of the human resources that can be shared among different simultaneous activities in proportion variable over time. In this case the amount of work per time unit devoted to each activity, so as its duration, are not univocally defined. This kind of problems is in general modelled by means of the so called Variable Intensity formulation, that is a variant of the Resource Constrained Project Scheduling Problem (see, e.g., Kis, 2006). As the durations of the activities cannot be taken into play, the traditional finish-to-start precedence relations, so as the generalized precedence relations, cannot be used any longer, and we need to introduce the so called “feeding precedences” (see, e.g., Kis, 2005, 2006). Feeding precedences are of four types:

- *Start-to-%Completed (S%C) between two activities (i, j) .* This constraint imposes that the processed percentage of activity j successor of i can be greater than $0 \leq g_{ij} \leq 1$ only if the execution of i has already started.
- *%Completed-to-Start (%CS) between two activities (i, j) .* This constraint is used to impose that activity j successor of i can be executed only if i has been processed for at least a fractional amount $0 \leq q_{ij} \leq 1$.

- *Finish-to-%Completed (F% C) constraints between two activities (i, j) .* This constraint imposes that the processed fraction of activity j successor of i can be greater than $0 \leq g_{ij} \leq 1$ only if the execution of i has been completed.
- *%Completed-to-Finish (% CF) constraints between two activities (i, j) .* This constraint imposes that the execution of activity j successor of i can be completed only if the fraction of i processed is at least $0 \leq q_{ij} \leq 1$.

In the following we propose a new mathematical formulation of the RCPSP with FP constraints in terms of mixed integer programming. For this formulation a branch and bound algorithm has been designed and a computational experimentation on randomly generated instances will be provided.

2. The Mathematical Model

We will assume that the planning horizon within which all the production processes have to be scheduled is $[0, T)$, where T is the project deadline, and it is discretized (without loss of generality) into T unit-width time periods $[0, 1), [1, 2), \dots, [T-1, T)$. Let us define with

- q_{ij}^1 , the fraction of activity i that has to be at least completed in order to let activity j start;
- q_{ij}^2 , the fraction of activity i that has to be at least completed in order to let activity j finish;
- g_{ij}^1 , the fraction of j that can be at most completed before the starting time of activity i ;
- g_{ij}^2 , the fraction of j that can be at most completed before the finishing time of activity i ;
- A , the set of activities to be carried out;
- A_1, A_2, A_3 and A_4 , the sets of pairs of activities for which a $S\%C$, $\%CS$, $F\%C$, and $\%CF$ constraint exists, respectively;
- K , the set of renewable resources each one available in an amount of b_k units, with $k = 1, \dots, K$;
- q_{ik} , the amount of units of resource k necessary to carry out activity i .

Furthermore, let us consider the following decision variables

- x_{it} , the percentage of i executed till time period t .
- s_{it}, f_{it} , binary variables that assumes value 1 if activity i has started or finished in a time period $\tau \leq t$, respectively, and assumes value 0 otherwise.

Since the completion time of an activity $i \in A$ can be expressed as $f_i = (T - \sum_{t=1}^T f_{it} + 1)$, the objective function can be written as:

$$\min \{ \max_{i \in A} f_i \} = \min \left\{ \max_{i \in A} \left(T - \sum_{t=1}^T f_{it} + 1 \right) \right\}$$

that can be easily linearized. The FP relations can be modelled as follows

$$x_{jt} \leq s_{i,t-1} + g_{ij}^1 \quad \forall (i, j) \in A_1, t = 1, \dots, T \quad (1)$$

$$s_{jt} \leq x_{i,t-1} + (1 - q_{ij}^1) \quad \forall (i, j) \in A_2, t = 1, \dots, T \quad (2)$$

$$x_{jt} \leq f_{i,t-1} + g_{ij}^2 \quad \forall (i, j) \in A_3, t = 1, \dots, T \quad (3)$$

$$f_{jt} \leq x_{i,t-1} + (1 - q_{ij}^2) \quad \forall (i, j) \in A_4, t = 1, \dots, T \quad (4)$$

$$x_{it} \leq x_{i,t+1} \quad \forall i \in A, t = 1, \dots, T - 1 \quad (5)$$

$$s_{it} \leq s_{i,t+1} \quad \forall i \in A, t = 1, \dots, T - 1 \quad (6)$$

$$f_{it} \leq f_{i,t+1} \quad \forall i \in A, t = 1, \dots, T - 1 \quad (7)$$

$$s_{iT} = f_{iT} = x_{iT} = 1 \quad \forall i \in A \quad (8)$$

$$s_{i0} = f_{i0} = x_{i0} = 0 \quad \forall i \in A \quad (9)$$

$$f_{it} \leq x_{it} \leq s_{it} \quad \forall i \in A, t = 1, \dots, T \quad (10)$$

$$\sum_{i=1}^{|A|} q_{ik}(x_{it} - x_{i,t-1}) \leq b_k \quad k = 1, \dots, K, t = 1, \dots, T \quad (11)$$

$$s_{it}, f_{it} \in \{0, 1\} \quad \forall i \in A, t = 1, \dots, T \quad (12)$$

$$x_{it} \geq 0 \quad \forall i \in A, t = 1, \dots, T \quad (13)$$

Constraints from (1) to (4) model $S\%C$, $\%CS$, $F\%C$ and $\%CF$ feeding constraints, respectively. Constraints (5) regulate the total amount processed of an activity $i \in A$ over time. Constraints (6) imply that if an activity $i \in A$ is started at time t , then variable $s_{i\tau} = 1$ for every $\tau \geq t$, and, on the contrary, if activity i is not started at time t , $s_{i\tau} = 0$ for every $\tau \leq t$. Constraints (7) are the same as constraints (6) when finishing times are concerned. Constraints (8) say that every activity $i \in A$ must start and finish within the planning horizon. Constraints (9) represent initialization conditions for variable s_{it}, f_{it}, x_{it} when $t = 0$. Constraints (10) force x_{it} to be zero if $s_{it} = 0$, and f_{it} to be zero if $x_{it} < 1$. Resource constraints are represented by relations (11). Constraints (12) and (13) limit the range of variability of the variables.

3. The Exact Algorithm Scheme

The exact algorithm proposed exploits the mathematical formulation presented in the previous section and is based on branch and bound rules. The root node α_0 of the search tree is associated with the whole problem P_0 and two bounds, i.e., the trivial upper bound on the minimum makespan given by the time horizon T and the lower bound on the minimum makespan based on a Lagrangian relaxation of the resource constraints of the mathematical model (see, Bianco and Caramia, 2009). Each level of the search tree is associated with an activity in the set A of

activities, which means that the tree has at most $|A|$ levels. Assume that activity i is associated with the first level of the tree; then level 1 is formed by T subproblems, denoted P_{1t} with $t = 1, \dots, T$, each associated with a time slot t in the time horizon at which activity i can start at the very latest, i.e., at each node α_{1t} at level 1 of the tree the subproblem P_{1t} associated is obtained from P_0 (its parent) by imposing $s_{it} = 1$. The same analysis described for level 1 can be applied for every level of the tree, i.e., from a subproblem $P_{i\tau}$ at level i we can generate T subproblems $P_{i+1,t}$ at level $i + 1$, with $t = 1, \dots, T$, each obtained from $P_{i\tau}$ by fixing $s_{jt} = 1$, where j is the activity associated with level $i + 1$. Each subproblem P_{it} undergoes a bounding phase in which a lower bound on the minimum makespan is computed as done for the root node. Here we can have three alternative outcomes: (1) the mathematical program associated with the Lagrangian lower bound is empty, i.e., some feeding precedence relations cannot be obeyed, (2) the solution of the Lagrangian relaxation is not feasible with respect to some resource constraints, (3) the latter solution respects all the resource constraints. Clearly, in the first case the subtree generating from problem P_{it} is fathomed since a feasible solution cannot be found, with a consequent backtracking to the previous level; in the case (2) the Lagrangian solution value $La(P_{it})$ is a lower bound for subproblem P_{it} and it is compared with the best upper bound UB^* found so far; if $La(P_{it}) \geq UB^*$ then the tree is pruned again (with a consequent backtracking) otherwise the search is continued in a depth first search strategy. In the last occurrence, i.e., in the case (3), the solution value $f(P_{it})$, obtained by substituting x_{it}, s_{it}, f_{it} values obtained by the Lagrangian relaxation in P_{it} , is an upper bound for the latter subproblem and therefore it is an upper bound on the whole problem P_0 . Now, if this solution to P_{it} satisfies the complementary slackness conditions, it is the optimal solution for P_{it} and the tree can be pruned, possibly updating UB^* to $f(P_{it})$ if the former is greater than the latter. If the solution is not optimal for P_{it} then the search continues in a depth first search strategy possibly updating UB^* to $f(P_{it})$ if $UB^* > f(P_{it})$.

4. Preliminary Computational Results

The implementation of our algorithm has been carried out in the C language. The performance of our approach has been compared to that of the commercial solver CPLEX, implementing the mathematical formulation presented in Section 2 in the AMPL language, version 8.0.0. The machine used for the experiments is a PC Core Duo with a 1.6 GHz Intel Centrino Processor and 1 GB RAM. Experiments have been generated with the following features:

- the number of activities $|A|$ has been chosen equal to 10, 20, and 30;
- a density fd of feeding precedences has been set equal to 30%;
- the number K of renewable resources has been kept equal to 4;
- an amount b_k of resource availability per period for each resource $k = 1, \dots, K$ has been set equal to 4;

- a request q_{ik} of resource $k = 1, \dots, K$ for every activity $i \in A$ has been assigned uniformly at random from 1 to 3;
- the values $g_{ij}^1, q_{ij}^1, g_{ij}^2, q_{ij}^2$ have been assigned uniformly at random in the range $(0.00, 1.00)$;
- the time horizon T , that is the starting upper bound at the root node of the search tree, has been fixed to 80.

Preliminary results, given as averages over five instances, are shown in Table 11 (an extensive experimentation is in progress). We listed the following values: OPT_CPLEX and OPT_BB , being the average values of the makespan, computed over the instances solved at the optimum, achieved by CPLEX and by our algorithm, respectively; $\#_Opt_CPLEX$ and $\#_Opt_BB$, being the number of instances, out of the five, solved at the optimum by CPLEX and by our algorithm, respectively; CPU_CPLEX and CPU_BB , being the average CPU time (in seconds) elapsed by CPLEX and by our algorithm to find the optimal solutions, respectively.

The comparison shows that our algorithm is able to solve all the instances with sizes from 10 to 30 activities, differently from CPLEX that for 30 activities solved only two out of the five instances within the time limit of two hours. Moreover, CPU_BB is always lower than CPU_CPLEX .

$ A $	Opt_CPLEX	$\#_Opt_CPLEX$	CPU_CPLEX	OPT_BB	$\#_Opt_BB$	CPU_BB
10	5.2	5/5	0.7	5.2	5/5	0.3
20	9.0	5/5	67.6	9.0	5/5	34.2
30	14.2	2/5	1826.6	15.8	5/5	1248.9

Table 11. Comparison between the performance of CPLEX and our algorithm

References

- [1] Bianco, L., M. Caramia. 2009. Minimizing the Completion Time of a Project Under Resource Constraints and Feeding Precedence Relations: a Lagrangian Relaxation Based Lower Bound, RR-03.09 - University of Rome “Tor Vergata”.
- [2] Kis, T. 2005. A branch-and-cut algorithm for scheduling of projects with variable-intensity activities, *Mathematical Programming* 103(3), pp. 515-539.
- [3] Kis, T. 2006. Rcps with variable intensity activities and feeding precedence constraints. In *Perspectives in Modern Project Scheduling*, Springer, pp. 105-129.

Exact Algorithms for Vehicle Routing Problems with Different Service Constraints

Rita Macedo ,^a Cludio Alves ,^a J. M. Valrio de Carvalho^a

^a*Centro de Investigacao Algoritmi da Universidade do Minho,
Escola de Engenharia, Universidade do Minho, 4710-057 Braga, Portugal
{claudio,rita,vc}@dps.uminho.pt*

Key words: Vehicle Routing Problem, Branch-and-price, Cutting planes

1. Introduction

In this paper we analyze a routing problem related with the waste collection in urban areas, which is formulated as a Vehicle Routing Problem with additional constraints. There is a single depot, which is the beginning, o , and the end, d , of all vehicle routes. The fleet of vehicles is homogeneous, with a capacity of W units. It is assumed that there are K available vehicles in the fleet. A vehicle that leaves the depot must fulfill a minimum filling constraint by coming back with at least L_{min} units of waste. Each waste collection point is seen as a client $i \in N = \{1, \dots, m\}$, with a given load, l_i , that is stored in a container. A container is considered to be full if it has more than a given amount of waste w_{max} . It is only mandatory that the waste of a given client i is collected if its container is full. Clients that do not have full containers will only be visited if necessary, in order to ensure that a pre-defined minimal amount of waste L_{min}^T is collected. The first set of clients is denoted by N_1 and the second by N_2 . We use an exact branch-and-price-and-cut algorithm to solve the integer problem. The column generation model is a Dantzig-Wolfe decomposition of a network flow model over arcs, whose variables are also used in our branching scheme. We apply dynamic stabilization techniques to the column generation algorithm and use dual-feasible functions to derive valid cutting planes from implicit constraints of the model. An extensive survey on time constrained routing and scheduling problems can be found in [1].

2. Mathematical Model

Our routing problem is defined in a graph $G = (V, A)$, where $V = N \cup \{o, d\}$ represents the set of nodes in the graph and A the set of oriented arcs. The graph is assumed to be unique and complete. Each arc $(i, j) \in A$ has a cost c_{ij} , that represents the distance between i and j , as well as other eventual costs that may be incurred by traveling through it. The optimization objective of the plan is to minimize the total cost of the vehicles routes.

2.1 Column Generation Model

The column generation model is a network flow model over paths that results from a Dantzig-Wolfe decomposition of a network flow model over arcs. The reformulated model consists of a master problem (2.1)-(2.6) with binary variables, λ_p , that represent feasible vehicle routes, and a pricing subproblem that consists of a shortest path problem with additional constraints. The column generation model is stronger than the original arc-flow formulation and does not have symmetry. Let Ω denote the set of all feasible routes, c_p the cost of a path $p \in \Omega$ and l_p the total amount of waste picked up in route p . Constraints (2.2) and (2.3) guarantee, respectively, that clients that belong to set N_1 are visited exactly once, and that the other clients have, at most, one visit. Constraint (2.4) ensures that the number of used routes does not exceed the number of available vehicles. The minimum filling constraint is guaranteed in (2.5). Model (2.1)-(2.6) can be seen as a set-partitioning model with additional constraints.

$$\min \sum_{p \in \Omega} c_p \lambda_p \quad (2.1)$$

s.t.

$$\sum_{p \in \Omega} a_{ip} \lambda_p = 1, \quad \forall i \in N_1, \quad (2.2)$$

$$\sum_{p \in \Omega} a_{ip} \lambda_p \leq 1, \quad \forall i \in N_2, \quad (2.3)$$

$$\sum_{p \in \Omega} \lambda_p \leq K, \quad (2.4)$$

$$\sum_{p \in \Omega} l_p \lambda_p \geq L_{min}^T, \quad (2.5)$$

$$\lambda_p \in \{0, 1\}, \quad \forall p \in \Omega. \quad (2.6)$$

Solving the pricing subproblem generates a new column to be added to the master problem. This column, which represents a valid path in the primal space, is the one with the lowest reduced cost. The reduced cost of a path $p \in \Omega$ is given by $c'_p = c_p - \sum_{i \in N_1 \cup N_2} a_{ip} \pi_i - l_p \delta - \theta$. In our implementation, we solved this subproblem using a dynamic programming algorithm. Recently, new cuts based on dual-feasible

functions have been described in the literature. In our routing problem, we can apply these principles by considering a valid constraint for (2.1)-(2.6). Being f_p the free space in a vehicle that goes through route p , we have that $\sum_{p \in \Omega} f_p \lambda_p \leq K \times W - L_{min}^T$.

2.2 Stabilization Strategies

The introduction of dual cuts [2; 3] is one of the most promising methods proposed in the literature to accelerate the convergence of column generation algorithms. We derived a new cut, valid in the dual space, which represents the possibility of exchanging clients in a route.

Proposition 2.1. Given a client $i \in N$, let S be a subset of clients such that $|S| > 1$ and $\sum_{j \in S} l_j = l_i$, and P be a path between all the clients of S . The cost of that path is denoted by c_S . Let c_{min}^1 and c_{min}^2 be the first and the second smaller cost of arcs incident in i , respectively. Let a and b be the nodes at the extremities of P , and d_{max}^1 and d_{max}^2 be the higher costs among the arcs incident in a and b , respectively. If P is a circuit, pick the higher arc costs incident in two nodes of S . Let π_n , $n \in N$, θ and δ , be the dual variables associated to the constraints (2.2)-(2.3), (2.4) and (2.5), respectively. The cut $-\pi_i + \sum_{j \in S} \pi_j \leq c_S - (c_{min}^{i1} + c_{min}^{i2}) + (d_{max}^{S1} + d_{max}^{S2})$ is valid in the dual space.

Proof. Let $(\bar{\pi}, \bar{\theta}, \bar{\delta})$ be the dual solution corresponding to an optimal solution to the column generation problem (2.1)-(2.6). This solution is, according to the optimality conditions, valid in the dual space, which means that $\sum_{n \in N} a_{np} \bar{\pi}_n + \bar{\theta} + l_p \bar{\delta} \leq c_p$, $\forall p \in \Omega$. Two cases may occur: (i) $i \in N_1$ or ($i \in N_2$ and client i is visited in the optimal solution) or (ii) $i \in N_2$ and client i is not visited in the optimal solution. Let us consider case (i). There is, in the optimal solution, at least one positive basic variable corresponding to a path $p' = (a_{1p'}, \dots, a_{mp'})$, with $a_{ip'} = 1$ and $a_{sp'} \in \{0, 1\}$, $\forall s \in S$. Its reduced cost is null, which means that $c_{p'} = \sum_{n \in N} a_{np'} \bar{\pi}_n + \bar{\theta} + l_{p'} \bar{\delta}$. Consider path $\tilde{p} = (a_{1\tilde{p}}, \dots, a_{m\tilde{p}})$, with $a_{i\tilde{p}} = a_{ip'} - 1 = 0$, $a_{j\tilde{p}} = a_{jp'} + 1$, $\forall j \in S$, and $a_{n\tilde{p}} = a_{np'}$, $\forall n \in N \setminus (S \cup \{i\})$. Path \tilde{p} is a valid path corresponding to the exchange of client i by clients $j \in S$. Clearly, $c_{\tilde{p}} - c_{p'} \leq c_S - (c_{min}^{i1} + c_{min}^{i2}) + (d_{max}^{S1} + d_{max}^{S2})$ and $l_{\tilde{p}} = l_{p'}$. Suppose that there is a cut that is not valid in the dual space, i.e., $-\bar{\pi}_i + \sum_{j \in S} \bar{\pi}_j > c_S - (c_{min}^{i1} + c_{min}^{i2}) + (d_{max}^{S1} + d_{max}^{S2})$. Then, $-\bar{\pi}_i + \sum_{j \in S} \bar{\pi}_j > c_{\tilde{p}} - c_{p'} \Rightarrow -\bar{\pi}_i + \sum_{j \in S} \bar{\pi}_j > c_{\tilde{p}} - (\sum_{n \in N} a_{np'} \bar{\pi}_n + \bar{\theta} + l_{p'} \bar{\delta})$. It is clear that $-\bar{\pi}_i + \sum_{j \in S} \bar{\pi}_j = \sum_{n \in N} a_{n\tilde{p}} \bar{\pi}_n - \sum_{n \in N} a_{np'} \bar{\pi}_n$, which implies that $\sum_{n \in N} a_{n\tilde{p}} \bar{\pi}_n + \bar{\theta} + l_{\tilde{p}} \bar{\delta} > c_{\tilde{p}}$, contradicting the validity of solution $(\bar{\pi}, \bar{\theta}, \bar{\delta})$. Let us now consider case (ii). Given that $\sum_{p \in \Omega} a_{ip} \lambda_p = 0$, by the complementary slackness theorem, $\bar{\pi}_i = 0$. Moreover, as $i \in N_2$ and $\sum_{j \in S} l_j = l_i$, we know that $j \in N_2, \forall j \in S$, and thus that $\sum_{j \in S} \bar{\pi}_j \leq 0$. Therefore, $-\bar{\pi}_i + \sum_{j \in S} \bar{\pi}_j \leq 0$. Let c_{ij_1} and c_{ij_2} be the costs of the arcs incident in i and j_1 , and i and j_2 , respectively, with $j_1, j_2 \in S$. We know that $c_{ij_1} + c_{ij_2} \geq c_{min}^1 + c_{min}^2$ and $c_{ij_1} + c_{ij_2} \leq d_{max}^1 + d_{max}^2$. This means

that $d_{max}^1 + d_{max}^2 \geq c_{min}^1 + c_{min}^2 \implies c_p - (c_{min}^1 + c_{min}^2) + (d_{max}^1 + d_{max}^2) \geq 0 \implies -\bar{\pi}_i + \sum_{j \in S} \bar{\pi}_j \leq c_p - (c_{min}^1 + c_{min}^2) + (d_{max}^1 + d_{max}^2)$. □

2.3 Branch-and-Bound

In the branching scheme, we used the binary decision variables of the network flow model over arcs, x_{ij} , being i and j the beginning and end of an arc. The resulting branching constraints, $x_{ij} = 1$ and $x_{ij} = 0$, representing two new branching nodes of the branching tree, are easily enforced in the master problem (2.1)-(2.6), whose variables are not used to branch as it would cause a regeneration of columns, unless the pricing subproblem was reformulated in a more complicated way.

3. Conclusions

In this paper, we defined an exact branch-and-price-and-cut algorithm for a routing problem arising in an urban waste management system. We conducted preliminary computational experiments on a set of random instances with promising results. For the sake of brevity, we do not present the list of results obtained, but we can say that for instances with 50 clients, we usually converge to a small optimality gap in a reasonable amount of time. In what concerns the cut described in §2.1, its performance depends on the values of parameters like K , L_{min} or L_{min}^T , whereas the dual cut described in §2.2 performs well when the clients are organized in geographically scattered groups.

References

- [1] J. Desrosiers, Y. Dumas, M. Solomon and F. Soumis. Time Constrained Routing and Scheduling. *in M.O. Ball et al., Eds., Handbooks in OR & MS*, Vol. 8, Elsevier Science B.V., 1995.
- [2] J.M. Valrio de Carvalho. Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, 17(2):175-182, 2005.
- [3] F. Clautiaux, C. Alves and J. M. Valrio de Carvalho. New stabilization procedures for the cutting stock problem. *In revision for INFORMS Journal on Computing*, 2007.

Networks I

Lecture Hall B

Wed 3, 14:00–15:30

Algorithmic Solutions of Discrete Control Problems on Stochastic Networks

Dmitrii Lozovanu,^a Stefan Pickl^b

^a*Institute of Mathematics and Computer Science, Academy of Sciences,
Academy str., 5, Chisinau, MD-2028, Moldova*
lozovanu@math.md

^b*Institut für Theoretische Informatik, Mathematik und Operations Research,
Fakultät für Informatik, Universität der Bundeswehr, München*
stefan.pickl@unibw.de

Abstract

Classical discrete optimal control problems which are based on a graph-theoretic structure are introduced. The paper focusses now on a stochastic extension of such processes. Based on the concept of general Markov processes, stochastic networks are characterized. Suitable algorithms exploiting the time-expanded network method are presented.

Key words: Stochastic Networks, Markov Processes, Time-Expanded Network

1. Introduction

Classical discrete optimal control problems are introduced in [1; 2; 4]. We consider now such control problems for which the discrete system in the control process may admit dynamical states where the vector of control parameters is changing in a random way. We call such states of the dynamical system *uncontrollable dynamical states*. So, we consider the control problems for which the dynamics may contain controllable states as well as uncontrollable ones. We show that these types of problems can be modeled on stochastic networks. New algorithmic approaches for their solving based on the concept of Markov processes and dynamic programming from [3] can be described. The approach is based on the time-expanded network method. This is a comfortable graph-theoretic structure which was introduced in [4; 5].

2. The General Graph-Theoretic Structure

We consider a time-discrete system L with a finite set of states $X \subset R^n$. At every time-step $t = 0, 1, 2, \dots$, the state of the system L is $x(t) \in X$. Two states

x_0 and x_f are given in X , where $x_0 = x(0)$ represents the starting state of system L and x_f is the state in which the system L must be brought, i.e. x_f is the final state of L . We assume that the system L should reach the final state x_f at the time-moment $T(x_f)$ such that $T_1 \leq T(x_f) \leq T_2$, where T_1 and T_2 are given. The dynamics of the system L is described as follows

$$x(t+1) = g_t(x(t), u(t)), \quad t = 0, 1, 2, \dots, \quad (2.1)$$

where

$$x(0) = x_0 \quad (2.2)$$

and $u(t) = (u_1(t), u_2(t), \dots, u_m(t)) \in R^m$ represents the vector of control parameters. For any time-step t and an arbitrary state $x(t) \in X$ a feasible finite set $U_t(x(t)) = \{u_{x(t)}^1, u_{x(t)}^2, \dots, u_{x(t)}^{k(x(t))}\}$, for the vector of control parameters $u(t)$ is given, i.e.

$$u(t) \in U_t(x(t)), \quad t = 0, 1, 2, \dots \quad (2.3)$$

We assume that in (2.1) the vector functions $g_t(x(t), u(t))$ are determined uniquely by $x(t)$ and $u(t)$, i.e. $x(t+1)$ is determined uniquely by $x(t)$ and $u(t)$ at every time-step t . Additionally, we assume that at each moment of time t the cost $c_t(x(t), x(t+1)) = c_t(x(t), g_t(x(t), u(t)))$ of system's passage from the state $x(t)$ to the state $x(t+1)$ is known. Let $x_0 = x(0), x(1), x(2), \dots, x(t), \dots$ be a trajectory generated by given vectors of control parameters $u(0), u(1), \dots, u(t-1), \dots$. Then either this trajectory passes through the state x_f at the time-moment $T(x_f)$ or it does not pass through x_f . We denote by

$$F_{x_0x_f}(u(t)) = \sum_{t=0}^{T(x_f)-1} c_t(x(t), g_t(x(t), u(t))) \quad (2.4)$$

the integral-time cost of system's passage from x_0 to x_f if $T_1 \leq T(x_f) \leq T_2$; otherwise we put $F_{x_0x_f}(u(t)) = \infty$. In [1; 2; 4] the following problem has been formulated: Determine vectors of control parameters $u(0), u(1), \dots, u(t), \dots$ which satisfy the conditions (2.1)-(2.3) and minimize the functional (2.4). This problem can be regarded as a control model with controllable states because for an arbitrary state $x(t)$ at every moment of time the determination of a vector of control parameter $u(t) \in U_t(x(t))$ is assumed to be at our disposition.

In this paper we assume that the dynamical system L may contain uncontrollable states, i.e. for the system L there exists dynamical states in which we are not able to control the dynamics of the system and the vector of control parameters $u(t) \in U_t(x(t))$ for such states is changing by a random way according to a given

distribution function

$$p : U_t(x(t)) \rightarrow [0, 1], \quad \sum_{i=1}^{k(x(t))} p(u_{x(t)}^i) = 1. \quad (2.5)$$

on the corresponding dynamical feasible sets $U_t(x(t))$. If an arbitrary dynamic state $x(t)$ of system L at given moment of time t is characterized by its position (x, t) then the set of positions $XT = \{(x, t) \mid x \in X, t \in \{0, 1, 2, \dots\}\}$ of the dynamical system can be divided into two disjoint subsets $XT = XT_C \cup XT_N$ ($XT_C \cap XT_N = \emptyset$), where XT_C represents the set of controllable positions of L and XT_N represents the set of positions $(x, t) = x(t)$ for which the distribution function (2.5) of the vectors of control parameters $u(t) \in U_t(x(t))$ are given. This means that the dynamical system L is expressed by the following behavior: If the starting point belongs to the set of the controllable positions then the *decision maker* choose a vector of these control parameters and we reach the state $x(1)$. If the starting state belongs to the set of uncontrollable positions then the system passes to the next state in a random way. After that step if at the time-moment $t = 1$ the state $x(1)$ belongs to the set of controllable positions then the decision maker may choose the vector of control parameter $u(t) \in U_t(x(t))$ and we obtain the state $x(2)$. If $x(1)$ belong to the set of uncontrollable positions then the system passes to the next state again in a random way and so on.

3. The Main Concept

In this dynamic process the final state may be reached at a given moment of time with a probability which depends on the control vectors (in the controllable states) as well as on the expectation of the integral time cost. Our main results are concerned with solving the following problems which are based on a graph-theoretic structure:

Algorithmic Procedure

1. For given vectors of control parameters $u^0(t) \in U_t(x(t))$, $x(t) \in XT_C$, determine the probability that the final state will be reached at the moment of time $T(x_2)$ such that $T_1 \leq T(x_f) \leq T_2$.

2. Find the vectors of control parameters $u^*(t) \in U_t(x(t))$, $x(t) \in XT_C$ for which the probability in problem 1 is maximal?

3. For given vectors of control parameters $u^0(t) \in U_t(x(t))$, $x(t) \in XT_C$, estimate the integral-time cost after T stages.

4. For given vectors of control parameters $u^0(t) \in U_t(x(t))$, $x(t) \in XT_C$, determine the integral-time cost of system's passage from the starting state x_0 to the final state x_f when the final state is reached at the time-moment $T(x_f)$ such that $T_1 \leq T(x_f) \leq T_2$.

5. Minimization problem I:

For which vectors of control parameters $u^*(t) \in U_t(x(t))$, $x(t) \in XT_C$ is the expectation of the integral-time cost in problem 3 minimal?

6. Minimization problem II:

For which vectors of control parameters $u^*(t) \in U_t(x(t))$, $x(t) \in XT_C$ is the expectation of the integral-time cost in problem 4 minimal?

Note that problems 1-6 extend and generalize the deterministic and stochastic dynamic problems from [1; 2; 3] using a certain graph-theoretic approach.

4. Control Problems on Stochastic Networks

If the dynamics and input data of the problems 1-6 are known then the stochastic network can be obtained by the following way:

Each position (x, t) , $x \in X$, $t = 0, 1, 2, \dots, T_2$ of the dynamical system L can be identified with a vertex (x, t) of the network and each vector of the control parameter $u(t)$ which provide a system passage from the state $x(t) = (x, t)$ to the state $x(t + 1) = (y, t + 1)$ is associated with a directed edge $e = ((x, t), (y, t + 1))$ of our network. To each directed edge $e = ((x, t), (y, t + 1))$ -originated in the uncontrollable position (x, t) - we associate the probability $p(e) = p(u(t))$ (, where $u(t)$ is the vector of control parameters which determines the passage from the state $x = x(t)$ to the state $x(t + 1) = (y, t + 1)$). Additionally we associate to each edge $e = ((x, t), (y, t + 1))$ the cost $c((x, t), (y, t + 1)) = c_t(x(t), x(t + 1))$ (which corresponds to the cost of the system to pass from the state $x(t)$ to the state $x(t + 1)$). The obtained stochastic network has a structure of a T_2 -partite network.

5. Resume

After having introduced a new comfortable structure to analyze stochastic networks on partite networks, we propose a new procedural concept for the underlying control problems. Suitable algorithms exploiting the time-expanded network method and distinguished Markov properties method will be presented.

References

- [1] Bellman R., Kalaba R., Dynamic programming and modern control theory. Academic Press, New York and London (1965).
- [2] Boltjanski, W.G., Optimale Steuerung diskreter Systeme. Leipzig Akademische Verlagsgesellschaft Geest & Portig K.-G., Leipzig (1976).
- [3] Howard, R.A., Dynamic Programming and Markov Processes. Wiley (1960).
- [4] Lozovanu D., Pickl S., Optimization and Multiobjective Control of Time-Discrete Systems. Springer (2009).
- [5] Lozovanu D., Pickl S., Algorithms for solving multiobjective discrete control problems and dynamic c -games on networks. Discrete Applied Mathematics 155 (2007) 1856–1857.

Routing and Wavelength Assignment in optical networks by independent sets in conflict graphs

Lucile Belgacem,^a Irène Charon,^b Olivier Hudry^b

^aOrange Labs FT R&D, 38-40, rue du Gnral Leclerc, 92794 Issy-les-Moulineaux Cedex 9, France

^bTelecom ParisTech & CNRS - LTCI UMR5141, 46, rue Barrault, 75634 Paris Cedex 13, France

Key words: Combinatorial optimization, Optical networks WDM, Routing and wavelength assignment (RWA), Scheduled Lightpath Demands (SLD), Heuristic, Descent method, Independent set, Conflict graph

1. Introduction

We consider two problems related to the *routing and wavelength assignment problem* (RWA) in *wavelength division multiplexing (WDM) optical networks*. For a given network topology, represented by an undirected graph \mathcal{G} , the RWA problem consists in establishing a set of traffic demands (or connection requests) in this network. Traffic demands may be of three types: static (permanent and known in advance), scheduled (requested for a given period of time) and dynamic (unexpected). In this communication, we deal with the case of *scheduled lightpaths demands* (SLDs), which is relevant because of the predictable and periodic nature of the traffic load in real transport networks (more intense during working hours, see [4]).

An SLD can be represented by a quadruplet $s = (x, y, \alpha, \beta)$, where x and y are some vertices of \mathcal{G} (source and destination nodes of the connection request), and where α and β denote the set-up and tear-down dates of the demand. The routing of $s = (x, y, \alpha, \beta)$ consists in setting up a lightpath between x and y , i.e. a path between x and y in \mathcal{G} and a wavelength w . In order to satisfy the SLD s , this lightpath must be reserved during all the span of $[\alpha, \beta]$.

The same wavelength must be used on all the links travelled by a lightpath (*wavelength continuity constraint*). Moreover, at any given time, a wavelength can be used at most once on a given link; in other words, if two demands overlap in

time, they can be assigned the same wavelength if and only if their routing paths are disjoint in edges (*wavelength clash constraint*).

The two problems that we consider here are the following:

- minimize the number of wavelengths necessary to satisfy all the demands;
- given a number of wavelengths, maximize the number of connection demands that can be satisfied with this number of wavelengths.

These problems are NP-hard (see [3]), and have been extensively studied (see, among others, [1], [2], [4], [5] and the reference therein). In both problems, a solution is defined by specifying, for each SLD, the lightpath chosen for supporting the connection (i.e. a path and a wavelength), so that there is no conflict between any two lightpaths (let us recall that two lightpaths are in conflict if they use the same wavelength, they have at least one edge in common and the corresponding demands overlap in time). To solve these problems, we design a modelisation of the problem as the search of successive independent sets (IS) in some conflict graphs. Then we apply a descent heuristic improved by a post-optimization method.

2. Independent sets in conflict graphs

To solve these problems, we build a *conflict graph* \mathcal{H} defined as follows. For each SLD $s = (x, y, \alpha, \beta)$, we compute a given number k of paths between x and y in \mathcal{G} (for instance, $k = 5$): $C_s^1, C_s^2, \dots, C_s^k$. We associate a vertex of \mathcal{H} with each path C_s^i for each SLD s . Thus, if δ denotes the number of SLDs, the number of vertices of \mathcal{H} is equal to $k\delta$. The edges of \mathcal{H} are of two types:

- for each SLD s and for $1 \leq i < j \leq k$, all the edges $\{C_s^i, C_s^j\}$ are in \mathcal{H} ; thus these edges induce, for each SLD s , a clique (i.e., a complete graph) on the vertices $C_s^1, C_s^2, \dots, C_s^k$;
- for any SLD $s = (x, y, \alpha, \beta)$ and any other SLD $s' = (z, t, \gamma, \epsilon)$, we add the edges $\{C_s^i, C_{s'}^j\}$ for $1 \leq i \leq k$ and $1 \leq j \leq k$ if the time windows of s and s' overlap ($[\alpha, \beta] \cap [\gamma, \epsilon] \neq \emptyset$) and if the paths C_s^i and $C_{s'}^j$ are not edge-disjoint; such an edge $\{C_s^i, C_{s'}^j\}$ represents a conflict between s and s' : it is not possible to assign a same wavelength to s and s' if we decide to route s thanks to C_s^i and s' thanks to $C_{s'}^j$.

Our algorithm consists in applying the following two steps successively:

- compute an independent set I in \mathcal{H} ;
- remove from \mathcal{H} all the cliques associated with the satisfied SLDs to obtain a new current conflict graph \mathcal{H} .

We perform this process in order to obtain a series of ISs I_1, I_2, \dots, I_q in successive conflict graphs. This will provide a solution to our problem. Indeed, if the vertex C_s^i belongs to I_j , then we route s thanks to the path C_s^i with the j -th wavelength. Each I_j allows us to route $|I_j|$ SLDs with a same wavelength. We stop when all the SLDs are satisfied (first problem) or when q is equal to the prescribed number of wavelengths (second problem).

3. The heuristic to compute an independent set

To compute an IS in the current conflict graph \mathcal{H} , we apply an *iterative improvement method*, also called *descent* (we tried more sophisticated methods as simulated annealing, but these methods were too long to obtain interesting results). We start from an IS of cardinality 1, and we look for another IS of cardinality 2, 3, and so on, until reaching a value λ for which we do not succeed in finding an IS of cardinality λ . Then the method returns the last IS of cardinality $\lambda - 1$ as a solution.

To look for an IS I_λ of cardinality λ from an IS $I_{\lambda-1}$ of cardinality $\lambda - 1$, we add a random vertex to $I_{\lambda-1}$. Usually, we thus obtain a set I_λ inducing a subgraph containing some edges. Then we try to minimize the number of edges by performing elementary (or local) transformations, in order to find a set I_λ which will be an IS. It is for this minimization that we apply a descent.

The elementary transformation that we adopt consists in removing a vertex belonging to I_λ and simultaneously to add another vertex which does not belong to I_λ . Such a transformation is indeed accepted if the number of edges decreases.

When the descent stops, if the set I_λ still induces a subgraph which is not an IS, then we stop and we keep the previous IS $I_{\lambda-1}$ as the solution. Otherwise, we add a vertex and we apply the same process once again.

In fact, we improve this method in two manners. The first one consists, after the construction of each IS I , in trying to add extra SLDs with a greedy algorithm. For this, we consider each unsatisfied SLD s , and we look for a path in \mathcal{G} that would allow us to route s with the current wavelength, i.e. a path which would not contain any edge of paths associated with another SLD s' routed with the same wavelength and of which the time window overlaps the one of s . Such a situation may occur since we limit ourselves to k paths in the construction of \mathcal{H} , while we look for a path in \mathcal{G} to add extra SLDs.

The other improvement consists in applying a post-optimization method (already applied in [1]), after the computation of the series of ISs I_1, I_2, \dots, I_q . The aim is to reduce the overall values of the wavelengths in order to decrease the total number of wavelengths for the first problem or to make some place to extra SLDs,

still unsatisfied, for the second problem. For this, given a wavelength w , we try to empty, at least partially, the set of SLDs routed with w , by assigning them lower wavelengths. So we change the wavelengths assigned to SLDs which are currently routed with the wavelengths $1, 2, \dots, w-1$; in this process, all the SLDs with a current wavelength between $1, 2, \dots, w-1$ will keep a wavelength in this interval. For the first problem, it may then happen that a wavelength becomes useless; then we remove it definitively and so the number of required wavelengths decreases. For the second problem, the changes involved in the assignments of the wavelengths are such that, sometimes, we may route an SLD which was unsatisfied; so this process allows us to route extra SLDs.

4. Results

We experimentally study the impact of the formulation of the problem as the search of successive ISs in conflict graphs as well as the impact of the post-optimization method. The experiments are done on several networks, with numbers of SLDs up to 3000. The results, not detailed here, show that these two approaches are quite beneficial for both problems, when their results are compared to the one of the method developed in [5].

References

- [1] L. Belgacem, I. Charon, O. Hudry: A post-optimization method for the routing and wavelength assignment problem. Submitted for publication
- [2] L. Belgacem, N. Puech: Solving Large Size Instances of the RWA Problem Using Graph Partitioning. Proceedings of the 12th Conference on Optical Network Design and Modelling, Vilanova i la Geltr, Spain, 1–6, 2008.
- [3] I. Chlamtac, A. Ganz, G. Karmi: Lightpath communications: an approach to high-bandwidth optical WANs. *IEEE Trans. Commun.* 40, 1171–1182, 1992.
- [4] J. Kuri, N. Puech, M. Gagnaire, E. Dotaro, R. Douville: Routing and Wavelength Assignment of Scheduled Lightpaths Demands, *IEEE Journal on Selected Areas in Communications* 21 (8), 1231–1240, 2003.
- [5] N. Skorin-Kapov: Heuristic Algorithms for the Routing and Wavelength Assignment of Scheduled Lightpath Demands in Optical Networks. *IEEE Journal on Selected Areas in Communications* 24 (8), 2-15, 2006.

Recognition of Reducible Flow Hypergraphs^{*}

A. L. P. Guedes,^a L. Markenzon,^b L. Faria^c

^a*Universidade Federal do Paraná, PR, Brazil*
andre@inf.ufpr.br

^b*Universidade Federal do Rio de Janeiro, RJ, Brazil*
markenzon@nce.ufrj.br

^c*Universidade do Estado do Rio de Janeiro, RJ, Brazil*
luerbio@cos.ufrj.br

Key words: Directed hypergraphs, Reducible flow graphs, Recognition algorithm

1. Introduction

Reducible flow graphs were introduced by Allen [1] to model the control flow of computer programs. Although they were initially used in code optimisation algorithms, several theoretical and applied problems have been solved for the class.

Directed hypergraphs [2; 4] are a generalisation of digraphs and they can model binary relations among subsets of a given set. Such relationships appears in different areas of Computer Science such as database systems [2], parallel programming [9] and scheduling [5]. Reducible flow hypergraphs were defined by Guedes *et al.* [7; 6].

In this paper, we present flow hypergraphs and the extension of reducibility for this family. We show that the characterisation of reducible flow graphs using the transformation approach yields a polynomial recognition algorithm.

^{*} Partially supported by grants 485671/2007-7, 306893/2006-1 and 473603/2007-1, CNPq, Brazil.

2. Directed Hypergraphs

Definition 1. A **directed hypergraph** $H = (V, A)$ is a pair, where V is a non empty finite set of vertices and A is a collection of hyper-arcs. A hyper-arc $a = (X, Y) \in A$ is an ordered pair where X and Y are non empty subsets of V , such that $X = \text{Org}(a)$ is called the **origin** and $Y = \text{Dest}(a)$ is called the **destination** of a .

The notation Org and Dest can be extended to a collection A' of hyper-arcs, where $\text{Org}(A') = \cup_{e \in A'} \text{Org}(e)$ and $\text{Dest}(A') = \cup_{e \in A'} \text{Dest}(e)$.

Definition 2. Let $H = (V, A)$ be a directed hypergraph and $v \in V$. The collection of hyper-arcs entering vertex v is denoted by $BS(v) = \{e \in A \mid v \in \text{Dest}(e)\}$, the **backward star** set of v .

Definition 3. [7] Let $H = (V, A)$ be a directed hypergraph and u and v be vertices of H . A **B-path** of size k from u to v is a sequence $P = (e_{i_1}, e_{i_2}, e_{i_3}, \dots, e_{i_k})$, such that $u \in \text{Org}(e_{i_1})$ and $v \in \text{Dest}(e_{i_k})$, and for each hyper-arc e_{i_p} of P , $1 \leq p \leq k$, we have:

- $\text{Org}(e_{i_p}) \subseteq (\text{Dest}(e_{i_1}, e_{i_2}, \dots, e_{i_{p-1}}) \cup \{u\})$
- $\text{Dest}(e_{i_p}) \cap (\text{Org}(e_{i_{p+1}}, e_{i_{p+2}}, \dots, e_{i_k}) \cup \{v\}) \neq \emptyset$.

$\text{Org}(e_{i_1})$ and $\text{Dest}(e_{i_k})$ are denoted by $\text{Org}(P)$ and $\text{Dest}(P)$, respectively.

Definition 4. A **flow hypergraph** $H = (V, A, s)$ is a triple, such that (V, A) is a directed hypergraph, $s \in V$ is a distinguished source vertex, and there is a B-path from s to each other vertex in V .

3. Reducibility of Flow Hypergraphs

In [8], Hecht and Ullman presented a characterisation of reducible flow graphs based on two transformations. We extend these operations in order to define reducible flow hypergraphs and to develop a recognition algorithm.

Given a flow hypergraph, two transformations, T_1 and T_2 , can be defined, performing the contraction of a hyper-arc.

Definition 5. (T_1) Let $H = (V, E, s)$ be a flow hypergraph and $a = (\{x\}, \{x\}) \in E$ be a simple loop. The hypergraph $T_1(H, a)$ is defined as $H - \{a\}$.

Definition 6. (T_2) Let $H = (V, E, s)$ be a flow hypergraph and $a = (\{x\}, Y) \in E$ be a hyper-arc (with $|\text{Org}(a)| = 1$), such that $\forall y \in Y \setminus \{x\}, \text{Org}(BS(y)) = \{x\}$;

$x = s$ or $s \notin Y$; and a is not a simple loop. The hypergraph $T_2(H, a)$ is defined by removing a from H and merging the vertices of Y with x .

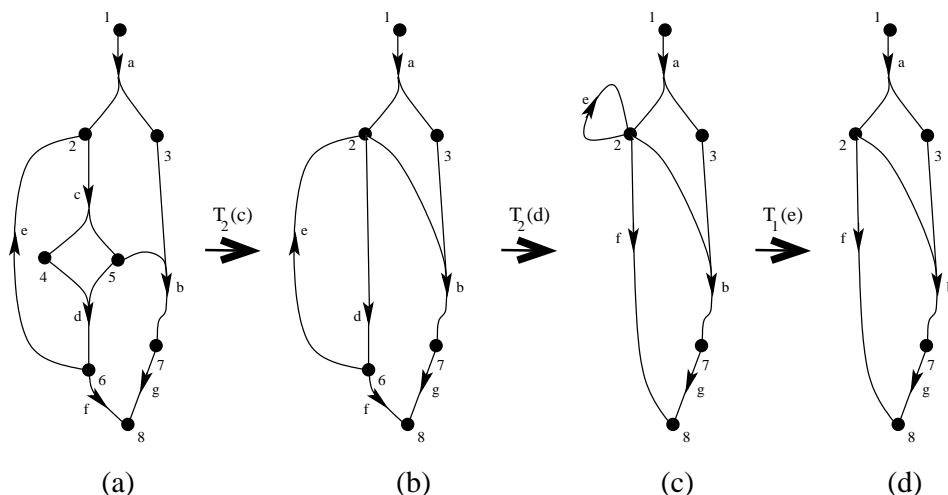


Fig. 1. Transformations T_1 and T_2

To use such transformations in an algorithm they need to be, together, a finite “Church-Rosser” transformation [3; 8]. This means that starting with flow hypergraph H , any sequence of hypergraphs generated by applying of these transformations is finite and ends generating the same hypergraph, says $T^*(H)$.

Theorem 3.1. Let H be a flow hypergraph. There is a unique flow hypergraph $T^*(H)$ resulting from any sequence of applications of T_1 and T_2 in H and in which T_1 and T_2 can not be applied.

The reducibility can now be defined in terms of the transformations.

Definition 7. H is called **reducible** if $T^*(H)$ is a flow hypergraph with just one vertex and no hyper-arcs.

Definition 8. Let $H = (V, A)$ be a directed hypergraph and $a \in A$. The hyper-arc a is **contractible** if one of the transformations T_1 or T_2 can be applied at a (see Definitions 5 and 6).

It can be proved that any sequence of possible transformations is valid. So, the recognition algorithm consists in applying the transformations to any contractible hyper-arc (it is necessary to verify this condition). It stops when there are no more contractible hyper-arcs; at this point, the resulting flow hypergraph must be checked.

It is easy to see that the algorithm always stop and leads to the $T^*(H)$ flow hypergraph.

```

reducible( $H = (V, A, s)$ )
   $T^* \leftarrow H$ 
  repeat
    find a contractible hyper-arc  $a$  in  $G$ 
    apply the appropriate transformation in  $T^*$  at  $a$ 
  until there is no contractible hyper-arc
  test if  $T^* = (\{s\}, \emptyset, s)$ 

```

The following aspects must be considered to establish the complexity of the algorithm:

- testing whether or not a hyper-arc $a = (\{x\}, Y)$ is contractible can be performed in time $\mathcal{O}(|Y|\Delta)$, being Δ the maximum size of the backward star sets ($BS(v)$). So, it takes $\mathcal{O}(|V| \times |A|)$.
- Finding a contractible hyper-arc in the flow hypergraph $H = (V, A, s)$ may imply that all hyper-arcs are tested. So, it takes $\mathcal{O}(|V| \times |A|^2)$.
- If H is reducible then every hyper-arc will be contractible, at some point. So, the above test may be performed $|A|$ times. The whole algorithm takes time $\mathcal{O}(|V| \times |A|^3)$.

References

- [1] F. E. Allen. Control flow analysis. *SIGPLAN Not.*, 5(7):1–19, 1970.
- [2] G. Ausiello, A. D’Atri, and D. Saccà. Strongly equivalent directed hypergraphs. In *Analysis and Design of Algorithms for Combinatorial Problems*, vol. 25 of *Ann. of Disc. Math.*, 1–25. North-Holland, Amsterdam, 1985.
- [3] A. Church and J. B. Rosser. Some properties of conversion. *Trans. of the American Mathematical Society*, 39(3):472–482, 1936.
- [4] G. Gallo, G. Longo, S. Nguyen, and S. Pallottino. Directed hypergraphs and applications. *Disc. Appl. Math.*, 42:177–201, 1993.
- [5] G. Gallo and M. Scutellà. Minimum makespan assembly plans. Technical Report TR-10/98, Dip. di Inf., Univ di Pisa, 1998.
- [6] A.L.P. Guedes, L. Markenzon, and L. Faria. Flow hypergraph reducibility. *Electronic Notes in Discrete Mathematics*, 30:255 – 260, 2008.
- [7] A.L.P. Guedes. *Hipergrafos Direcionados*. D.Sc. thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2001. (in portuguese)
- [8] M. S. Hecht and J. D. Ullman. Flow graph reducibility. *SIAM J. of Computing*, 2(2):188–202, 1972.
- [9] S. Nguyen, D. Pretolani, and L. Markenzon. On some path problems on oriented hypergraphs. *RAIRO - Informatique Theorique et Applications (Theoretical Informatics and Applications)*, 32(1):1–20, 1998.

Complexity

Lecture Hall A

Wed 3, 15:45–16:45

On the complexity of graph-based bounds for the probability bounding problem

Andrea Scozzari,^a Fabio Tardella^a

^aUniversità di Roma “La Sapienza” Via del Castro Laurenziano 9, 00161 Roma,
Dip. di Matematica per le Decisioni Economiche, Finanziarie ed Assicurative
{andrea.scozzari, fabio.tardella}@uniroma1.it

Key words: coherent probability, boolean probability bounding problem, cherry trees, chordal graphs, NP-completeness

1. Extended Abstract

In many real world applications we often need to reason with uncertain information under partial knowledge. A common situation is when a coherent probability assessment \mathcal{P}_n is defined on a family of n conditional or unconditional events. Given a further event, logically dependent on the others, there exists an interval $[p', p'']$ such that every probability $p_{n+1} \in [p', p'']$ assigned to the new event together with \mathcal{P}_n forms a coherent probability assessment on the resulting family of $n + 1$ events. In case of unconditional events the above result is known as the *Fundamental Theorem of the Theory of Probability* of de Finetti [5; 6]. The problem of finding the maximum and minimum value of p_{n+1} such that $\mathcal{P}_n \cup p_{n+1}$ is coherent, has been already identified in the seminal work of Boole, who called it the “general problem in the theory of probabilities” [1]. In this paper we focus on the special case of finding upper bounds for the probability of the (logical) union of n events when the individual probabilities of the events as well as the probabilities of all intersections of k -tuples of these events are known, where $k \leq m < n$, and m is called the *order* of the bound. This problem is also known as the *Boolean Probability Bounding Problem* (BPBP) [3]. In spite of its long history, its theoretical complexity is still unknown. More precisely, we have recently shown [2] that for this problem the complexity of deciding whether a given probability assessment is coherent is NP-hard, and we strongly conjecture that this is also the case for the problem of finding the best upper bound for the union, when we are given a feasible probability assessment for the events and their intersections. However, this is still an open problem. Since BPBP is hard in practice, several authors have proposed efficient techniques for finding relaxed solutions (bounds) for this problem (see [3; 4; 7; 9; 11; 13]

and the references therein). However, while most bounding procedures clearly require polynomial time, the complexity of some recent graph-based methods has not yet been established. Let A_1, \dots, A_n be a set of arbitrary events in a probability space Ω . We assume that the single events probability $P(A_i)$, $i = 1, \dots, n$, and the probability of the intersections $P(\bigcap_{i \in I} A_i)$ for all subsets $I \subseteq \{1, 2, \dots, n\}$ with cardinality $|I| \leq m < n$, are known. Our problem consists in finding upper bounds for the probability of the union of the n events $P(A_1 \cup A_2 \cup \dots \cup A_n)$. In this paper we focus on graph-based upper bounds. Let $G = (V, E)$ be the *complete* graph associated to the events A_1, \dots, A_n , with vertex set $V = \{1, 2, \dots, n\}$, and associate with each edge (i, j) a weight $w_{ij} = P(A_i \cap A_j)$. Hunter and Worsley [9; 13] have shown that the inequality

$$P(A_1 \cup A_2 \cup \dots \cup A_n) \leq \sum_{i=1}^n P(A_i) - \sum_{(i,j) \in T} w_{ij} \quad (1.1)$$

holds for every spanning tree T of G . Thus, the second order bound provided by the right hand side of (1.1), which can be computed by solving a max weight spanning tree of G , is called the Hunter-Worsley bound. Bukszár and Prékopa [4] presented a third order bound, which considers also the intersections among triples of events, and improves on the one by Hunter and Worsley. This third order bound is based on a new type of graph called cherry tree. A *cherry tree* is a triple $\Delta = (V, E, \varphi)$, where (V, E) is an undirected graph, and the set φ is a collection of subsets of vertices of cardinality three called cherries. *Cherries* are recursively defined in the following manner (see [4]): (i) An adjacent pair of vertices is the only cherry tree with exactly two vertices of G ; (ii) from a cherry tree, obtain a new cherry tree by adding a new vertex and two new edges connecting this new vertex with two already existing vertices of the tree. The vertices belonging to these two edges constitute a cherry. With every cherry $\{i, j, k\} \in \varphi$ we associate a weight w_{ijk} . The weight of a cherry tree Δ is:

$$W(\Delta) = \sum_{(i,j) \in E} w_{ij} - \sum_{\{i,j,k\} \in \varphi} w_{ijk}. \quad (1.2)$$

Given weights $w_{ij} = P(A_i \cap A_j)$, $i \neq j$, and $w_{ijk} = P(A_i \cap A_j \cap A_k)$, $i \neq j \neq k$, Bukszár and Prékopa provided the following bound on the probability of the union:

$$P(A_1 \cup A_2 \cup \dots \cup A_n) \leq \sum_{i=1}^n P(A_i) - W(\Delta). \quad (1.3)$$

In particular, the Bukszár and Prékopa's bound is obtained by considering a special cherry tree called *t-cherry tree*. A *t-cherry tree* is found when at each step in (ii), the two vertices to which a new vertex is connected are adjacent. In this case a cherry constitutes a triangle and a *t-cherry tree* is a triangulated graph. Bukszár and Prékopa also provide a polynomial time algorithm for finding a *t-cherry tree* by first calculating a maximum spanning tree of the graph G associated to the events A_1, \dots, A_n , and then improving it to find a spanning *t-cherry tree*. However, the *t-cherry tree* provided by this algorithm is not guaranteed to be the heaviest one. Moreover, in their paper the complexity of finding the heaviest *t-cherry tree* spanning the complete graph G is not assessed. We note that the

weights $w_{ij} = P(A_i \cap A_j)$ and $w_{ijk} = P(A_i \cap A_j \cap A_k)$ are not arbitrary values, but they must represent a feasible and coherent probability distribution for the events A_1, \dots, A_n . Another third order upper bound, which exploits the concept of chordal graphs, is introduced in [3; 7]. Let $C = (V, E)$ be a chordal graph with weights w_{ij} associated to the edges and weights w_{ijk} associated to the triangles of C . Denote by $E(C)$ the set of the edges of C , and by $\Gamma(C)$ the set of triangles whose sides belong to C . We define the weight $W(C)$ of the graph C as:

$$W(C) = \sum_{(i,j) \in E(C)} w_{ij} - \sum_{\{i,j,k\} \in \Gamma(C)} w_{ijk}. \quad (1.4)$$

Given any chordal graph C spanning G with weights $w_{ij} = P(A_i \cap A_j)$ and $w_{ijk} = P(A_i \cap A_j \cap A_k)$, the following is an upper bound on the probability of the union [3; 7]:

$$P(A_1 \cup A_2 \cup \dots \cup A_n) \leq \sum_{i=1}^n P(A_i) - W(C) \quad (1.5)$$

Also in this case, neither in [3] nor in [7], the complexity of finding the chordal graph C of maximum weight $W(C)$ is assessed. We note that finding a spanning chordal graph of maximum weight $W(C)$ is an extension of the problem of finding a maximum weight spanning chordal subgraph of a given graph G in the special case where $w_{ijk} = 0 \forall i \neq j \neq k$. This problem has been uncertain for a long time. The first proof of NP -completeness is attributed to A. Ben-Dor in [10]. Here, we study the problem of finding a chordal graph C spanning G of maximum weight $W(C)$ by exploiting some recent results on the complexity of the problem of deciding the existence of a maximum spanning chordal subgraph of a given graph [12]. Our main results concern the NP -completeness of the following two open problems:

1. Maximum Weight Spanning t -Cherry Tree Bounding problem

INSTANCE: A complete graph $G = (V, E)$ with weights $0 \leq w_{ij} \leq 1$ assigned to its edges, and weights $0 \leq w_{ijk} \leq 1$ assigned to its triangles that represent a coherent probability distribution for the events A_1, \dots, A_n .

QUESTION: Does there exist a t -Cherry Tree spanning the graph G with total weight $W(\Delta)$ at least L ?

2. Maximum Weight Chordal Bounding problem

INSTANCE: A complete graph $G = (V, E)$ with weights $0 \leq w_{ij} \leq 1$ assigned to its edges, and weights $0 \leq w_{ijk} \leq 1$ assigned to its triangles that represent a coherent probability distribution for the events A_1, \dots, A_n .

QUESTION: Does there exist a chordal graph C spanning G with total weight $W(C)$ at least L ?

References

- [1] Boole G., Laws of thought, *American Reprint of 1854 edition*, Dover, 1854.
- [2] Boros E., Scozzari A., Tardella F., Veneziani P., Polynomially Computable Bounds for the Probability of a Union of Events, working paper.
- [3] Boros E., Veneziani P., Bounds of degree 3 for the probability of the union of events, *Rutcor Research Report*, RRR 3-2002, 1-21.
- [4] Bukszár J., Prékopa A., Probability bounds with cherry trees, *Mathematics of Operations Research*, 26 2001, 174-192.
- [5] de Finetti B., Sull'impostazione assiomatica delle probabilità, *Annali Università di Trieste*, 19 1949, 3-55.
- [6] de Finetti B., Teoria della Probabilità, Einaudi, Torino, 1970.
- [7] Dohmen K., Bonferroni-Type Inequalities via Chordal Graphs, *Combinatorics, Probability and Computing*, 11 2002, 349-351.
- [8] Hailperin T., Best possible inequalities for the probability of a logical function of events, *American Mathematical Monthly*, 72 1965, 343-359.
- [9] Hunter D., An upper bound for the probability of a union *Journal of Applied Probability*, 13 1976, 597-603.
- [10] Natanzon A., Shamir R., Sharan R., Complexity classification of some edge modification problems, *Discrete Applied Mathematics*, 113 2001, 109-128.
- [11] Prékopa A., Gao L., Bounding the Probability of the Union of Events by the Use of Aggregation and Disaggregation in Linear Program, *Discrete applied Mathematics*, 145 2005, 444-454.
- [12] Scozzari A., Tardella F., On the complexity of some subgraph problems *Discrete applied Mathematics*, to appear.
- [13] Worsley K.J., An improved bonferroni inequality and application *Biometrika*, 69 1982, 297-302.

Integer Flow with Multipliers: The Special Case of Multipliers 1 and 2

Birgit Engels^a Sven Krumke^b Rainer Schrader^a Christiane Zeck^b

^aZAIK, University of Cologne

^bAG Optimization, University of Kaiserslautern

Abstract

The problem to find a valid integer flow with flow multipliers on nodes or arcs is long known to be NP-complete [8]. We show that the problem is still hard when restricted to instances with a limited number of integral multipliers. We demonstrate that for the multipliers 1 and 2 optimal solutions with fractions $\frac{1}{2^n}$, $n \in \mathbb{N}$ can occur. For special instances which are motivated by some applications we prove that the optimal solution is halfintegral. Finally, we extend the *Successive Shortest Path Algorithm* [2; 6; 7], to the minimum cost flow problem with multipliers. For the application based instances with halfintegral optimal solutions, we try to find acceptable integral solutions.

Key words: generalized flow, successive shortest paths, rounding heuristics

1. Introduction

A flow $f : A \rightarrow \mathbb{R}$ is a function which assigns a flow value $f(a_{ij})$ to each arc of a digraph $N = (V, A)$ (called network) such that the capacity ($\forall a_{ij} \in A : l_{ij} \leq f(a_{ij}) \leq u_{ij}$) and balance ($\forall v_i \in V : \sum_{a_{li} \in A} f(a_{li}) - \sum_{a_{ik} \in A} f(a_{ik}) = b(v_i)$, $\forall a_i \in A : f_{out}(a_{ij}) = f_{in}(a_{ij})$) constraints hold. Generalized flows or flows with gains and losses differ from such flows in networks in one respect: a flow may be damped or amplified when traversing an arc. Depending on the arc multiplier μ_{ij} , a unit of flow entering arc a_{ij} can result in more or less than one unit leaving the arc. Denoting the incoming and outgoing flow by $f_{in}(a_{ij})$ and $f_{out}(a_{ij})$ we have $f_{out}(a_{ij}) = \mu_{ij} \cdot f_{in}(a_{ij})$.

The introduction of flow multipliers destroys the total unimodularity of the network matrix and thus integral optimal solutions are no longer guaranteed or even impossible. It is known that finding an optimal integer solution is NP-complete for generalized flows [5; 8]. There seem to be no results for the special case where the multipliers are restricted to a few fixed numbers or even to the single additional

multiplier 2. However, this special case occurred when we modeled a railway disposition problem. We encountered some 'merely generalized' flow instances with the property that the underlying network essentially is a bipartite digraph, all excesses, demands (node balances), arc capacities (costs) are integral and we have only multipliers 1 and 2.

2. Complexity and fractional solutions

The NP-completeness proof for generalized flows by Sahni [8] employs the subset sum problem. It can be extended to hold for the restriction to multipliers 1 and 2. However this extension does not hold for the disposition networks, i.e. the special graph instances which arise from our application. Thus, we show a reduction from 3V2LSAT, where the constructed graph meets all requirements of Definition 1.

Definition 1. A *disposition network* $N = (V = X \cup Y, A)$ is bipartite digraph with $\mu_a \in \{1, 2\}$ for all arcs a and all arcs with $\mu(a) = 2$ are directed from X to Y . Moreover, for all paths from a source s to a sink t the product of multipliers of all arcs along the path (path multiplier) is *either 1 or 2*.

Let $\alpha = C_1 \wedge \dots \wedge C_n$ be a boolean formula with at most three literals per clause where each variable $v_k, 1 \leq k \leq m$ can only occur 3 times in total and maximal 2 times as one of the corresponding literals. Tovey showed that 3V2LSAT is NP-complete [9]. In our network N_α we have vertices n_k for every variable v_k and additional two vertices n_k^0, n_k^1 for the corresponding literals $\neg l_k, l_k$, which occur in α . For every n_k the node balance b_k is $+1$ and 0 for the literal nodes. Nodes n_k are connected to 'their' literal nodes by arcs with multiplier 2. For each clause C_i in α we add a vertex n_i , which has incoming arcs from those literal nodes which correspond to literals occurring in C_i . Finally we add two sink nodes s_{sat}, s_{rest} with node balances $b_{sat} = -n$ and $b_{rest} = -2m + n$ to the graph. We connect the clause nodes to both sinks by arcs (n_i, s_{sat}) with capacity 1 and arcs (n_i, s_{rest}) with unlimited capacity (see Figure 1). Unless otherwise noted, any node balances are 0 multipliers are 1, capacity unlimited and costs 0. We conclude:

Theorem 11. The decision problem for a valid integral flow in a disposition network with integral values is NP-complete.

An optimal solution to the min cost flow problem with multipliers 1 and 2 can contain arbitrarily small fractions of flow: We can construct examples with flows $\frac{1}{2^n}$ in a graph $G(V, A), |V| = 3n$ in the optimal solution (Figure 1). Yet, we can transform disposition networks into a generalized minimum cost flow circulation instance G' and show that the Circuit Cancelling (CC) algorithm always yields a half-integral solution to the minimum cost circulation problem on G' which can be transferred to the flow instance G .

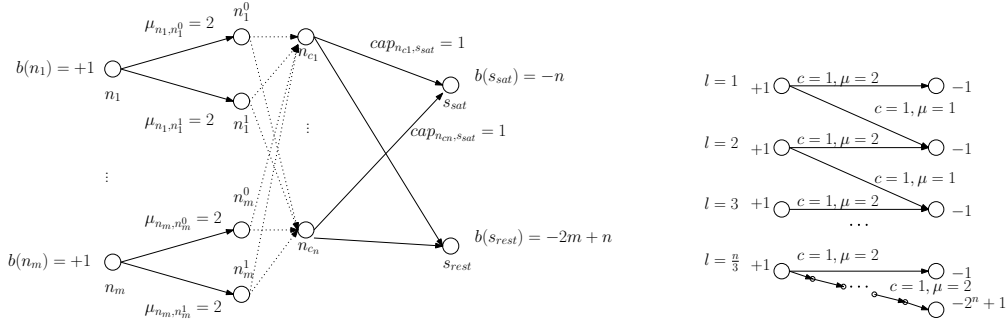


Fig. 1. Construction of network N_α and $\frac{1}{2^n}$ -solution for a graph with $3n$ nodes.

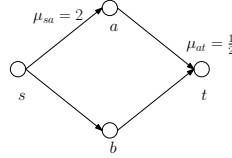


Fig. 2. Two s - t -ways with identical arc cost, but different de facto per unit cost.

3. Modified SSP Algorithm (MSSP)

The CC algorithm is the first (and to our knowledge only) classical combinatorial minimum cost flow or circulation algorithm so far, which was extended to the generalized case by Wayne [10]. (Other approaches for solving generalized flow problems are of course provided by LP techniques and the modified network simplex method of Dantzig [3; 1].) We give a generalization of the *Successive Shortest Path* (SSP) algorithm [7; 6; 2], which is based on the principle of pseudoflows, i.e. flows respecting the non-negativity and capacity constraints, but not the node balance constraints. We first need to define a shortest path on a network with multipliers: Consider Figure 2, where all arc costs and multipliers are 1 if not depicted otherwise. The two possible paths from s to t result in costs of 2 accounting only on arc costs. Yet actually sending one unit of flow along $s - a - t$ creates two units of flow at a which are passed on to t to deliver one unit at t but arise arc cost of 2 on (a, t) .

Definition 2. We define the *path costs* of a flow multiplier path $\pi_{uv} = u_1 - \dots - u_n$ with $u = u_1$ and $u_n = v$ from u to v as: $c'(\pi_{uv}) = \sum_{i=2}^n \prod_{j=1}^{i-1} \mu_{j(j+1)} \cdot c((i-1)i)$.

We can easily adjust Dijkstra's [4] algorithm* by introducing a tentative per node parameter m_v , which accounts for the product of arc multipliers on the (tentative) shortest path from s to every node v . Besides this modification, we take the flow multipliers into account when we compute the maximum flow δ to be augmented. After each augmentation a residual network is built: For each arc $e(u, v)$ with a positive flow $f(e)$, an arc $\bar{e} = (v, u)$ with capacity $cap(\bar{e}) = f(e)$, cost $c(\bar{e}) = -c(e)$, multiplier $\mu_{\bar{e}} = \frac{1}{\mu_e}$ and flow 0 is added. If $f(e)$ equals $cap(e)$, then

* Dijkstra's algorithm can be used as a plug-in to the SSP, because of the throughout non-negative edge weights.

arc e is removed from the network. The correctness can be shown with the reduced cost optimality criterion. The running time of the (unscaled) SSP is pseudopolynomial in the sum of excesses and demands, as in each augmentation at least one unit of flow is sent. As δ does not need to be integral in our case and because there is no general lower bound $\epsilon(n) < \delta(n)$, we cannot give an appropriate running time for general instances. Still we can use the modified SSP on all instances with optimal solutions of certain fractions. Especially, in the case of disposition networks $\delta \geq \frac{1}{2}$, which (without scaling) also results in a (pseudo)polynomial running time.

4. Rounding to acceptable integer solutions

To obtain an acceptable integral solution we temporarily allow $cap(b_j, t)$ to be violated by $\frac{1}{2}$. The heuristic can always be applied to a halfintegral solution until there are only integral flows, because in each iteration, at least two halfintegral flows are rounded (up or down) to integral flows. The node balances can be violated: Although demands do not stay over-saturated in the end (without halfintegral flows, a violation of $cap(b_j, t)$ by 'only' $\frac{1}{2}$ is impossible), there can be additional unsaturated deficits and rest excesses, which could be reallocated by another MSSP application. If we accept the solution nevertheless, we gain a 2-approximate solution.

Rounding
1: while (\exists halfintegral flow f)
3: Find cheapest f from s to t
4: if ($f = f + \frac{1}{2}$ violates $bal(t)$ by at most $\frac{1}{2}$)
5: Round f up, most expensive s - t -flow f' down.
6: else Round f down, cheapest s - t -flow f' up.
8: end

References

- [1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin "Network Flows - Theory, Algorithms and Applications" Prentice Hall, 1993.
- [2] R.G. Busaker, P.J. Gowen "A procedure for determining minimal-cost network flow patterns" ORO Technical Report 15, Operational Research Office, John Hopkins University, Baltimore, MD, 1961.
- [3] G.B. Dantzig "Linear Programming and Extensions" Princeton University Press, Princeton NJ, 1963.
- [4] E. Dijkstra "A note on two problems in connexion with graphs" Numerische Mathematis 1 pp. 269-271, 1959.
- [5] M. Garey and D. Johnson "Computers and Intractability: A guide to the theory of NP-Completeness" W.H. Freeman, New York, 1979.

- [6] M. Iri "A new method of solving transportation-network problems" Journal of the Operations Research Society of Japan 3, 27-87, 1960.
- [7] W.S. Jewell "Optimal Flow through networks with gains" Operations Research 10, 476-499, 1962.
- [8] S. Sahni "Computationally Related Problems" SIAM Jr. on Computing, 3, 4, 262-279, 1974.
- [9] Craig A. Tovey "A Simplified NP-Complete Satisfiability Problem" Discrete Applied Mathematics 8, 85-89, 1984.
- [10] Kevin D. Wayne "A polynomial combinatorial algorithm for generalized minimum cost flow" Mathematics of Operations Research, Vol.27, No.3, 445-459, 2002.

Polyhedra

Lecture Hall B

Wed 3, 15:45–16:45

Classification of 0/1-facets of the hop constrained path polytope defined on an acyclic digraph

Rüdiger Stephan ^a

^a*Institut für Mathematik, Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin*

stephan@math.tu-berlin.de

Key words: path polytope, hop constraints, dynamic programming

1. Introduction

In this paper we study the polytope associated with the hop constrained shortest path problem defined on an acyclic digraph. The aim is to show that all facet defining 0/1-inequalities for this polytope can be classified via the inherent structure of dynamic programming.

Let $D = (N, A)$ be an acyclic digraph with node set $N = \{0, 1, \dots, n\}$ and arc set $A = \{(i, j) : i = 0, \dots, n-1, j = i+1, \dots, n\}$. A $(0, n)$ -path is a set of arcs $\{a_1, \dots, a_r\}$ such that $a_i = (i_{p-1}, i_p)$ for $p = 1, \dots, r$ with $i_0 = 0$ and $i_r = n$. Given a length function $d : A \rightarrow \mathbb{R}$ and a nonnegative integer $k \leq n$, the *hop constrained shortest path problem* is the problem of finding a $(0, n)$ -path with at most k arcs of minimum length. Since D is an acyclic digraph, the problem can be solved in polynomial time for every k and length function d .

The *hop constrained path polytope*, denoted by $P_{0,n\text{-path}}^k(D)$, is the convex hull of the incidence vectors of all paths with at most k arcs. Its integer points are characterized by the system

$$x(\delta^{\text{out}}(0)) = 1, \tag{1.1}$$

$$x(\delta^{\text{in}}(n)) = 1, \tag{1.2}$$

$$x(\delta^{\text{out}}(i)) - x(\delta^{\text{in}}(i)) = 0, \quad i = 1, \dots, n-1, \tag{1.3}$$

$$x(A) \leq k, \tag{1.4}$$

$$x_{ij} \in \{0, 1\} \quad \text{for all } (i, j) \in A. \tag{1.5}$$

Here, $\delta^{\text{out}}(j)$ and $\delta^{\text{in}}(j)$ denote the set of arcs leaving and entering node j , respectively. Moreover, for an arc set $F \subseteq A$ we set $x(F) := \sum_{(i,j) \in F} x_{ij}$.

The hop constrained path polytope and some closely related polyhedra have been paid some attention in the literature, see [3; 4; 5; 6; 7]. However, a complete linear description of $P_{0,n\text{-path}}^k(D)$ is unknown, and to the best of our knowledge just a characterization of all facet defining inequalities $b^T x \geq \beta$ with coefficients $b_{ij} \in \{0, 1\}$ for $(i, j) \in A$ were not given before.

2. All 0/1-facet defining inequalities for $P_{0,n\text{-path}}^k(D)$

The hop constrained path polytope $P_{0,n\text{-path}}^k(D)$ has some nice properties that gain access to a promising polyhedral investigation by the dynamic programming paradigm. We start with a tight connection of this polytope and its *dominant* $\text{dmt}(P_{0,n\text{-path}}^k(D)) := P_{0,n\text{-path}}^k(D) + \mathbb{R}_+^A$. Proofs are omitted due to lack of space.

Theorem 12. Denote by $P_{0,n\text{-path}}(D)$ the ordinary path polytope ($k = n$). Then, for every nonnegative integer k , $P_{0,n\text{-path}}^k(D) = \text{dmt}(P_{0,n\text{-path}}^k(D)) \cap P_{0,n\text{-path}}(D)$.

Theorem 12 implies that a complete linear description of $P_{0,n\text{-path}}^k(D)$ can be given by nonnegative inequalities $b^T x \geq \beta$, that is, $b \geq 0$.

From the inherent structure of the Bellman-Ford algorithm [2] a so called *dynamic programming graph* $\mathcal{D} = (\mathcal{N}, \mathcal{A})$ for the hop constrained shortest path problem can be quite easily constructed. The algorithm computes (the value of) a shortest (i, j) -path for each pair of nodes (i, j) , provided D has no negative cycles. In the main loop of the algorithm the length of a shortest (i, j) -path with at most ℓ arcs will be computed for $\ell = 2, \dots, n$. The correctness of the algorithm is based on the *Bellman Equations*

$$u_{ij}^{(1)} = d_{ij}, \quad u_{ij}^{(\ell)} = \min_{m \in N} (u_{im}^{(\ell-1)} + d_{mj}) \quad \text{for } \ell = 2, \dots, n, \quad (2.6)$$

where u_{ij}^ℓ denotes the length of a shortest (i, j) -path with at most ℓ arcs.

Suppose that the Bellman-Ford algorithm will be executed until $\ell = k$. Fixing $i = 0$, we associate with each accessible state u_{0j}^ℓ a node $[j, \ell]$ and with each possible decision $u_{0j}^{(\ell)} = u_{0m}^{(\ell-1)} + d_{mj}$ an arc $([m, \ell - 1], [j, \ell])$ at cost d_{mj} . Substituting all arcs $([i, \ell], [n, \ell + 1])$ by $([i, \ell], [i, \ell + 1])$ for $i = 2, \dots, n - 1$, $\ell = 1, \dots, \min\{i - 1, k - 2\}$ at cost 0 and removing the nodes $[n, \ell]$, $\ell = 1, \dots, k - 1$, the hop constrained shortest path problem is quite easily viewed as one of finding a shortest $([0, 0], [n, k])$ -path in the digraph $\mathcal{D} = (\mathcal{N}, \mathcal{A})$, where \mathcal{N} is a disjoint union of node sets

$$\mathcal{N}_0 := \{[0, 0]\}, \quad \mathcal{N}_i := \{[i, j] : j = 1, \dots, \gamma_i\}, \quad i = 1, \dots, n - 1, \quad \mathcal{N}_n := \{[n, k]\}$$

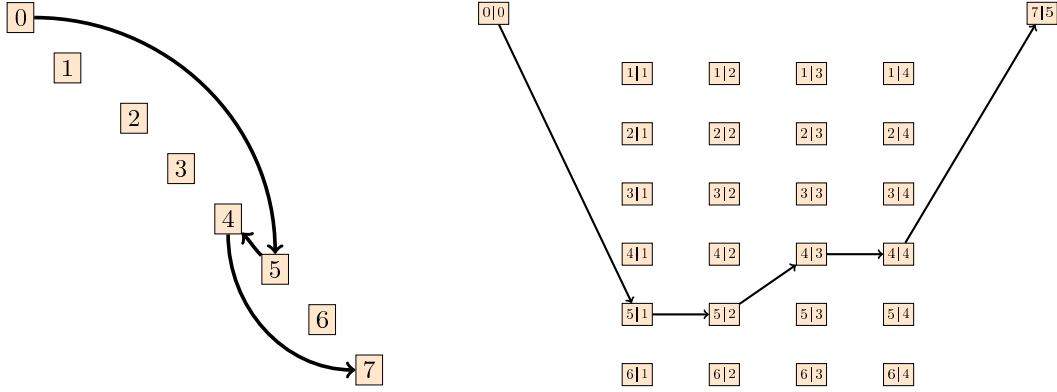


Fig. 1. An acyclic digraph $D = (N, A)$ on node set $N = \{0, 1, \dots, 7\}$ and associated DP-graph $\mathcal{D} = (\mathcal{N}, \mathcal{A})$ for $k = 5$. Arc set A is omitted; Illustration of a $(0, 7)$ -path and its analogue in \mathcal{D} .

with $\gamma_i := \min\{i, k - 1\}$, and \mathcal{A} is given by

$$\begin{aligned} \mathcal{A} = & \{([0, 0], [n, k])\} \cup \{([0, 0], [i, 1]), ([i, \gamma_i], [n, k]) : i = 1, \dots, n - 1\} \\ & \cup \{([i, j], [i, j + 1]) : i = 2, \dots, n - 1, j = 1, \dots, \gamma_i - 1\} \\ & \cup \{([i, j], [h, j + 1]) : i = 1, \dots, n - 2, j = 1, \dots, \min\{k - 2, \gamma_i\}, \\ & \qquad \qquad \qquad h = i + 1, \dots, n - 1\}. \end{aligned}$$

An illustration of the model is given in Figure 1.

The Bellman equations (2.6) provide us an avenue to derive a classification of all 0/1-facet defining inequalities for $\text{dmt}(P_{0,n\text{-path}}^k(D))$. To this end, we consider the numbers $u_{0j}^{(h)}$, $[j, h] \in \mathcal{N}$ as values of a set function $\pi_0 : \mathcal{N} \rightarrow \mathbb{R}$, that is, $\pi_0([j, h]) := u_{0j}^{(h)}$ for all $[j, h] \in \mathcal{N}$. Each function $\pi : \mathcal{N} \rightarrow \mathbb{R}$ induces a valid inequality $\sum_{(i,j) \in A} \mu_{ij}^\pi x_{ij} \geq \pi([n, k]) - \pi([0, 0])$ for $\text{dmt}(P_{0,n\text{-path}}^k(D))$ via

$$\mu_{ij}^\pi := \max\{0\} \cup \{\pi([j, \ell]) - \pi([i, h]) : ([i, h], [j, \ell]) \in \mathcal{A}\} \text{ for } (i, j) \in A. \quad (2.7)$$

This result can be proved with the projection theory of Balas [1] applied to $\text{dmt}(P_{0,n\text{-path}}^k(D))$ and $\text{dmt}(P_{[0,0],[n,k]\text{-path}}(\mathcal{D}))$.

Observations

- (i) π_0 is row-monotone. A function $\pi : \mathcal{N} \rightarrow \mathbb{R}$ is called *row-monotone* if $\pi([i, j]) \geq \pi([i, j + 1])$ for $i = 1, \dots, n - 1, j = 1, \dots, \gamma_i - 1$.
- (ii) π_0 is up-monotone. A function $\pi : \mathcal{N} \rightarrow \mathbb{R}$ is said to be *up-monotone* if for every node $[i, j] \in \mathcal{N} \setminus \{[0, 0]\}$, $\pi([i, j]) = \pi([h, \ell]) + \mu_{hi}^\pi$ for some arc $([h, \ell], [i, j]) \in \delta^{\text{in}}([i, j])$, where $\mu_{ii}^\pi := 0$.

We define an analogue of property (2) for $\delta^{\text{out}}([i, j])$. A function $\pi : \mathcal{N} \rightarrow \mathbb{R}$ is said to be *down-monotone* if for every node $[i, j] \in \mathcal{N} \setminus \{[n, k]\}$, $\pi([h, \ell]) =$

$\pi([i, j]) + \mu_{ih}^\pi$ for some arc $([i, j], [h, l]) \in \delta^{\text{out}}([i, j])$, where $\mu_{ii}^\pi := 0$. Next, π is said to be *row-up-and-down-monotone* if it is row-, up-, and down-monotone. Finally, π is called *tight* if for each arc $(i, j) \in A$ it exists an arc $([i, h], [j, \ell]) \in \mathcal{A}$ such that $\mu_{ij}^\pi = \pi([j, \ell]) - \pi([i, h])$.

Denote by Π the collection of all set functions $\pi : \mathcal{N} \rightarrow \mathbb{R}$ with $\pi([0, 0]) = 0$, $\pi([n, k]) = 1$, and $0 \leq \pi([i, j]) \leq 1$ for all $[i, j] \in \mathcal{N} \setminus \{[0, 0], [n, k]\}$. We are now able to characterize all 0/1-facet defining inequalities for the dominant $\text{dmt}(P_{0,n\text{-path}}^k(D))$.

Theorem 13. Row-up-and-down-monotone 0/1-functions $\pi \in \Pi$ and nontrivial facet defining 0/1-inequalities for $\text{dmt}(P_{0,n\text{-path}}^k(D))$ are in 1-1-correspondence, that is,

- (a) each row-up-and-down-monotone 0/1-function $\pi \in \Pi$ induces a facet defining 0/1-inequality for $\text{dmt}(P_{0,n\text{-path}}^k(D))$;
- (b) each nontrivial facet defining 0/1-inequality for $\text{dmt}(P_{0,n\text{-path}}^k(D))$ is induced by a row-up-and-down-monotone 0/1-function $\pi \in \Pi$;
- (c) if two row-up-and-down-monotone 0/1-functions $\pi, \tilde{\pi} \in \Pi$ induce the same facet defining inequality for $\text{dmt}(P_{0,n\text{-path}}^k(D))$, then $\pi = \tilde{\pi}$.

Corollary 1. Tight row-up-and-down-monotone 0/1-functions $\pi \in \Pi$ and nontrivial facet defining 0/1-inequalities for $P_{0,n\text{-path}}^k(D)$ are in 1-1-correspondence.

References

- [1] E. BALAS, *Projection, lifting and extended formulation in integer and combinatorial optimization.*, Ann. Oper. Res., 140 (2005), pp. 125–161.
- [2] R. BELLMAN, *On a routing problem.*, Q. Appl. Math., 16 (1958), pp. 87–90.
- [3] G. DAHL, N. FOLDNES, AND L. GOUVEIA, *A note on hop-constrained walk polytopes.*, Oper. Res. Lett., 32 (2004), pp. 345–349.
- [4] G. DAHL AND L. GOUVEIA, *On the directed hop-constrained shortest path problem.*, Oper. Res. Lett., 32 (2004), pp. 15–22.
- [5] G. DAHL AND B. REALFSEN, *The cardinality-constrained shortest path problem in 2-graphs.*, Networks, 36 (2000), pp. 1–8.
- [6] V. H. NGUYEN, *A complete linear description for the k-path polyhedron*, Preprint 2003.
- [7] R. STEPHAN, *Facets of the (s,t)-p-path polytope*, <http://www.citebase.org/abstract?id=oai:arXiv.org:math/0606308>, 2006.

The k -gear composition and the stable set polytope

A. Galluccio,^a C. Gentile,^a M. Macina,^a P. Ventura^a

^a*IASI-CNR, viale Manzoni 30, 00185 Rome (Italy)*
{galluccio,gentile,ventura}@iasi.cnr.it,
matteo.macina@fastwebnet.it

Key words: Stable set polytope, Polyhedral combinatorics

1. Introduction

Given a graph $G = (V(G), E(G))$ and a vector $w \in \mathbb{Q}_+^V$ of node weights, the *stable set problem* is to find a set of pairwise nonadjacent nodes (*stable set*) of maximum weight. The *stable set polytope* $STAB(G)$ is the convex hull of the incidence vectors of the stable sets of G : finding its linear description has been one of the major research problems in combinatorial optimization. A useful strategy to face this problem is by defining graph compositions that have polyhedral counterparts for the stable set polytope, such as substitutions or complete joins [3]. The *gear composition*, introduced in [5], is one of these operations: it builds a graph G by replacing a suitable edge e of a given graph H with the graph B (*gear*) shown in Fig. 1(a).

The polyhedral properties of the gear composition have been extensively studied in [4]. There we proved that all the facet defining inequalities for $STAB(G)$ are obtained by extending the facet defining inequalities describing $STAB(H)$ and $STAB(H^e)$, where H^e is obtained from H by subdividing the edge e . Inequalities generated by repeated applications of the gear composition are named *multiple geared inequalities*.

The gear composition revealed also a very effective tool to approach the longstanding open problem of finding a linear description of the stable set polytope of claw-free graphs, i.e. graphs such that the neighborhood of each node has no stable set of size three. For these graphs there exist polynomial time algorithms to optimize over $STAB(G)$ [9; 10]; so, by the equivalence of optimization and separation problems [8], one would expect that an explicit linear description for $STAB(G)$ when G is claw-free is easy to get. But, as noticed in [8], “in spite of considerable efforts, no decent system of inequalities describing $STAB(G)$ for claw-free graphs is known”.

Using the decomposition theorem for claw-free graphs of Chudnovsky and Seymour [1; 2] and the gear composition, we provided the defining linear system of

$STAB(G)$ for a large subclass of claw-free graphs with stability number at least 4. Indeed, Chudnovsky and Seymour [1; 2] stated that a claw-free graph which does not admit a 1-join either has stability number at most 3 or it is a fuzzy circular interval graph or a composition of three types of graphs, called *strips*: fuzzy linear interval strips, fuzzy XX -strips, fuzzy antihat strips. In [6] we proved that an XX -strip is in fact a gear plus two extra nodes. This led us to consider the claw-free graphs that are obtained by composing fuzzy linear interval strips and XX strips. We named these graphs XX -graphs and we proved that:

Theorem 1.1. [7] The stable set polytope of XX -graphs is described by nonnegativity inequalities, rank inequalities, lifted 5-wheel inequalities and multiple geared inequalities.

In this paper we look for a generalization of the above result. In particular, we generalize the gear into a k -gear and, accordingly, we extend the class of claw-free graphs with the graphs obtained by composing fuzzy linear interval strips, fuzzy antihat strips, and k -gears. Interestingly, the stable set problem on this superclass of claw-free graphs is still polynomial time solvable by adapting the algorithm in [10]. Thus, as for the claw-free graphs, the linear description of their stable set polytope should be “easy” to obtain. Here, we study the polyhedral structure of $STAB(G)$ when G results from the k -gear composition of a k -gear and a given graph H .

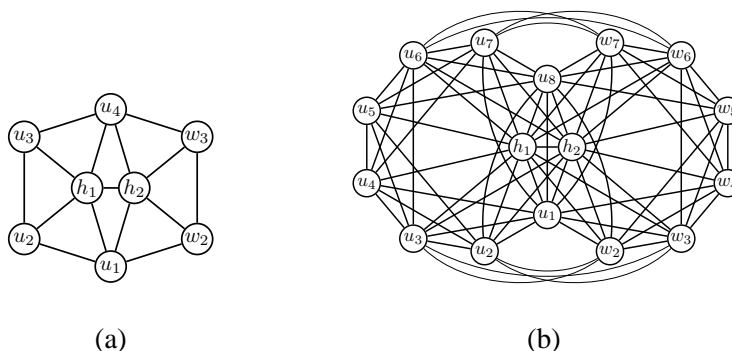


Fig. 1. (a) A gear; (b) A 4-gear.

2. k -gear graphs and the k -gear composition

A $(2k + 1)$ -antiwheel $W = (h : u_1, \dots, u_{2k+1})$ consists of a k -antihole \overline{C}_{2k+1} defined on nodes u_1, \dots, u_{2k+1} plus a *hub* h adjacent to each node of \overline{C}_{2k+1} .

Definition 1. Let $W_1 = (h_1 : u_1, \dots, u_{2k+1})$ and $W_2 = (h_2 : w_1, \dots, w_{2k+1})$ be two $(2k + 1)$ -antiwheels with $k \geq 2$. The k -gear B_k is the graph such that

- 1) $V(B_k) = V(W_1) \cup V(W_2)$ and $u_1 = w_1, u_{2k} = w_{2k}, h_1 = w_{2k+1}$, and $h_2 = w_{2k+1}$,
- 2) $E(B_k)$ is $E(W_1) \cup E(W_2)$ plus the edges $u_i w_j, \forall i, j \in \{2, \dots, k-1\}$ and $u_i w_j, \forall i, j \in \{k+2, \dots, 2k-1\}$.

A k -gear graph B_k with $k = 4$ is depicted in Fig. 1(b). Since a $(2k + 1)$ -wheel coincides with a $(2k + 1)$ -antiwheel when $k = 2$, we have that the gear as defined in [5] is actually a 2-gear. Interestingly, while $STAB(B_2)$ does not admit a facet defining inequality that has full support on the gear B_2 , for $k \geq 3$ the k -gear B_k does support an inequality that is facet defining for $STAB(B_k)$. Indeed we prove that:

Theorem 2.1. Let $k \geq 3$ and let B_k be a k -gear. Then the inequality

$$\sum_{v \in V(B_k) \setminus \{h_1, h_2\}} x_v + 2(x_{h_1} + x_{h_2}) \leq 3. \quad (2.1)$$

is facet defining for $STAB(B_k)$.

An edge $v_1 v_2$ of a graph H is said to be *simplicial* if $K_1 = N(v_1) \setminus \{v_2\}$ and $K_2 = N(v_2) \setminus \{v_1\}$ are two nonempty cliques of H .

Definition 2. Let H be a graph with a simplicial edge $v_1 v_2$ and let B_k be a k -gear, $k \geq 2$. The k -gear composition of H and B_k produces a k -geared graph G , denoted by $G = (H, B_k, v_1 v_2)$, such that $V(G) = V(H) \setminus \{v_1, v_2\} \cup V(B_k)$ and $E(G) = E(H) \setminus (\delta(v_1) \cup \delta(v_2)) \cup E(B_k) \cup F_1 \cup F_2$, with $F_1 = \{u_k u, u_{k+1} u \mid u \in K_1\}$ and $F_2 = \{w_k u, w_{k+1} u \mid u \in K_2\}$ (see Fig. 2).

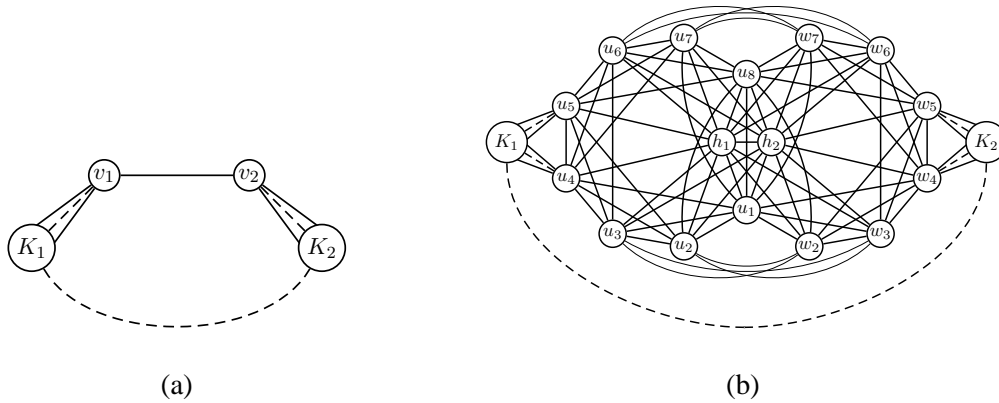


Fig. 2. (a) H with a simplicial edge $v_1 v_2$; (b) the k -geared graph $G = (H, B_k, v_1 v_2)$.

For $k = 2$ we obtain the definition of gear composition given in [5]. In the following we show that, as the gear composition, also the k -gear composition has the polyhedral feature of preserving the property of an inequality of being facet defining for the stable set polytope. In particular, we show that:

Theorem 2.2. Let H be a graph with simplicial edge v_1v_2 and let (π, π_0) be a facet defining inequality for $STAB(H)$ different from $x_{v_1} + x_{v_2} \leq 1$ and such that $\pi_{v_1} = \pi_{v_2} = \lambda > 0$. Let B_k be a k -gear graph with $k \geq 2$. Then the following inequality, called *k -geared inequality* associated with (π, π_0) ,

$$\sum_{v \in V(H) \setminus \{v_1, v_2\}} \pi_v x_v + \lambda \sum_{v \in V(B_k) \setminus \{h_1, h_2\}} x_v + 2\lambda(x_{h_1} + x_{h_2}) \leq \pi_0 + 2\lambda \quad (2.2)$$

is facet defining for $STAB(G)$, where $G = (H, B_k, v_1v_2)$ is the k -geared graph.

It is worth noticing that the above results do not hold if we generalize the gear with $(2k + 1)$ -wheels instead of $(2k + 1)$ -antiwheels.

References

- [1] M. Chudnovsky and P. Seymour. The structure of claw-free graphs. In *Surveys in Combinatorics*, volume 327 of *London Math. Soc. Lecture Notes*. 2005.
- [2] M. Chudnovsky and P. Seymour. Claw-free graphs IV: Decomposition theorem. *J. Comb. Th. B*, 98(5):839–938, 2008.
- [3] V. Chvátal. On certain polytopes associated with graphs. *J. Comb. Th. B*, 18:138–154, 1975.
- [4] A. Galluccio, C. Gentile, and P. Ventura. Gear composition of stable set polytopes and \mathcal{G} -perfection. Technical Report 661, IASI - CNR, 2007. To appear in *Mathematics of Operations Research*.
- [5] A. Galluccio, C. Gentile, and P. Ventura. Gear composition and the stable set polytope. *Operations Research Letters*, 36:419–423, 2008.
- [6] A. Galluccio, C. Gentile, and P. Ventura. The stable set polytope of claw-free graphs I: XX -strip composition versus gear composition. 2008. Submitted.
- [7] A. Galluccio, C. Gentile, and P. Ventura. The stable set polytope of claw-free graphs II: XX -graphs are \mathcal{G} -perfect. 2008. Submitted.
- [8] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer Verlag, Berlin, 1988.
- [9] G.J. Minty. On maximal independent sets of vertices in claw-free graphs. *J. Comb. Th. B*, 28:284–304, 1980.
- [10] G. Oriolo, U. Pietropaoli, and G. Stauffer. A new algorithm for the maximum weighted stable set problem in claw-free graphs. *LNCS*, 5035:77–96, 2008.

Plenary Session II

Lecture Hall A

Wed 3, 16:45-17:30

Diameter and Center Computations in Networks

Michel Habib

LIAFA, CNRS & Université Paris Diderot
habib@liafa.jussieu.fr

Key words: Diameter, Breadth-First Search, δ -hyperbolic metric spaces

Abstract:

Starting from a very practical question : *what is the best algorithm available to compute or to approximate the diameter of a huge network ?* At first, to compute the diameter of a given graph, it seems necessary to compute all-pairs shortest paths, which is not known to be computable in linear time, see [1] and [10].

We first present the well-known 2-sweep Breadth-First Search approximation procedure and some experimental results of its randomized version on huge graphs [2], [4], [5], [11]. In order to explain the efficiency of this 2-sweep **linear time** procedure, we study its properties on various graph classes and its relation with classical graph parameters such as: k-chordality or tree-length [7].

We then emphasize on a notion introduced by M. Gromov in 1987 [8], namely the δ -hyperbolic metric spaces via a simple 4-point condition: for any four points u, v, w, x the two larger of the distance sums $d(u, v) + d(w, x)$, $d(u, w) + d(v, x)$, $d(u, x) + d(v, w)$ differ by at most 2δ . δ -hyperbolic metric spaces play an important role in geometric group theory, geometry of negatively curved spaces, and have recently become of interest in several areas of computer science including computational geometry and networking.

A connected graph $G = (V, E)$ equipped with its standard graph metric d_G is δ -hyperbolic if the metric space (V, d_G) is δ -hyperbolic. This very interesting notion captures the distance from a graph to a tree in a metric way. Moreover δ -hyperbolicity is polynomially computable and easy to approximate.

We survey some results about δ -hyperbolicity of graphs, and in particular we show that δ -hyperbolicity generalizes tree-length with respect to the 2-sweep algorithm. We provide some experimental results on real data (i.e. graphs extracted

from the Internet), showing that δ -hyperbolicity can be a very practical measure [9].

We finish with a comparison on the computational complexity of the diameter and the center of a given graph, listing some open questions [12], [3].

References

- [1] T. M. Chan, All-pairs shortest paths for unweighted undirected graphs in $o(mn)$ time, SODA 2006.
- [2] V. Chepoi, F. Dragan, *A linear time Algorithm for finding a central vertex of a chordal graph*, ESA (1994) 159-170.
- [3] V. Chepoi, F. Dragan, B. Estellon, M. Habib, Y. Vaxès, *Diameters, centers, and approximation trees of δ -hyperbolic spaces and graphs*, ACM Computational Geometry Conf. (2008) 59-68.
- [4] D.G. Corneil, F. Dragan, M. Habib, C. Paul, *Diameter determination on restricted graph families*, Discrete Applied Mathematics 113 (2001) 143-166.
- [5] D.G. Corneil, F. Dragan, E. Khler, *On the power of BFS to determine a graph's diameter*, Networks, vol 42(4), (2003) 209-223.
- [6] F. Dragan, *Estimating all pairs shortest paths in restricted graph families: a unified approach*, J. of Algorithms 57(2005) 1-21.
- [7] Y. Dourisboure, C. Gavoille, *Tree-decomposition of graphs with small diameter bags*, Discrete Math. 307 (2007) 208-229.
- [8] M. Gromov, *Hyperbolic Groups*, in Essays in Group Theory (S.M. Gersten ed.), MSRI Series 8 (1987) 75-263.
- [9] R. Kleinberg, *Geographic routing Using hyperbolic space*, INFOCOM (2007) 1902-1909.
- [10] D. Kratsch, J. Spinrad, *Between $O(mn)$ and $O(n^\alpha)$* , SODA (2003) 709-716.
- [11] C. Magnien, M. Latapy, M. Habib, *Fast Computation of Empirically Tight Bounds for the Diameter of Massive Graphs*, ACM J. of Experimental Algorithms, 13 (2008).
- [12] M. Parnas, D. Ron, *Testing the diameter of Graphs*, Random Structures & Algorithms Vol 20, (2002)165 - 183.

Polynomial-time Algorithms

Lecture Hall A

Thu 4, 08:45–10:15

Integer programming with 2-variable equations and 1-variable inequalities

Manuel Bodirsky,^a Gustav Nordh,^b Timo von Oertzen^c

^a*École Polytechnique, LIX (CNRS UMR 7161), Palaiseau, France*
bodirsky@lix.polytechnique.fr

^b*Theoretical Computer Science Laboratory, Linköping University, Sweden*
nordh@lix.polytechnique.fr

^c*Max Planck Institute for Human Development, Berlin, Germany*
vonoertzen@mpib-berlin.mpg.de

Key words: Integer programming, computational complexity, efficient algorithms

1. Introduction

We present an efficient algorithm to find an *optimal integer solution* of a given system of 2-variable equalities and 1-variable inequalities with respect to a given linear objective function. More precisely, the input consists of

- a finite set of variables x_1, \dots, x_n ,
- equations of the form $ax + by = c$ where x and y are variables, and a, b, c are rational numbers,
- inequalities of the form $x \leq u$ or $x \geq l$, where x is a variable and u, l are rational numbers, and
- a linear objective function $\sum_{i=1}^n w_i x_i$ where the w_i 's are rational numbers.

The task is to find an assignment of *integer values* to the variables x_1, \dots, x_n such that all equations and inequalities are satisfied and the function $\sum_{i=1}^n w_i x_i$ is maximized.

If instead of 2-variable equalities we are given 2-variable *inequalities*, then the problem obviously becomes NP-hard (this can be seen by a reduction from the maximum independent set problem). If instead of 2-variable equalities we are given 3-variable equalities, then the problem again becomes NP-hard. This follows by a trivial reduction from the NP-complete problem 1-in-3-SAT, where each 1-in-3 clause $1\text{-in-3}(x, y, z)$ is reduced to $x + y + z = 1, x, y, z \geq 0$. Hence, 2-variable

equalities and 1-variable inequalities is a maximal tractable class in the sense that allowing longer equalities *or* longer inequalities results in NP-hard problems.

We remark that finding integer solutions to linear equation systems *without inequalities* is tractable [4], and has a long history. Faster algorithms have been found in for example [2; 8]. Integer programming can also be solved in polynomial time if the *total* number of variables is two [6]; Lenstra [5] has generalized this result to any fixed finite number of variables. See [3] for one of the fastest known algorithms for 2-variable integer programming, and for more references about linear programming in two dimensions.

In our algorithm for a system of 2-variable equalities and 1-variable inequalities over the integers we use the fact that such equation systems reduce to systems that define a one-dimensional solution space. This idea was also used by Aspvall and Shiloach [1] in their algorithm for solving systems of 2-variable equations over the *rational numbers*. We use this fact to compute an *optimal* integer solution by solving a system of modular equations in polynomial time. One of our contributions is to show that the necessary computations can be performed in total quadratic time in the input size.

Since the problem to decide whether a single 2-variable equation $ax + by = c$ with $a, b, c \in \mathbb{Z}$ has an integer solution for x, y is equivalent to deciding whether the gcd of a and b is a divisor of c , we cannot expect an algorithm for our problem that is faster than gcd computations. There are sub-quadratic algorithms for computing the gcd of two N bit integers with running times in $O(\log(N)M(N))$, where $M(N)$ is the bit-complexity of multiplying two N bit integers. Using the classical Schönhage Strassen [7] integer multiplication algorithm, this gives a running time for gcd in $O(N(\log(N))^2 \log(\log(N)))$.

We view it as an interesting open problem whether integer programming over 2-variable equalities and 1-variable inequalities can be shown to be no harder than gcd computations, i.e., with a running time of $O(G(N))$ where $G(N)$ is the bit complexity of computing gcd of two N bit integers. We also emphasize that the quadratic time algorithm we give does not rely on sub-quadratic algorithms for multiplication, division, or gcd.

2. Reduction to an acyclic system

We show how to partition the system of equations into independent subsystems, each having a one-dimensional solution space (i.e., the solution space can be expressed using one free parameter). The *graph* of an instance of our problem is the graph that has a vertex for each variable and an edge for each equation (connecting the two vertices corresponding to the variables in the equation). A instance of our

problem is called an *acyclic* (or *connected*) system if the graph of the system is acyclic (or *connected*, respectively), and a *connected component* of a system F is a subsystem of F whose graph is a connected component of the graph of F .

Proposition 1. There is an $O(N^2)$ time algorithm that computes for a given system of two variable linear equations an equivalent acyclic subsystem.

The upper and lower bounds on the variables translate into an upper and lower bound on the parameter x . If y has the upper bound u and the expression for y is $y = ax + c$, then in case that a is positive we get the bound $\lfloor (\lfloor u \rfloor - c) / a \rfloor \geq x$, and in case a is negative we get the bound $\lceil (\lfloor u \rfloor - c) / a \rceil \leq x$. Lower bounds l on y are treated analogously. After translating all bounds we can obtain the strongest upper bound and lower bound on x , denoted u^* and l^* respectively (obviously, if $u^* < l^*$, then there is no solution).

3. Computing an optimal solution

Assume for the sake of presentation that the coefficients a, b, c in all equations $ax + by = c$ are integer. This is without loss of generality since every equation can be brought into this form by multiplying both sides of the equation by the product of the denominators of a, b , and c . This can clearly be done in quadratic time and increases the bit size of the input by at most a constant factor. Check that each individual equation $ax + by = c$ has integer solutions. Recall that a Diophantine equation of the form $ax + by = c$ has integer solutions if and only if $\gcd(a, b) \mid c$. Simplify the equations by dividing a, b , and c by $\gcd(a, b)$. In the resulting system we now have $\gcd(a, b) = 1$ for each equation $ax + by = c$.

Proposition 2. There is an $O(N^2)$ time algorithm for solving acyclic connected systems of two variable equations over the integers.

Proof Sketch. We perform a depth-first search on the graph of the system, starting with any variable x from the system. The goal is to find an expression for the solution space of the form $x \equiv s \pmod{t}$. That is, the assignment $x := i$ can be extended to an integer solution to the entire system if and only if $i \equiv s \pmod{t}$. If we enter a variable y in the DFS and y has an unexplored child z , then continue recursively with z . If z is a leaf in the tree, meaning that there is a unique equation $ay + bz = c$ where z occurs, then rewrite the equation as $ay \equiv c \pmod{b}$. Note that an assignment $y := i$ can be extended to a solution of $ay + bz = c$ if and only if $ai \equiv c \pmod{b}$. Compute the multiplicative inverse a^{-1} of $a \pmod{b}$ (which exists since $\gcd(a, b) = 1$ and can be retrieved from the gcd computation). The congruence above can now be rewritten as $y \equiv c' \pmod{b}$ where $c' = ca^{-1}$. If all children of y have been explored, then y is explored and we backtrack. If v is the parent of y through the equation $dv + ey = f$, then rewrite the equation

using the congruence $y \equiv c' \pmod{b}$, into $dv + e(c' + kb) = f$ which is equivalent to $dv + ekb = f - ec'$. Check that $\gcd(d, eb) \mid (f - ec')$ (otherwise there is no solution and we reject) and divide d , eb , and $f - ec'$ by $\gcd(d, eb)$ giving the equation $d'v + e'k = f'$ with $\gcd(d', e') = 1$. Rewrite this equation as the congruence $d'v \equiv f' \pmod{e'}$ which in turn is rewritten as $v \equiv f'' \pmod{e'}$ by multiplying both sides with the multiplicative inverse of $d' \pmod{e'}$ (which again exists since $\gcd(d', e') = 1$). Suppose that v already has an explored child (with congruence $v \equiv c \pmod{b}$) when we have finished exploring another child of v , giving rise to the congruence $v \equiv c' \pmod{b'}$. We then combine these congruences in a similar fashion as discussed above already twice (by computing greatest common divisors and multiplicative inverses). The result (if a solution exists) is a congruence $v \equiv c'' \pmod{b''}$, which replaces the two old congruences, and we continue the depth first search. We omit the proof that the algorithm runs in quadratic time.

Theorem 14. There is an algorithm that computes the optimal solution of a given integer program with 2-variable equalities and 1-variable inequalities in $O(N^2)$ time, where N is the number of bits in the input.

References

- [1] B. Aspvall and Y. Shiloach. A fast algorithm for solving systems of linear equations with two variables per equation. *Linear Algebra and its Applications*, 34:117–124, 1980.
- [2] T. Chou and G. Collins. Algorithms for the solution of systems of linear diophantine equations. *SIAM J. Comput.*, 11(4):687–708, 1982.
- [3] F. Eisenbrand and S. Laue. A linear algorithm for integer programming in the plane. *Math. Program.*, 102(2):249–259, 2005.
- [4] R. Kannan and A. Bachem. Polynomial algorithms for computing the smith and hermite normal forms of an integer matrix. *SIAM J. Comput.*, 8(4):499–507, 1979.
- [5] H.W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.
- [6] H. E. Scarf. Production sets with indivisibilities. part ii: The case of two activities. *Econometrica*, 49:395–423, 1981.
- [7] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971.
- [8] A. Storjohann and G. Labahn. Asymptotically fast computation of hermite normal forms of integer matrices. In *ISSAC*, pages 259–266, 1996.

Faster Min-Max Resource Sharing and Applications

Dirk Müller¹

Research Institute for Discrete Mathematics, University of Bonn
mueller@or.uni-bonn.de

Key words: min-max resource sharing, fractional packing, approximation algorithms, VLSI design

The problem of sharing a set of limited resources between users (customers) in an optimal way is fundamental. The common mathematical model has been called the min-max resource sharing problem. Well-studied special cases are the fractional packing problem and the maximum concurrent flow problem. The only known exact algorithms for these problems use general linear (or convex) programming. Shahrokhi and Matula [9] were the first to design a combinatorial approximation scheme for the maximum concurrent flow problem. Subsequently, this result was improved, simplified, and generalized many times. This work is a further step on this line. In particular we provide a simple algorithm and a simple proof of the best performance guarantee in significantly smaller running time. Moreover, we implemented the algorithm for an application to global routing of VLSI chips.

The problem. The MIN-MAX RESOURCE SHARING PROBLEM is defined as follows. Given finite sets \mathcal{R} of *resources* and \mathcal{C} of *customers*, a convex set \mathcal{B}_c , called *block*, of feasible solutions for customer c (for $c \in \mathcal{C}$), and a nonnegative continuous convex function $g_c : \mathcal{B}_c \rightarrow \mathbb{R}_+^{\mathcal{R}}$ for $c \in \mathcal{C}$ specifying the *resource consumption*, the task is to find $b_c \in \mathcal{B}_c$ ($c \in \mathcal{C}$) approximately attaining

$$\lambda^* := \inf \left\{ \max_{r \in \mathcal{R}} \sum_{c \in \mathcal{C}} (g_c(b_c))_r \mid b_c \in \mathcal{B}_c (c \in \mathcal{C}) \right\}, \quad (0.1)$$

i.e., approximately minimizing the largest resource consumption. We assume that g_c can be computed efficiently and we have a constant $\epsilon_0 \geq 0$ and oracle functions $f_c : \mathbb{R}_+^{\mathcal{R}} \rightarrow \mathcal{B}_c$, called *block solvers*, which for $c \in \mathcal{C}$ and $\omega \in \mathbb{R}_+^{\mathcal{R}}$ return an element $b_c \in \mathcal{B}_c$ with $\omega^\top g_c(b_c) \leq (1 + \epsilon_0) \text{OPT}_c(\omega)$, where $\text{OPT}_c(\omega) := \inf_{b \in \mathcal{B}_c} \omega^\top g_c(b)$. Block solvers are called *strong* if $\epsilon_0 = 0$ or $\epsilon_0 > 0$ can be chosen arbitrarily small, otherwise they are called *weak*.

¹ joint work with J. Vygen

Note that previous authors often required that \mathcal{B}_c is compact, but we do not need this assumption. Some algorithms require *bounded block solvers*: for $c \in \mathcal{C}$, $\omega \in \mathbb{R}_+^{\mathcal{R}}$, and $\mu > 0$, they return an element $b_c \in \mathcal{B}_c$ with $g_c(b_c) \leq \mu \mathbf{1}$ and $\omega^\top g_c(b_c) \leq (1 + \epsilon_0) \inf\{\omega^\top g_c(b) \mid b \in \mathcal{B}_c, g_c(b) \leq \mu \mathbf{1}\}$ (by $\mathbf{1}$ we denote the all-one vector). They can also be strong or weak.

All algorithms that we consider are fully polynomial approximation schemes relative to ϵ_0 , i.e., for any given $\epsilon > 0$ they compute a solution $b_c \in \mathcal{B}_c$ ($c \in \mathcal{C}$) with $\max_{r \in \mathcal{R}} \sum_{c \in \mathcal{C}} (g_c(b_c))_r \leq (1 + \epsilon_0 + \epsilon) \lambda^*$, and the running time depends polynomially on ϵ^{-1} . By θ we denote the time for an oracle call (to the block solver). Moreover, we write $\rho := \sup\{\frac{(g_c(b))_r}{\lambda^*} \mid r \in \mathcal{R}, c \in \mathcal{C}, b \in \mathcal{B}_c\}$.

Previous work. Grigoriadis and Khachiyan [4] were the first to present such an algorithm for the general MIN-MAX RESOURCE SHARING PROBLEM. Their algorithm uses $O(|\mathcal{C}|^2 \log |\mathcal{R}| (\epsilon^{-2} + \log |\mathcal{C}|))$ calls to a strong bounded block solver. They also have a faster randomized version.

In [5] they proposed an algorithm which needs $O(|\mathcal{C}| |\mathcal{R}| (\epsilon^{-2} \log \epsilon^{-1} + \log |\mathcal{R}|))$ calls to a strong, but not bounded, block solver. They also showed that $O(|\mathcal{C}|^2 \log |\mathcal{R}| (\epsilon^{-2} + \log |\mathcal{R}|))$ calls to a strong bounded block solver suffice.

Jansen and Zhang [6] generalized this and allowed *weak block solvers*. Their algorithm needs $O(|\mathcal{C}| |\mathcal{R}| (\log |\mathcal{R}| + \epsilon^{-2} \log \epsilon^{-1}))$ calls to a block solver.

	block solver	running time
Grigoriadis, Khachiyan [4]	strong, bounded	$\tilde{O}(\epsilon^{-2} \mathcal{C} ^2 \theta)$
Grigoriadis, Khachiyan [5]	strong, unbounded	$\tilde{O}(\epsilon^{-2} \mathcal{C} \mathcal{R} \theta)$
Jansen, Zhang [6]	weak, unbounded	$\tilde{O}(\epsilon^{-2} \mathcal{C} \mathcal{R} \theta)$
our algorithm	weak, unbounded	$\tilde{O}(\epsilon^{-2} \rho \mathcal{C} \theta)$
our algorithm	weak, bounded	$\tilde{O}(\epsilon^{-2} \mathcal{C} \theta)$

Table 12. Approximation algorithms for the MIN-MAX RESOURCE SHARING PROBLEM. Running times are shown for fixed $\epsilon_0 \geq 0$, and logarithmic terms are omitted.

Fractional packing. The special case where the functions g_c ($c \in \mathcal{C}$) are linear is often called the FRACTIONAL PACKING PROBLEM (although sometimes this name is used for different problems). For this special case faster algorithms using unbounded block solvers are known. Plotkin, Shmoys and Tardos [8] require a strong block solver and $O(\epsilon^{-2} \rho |\mathcal{C}| \theta (\log |\mathcal{R}| + \epsilon^{-1}))$ oracle calls to solve the feasibility version (where $\lambda^* = 1$ is known). Young's algorithm [11] needs $O(\epsilon^{-2} \rho |\mathcal{C}| (1 + \epsilon_0)^2 \theta \ln |\mathcal{R}|)$ calls to a weak block solver. Charikar et al. extended the result of [8] to weak block solvers resulting in $O(\epsilon^{-2} \rho |\mathcal{C}| (1 + \epsilon_0)^2 \theta \log(\rho(1 + \epsilon_0)\epsilon^{-1}))$ oracle calls.

Bienstock and Iyengar [2] managed to reduce the dependence on ϵ from $O(\epsilon^{-2})$ to $O(\epsilon^{-1})$. Their algorithm does not call a block solver, but requires the resource consumption functions to be explicitly specified by a $|\mathcal{R}| \times \dim(\mathcal{B}_c)$ -matrix G_c for

each $c \in \mathcal{C}$. So their algorithm does not apply to the general MIN-MAX RESOURCE SHARING PROBLEM, but to an interesting special case which includes the MAXIMUM CONCURRENT FLOW PROBLEM. The algorithm solves $O(\epsilon^{-1} \sqrt{Kn \log |\mathcal{R}|})$ separable convex quadratic programs, where $n := \sum_{c \in \mathcal{C}} \dim(\mathcal{B}_c)$, and $K := \max_{1 \leq i \leq |\mathcal{R}|} \sum_{c \in \mathcal{C}} k_i^c$, with k_i^c being the number of nonzero entries in the i -th row of G_c .

	block solver	running time
Plotkin, Shmoys, Tardos [8] *	strong, unbounded	$\tilde{O}(\epsilon^{-2} \rho \mathcal{C} \theta)$
Young [11]	weak, unbounded	$\tilde{O}(\epsilon^{-2} \rho \mathcal{C} \theta)$
Charikar et al. [3] *	weak, unbounded	$\tilde{O}(\epsilon^{-2} \rho \mathcal{C} \theta)$
Bienstock, Iyengar [2]	—	$\tilde{O}(\epsilon^{-1} \sqrt{Kn} T_{QP})$
our algorithm	weak, unbounded	$\tilde{O}(\epsilon^{-2} \rho \mathcal{C} \theta)$
our algorithm	weak, bounded	$\tilde{O}(\epsilon^{-2} \mathcal{C} \theta)$

Table 13. Approximation algorithms for the fractional packing problem. Entries with * refer to the feasibility version ($\lambda^* = 1$). Running times are shown for fixed $\epsilon_0 \geq 0$, and logarithmic terms are omitted. T_{QP} is the time for solving a convex separable quadratic program over $\mathcal{B}_{c_1} \times \dots \times \mathcal{B}_{c_{|\mathcal{C}|}}$.

Our results. We describe an algorithm for the general MIN-MAX RESOURCE SHARING PROBLEM. It uses ideas of Grigoriadis and Khachiyan [5], Young [11], Albrecht [1], and Vygen [10]. The same algorithm and a quite simple analysis yields two results: With a weak unbounded block solver we obtain a running time of $O(|\mathcal{C}| \theta \rho (1 + \epsilon_0)^2 \log |\mathcal{R}| (\log |\mathcal{R}| + \epsilon^{-2} (1 + \epsilon_0)))$. This generalizes several results for the linear case and improves on results for the general case for moderate values of ρ . With a weak bounded block solver the running time is $O(|\mathcal{C}| \theta (1 + \epsilon_0)^2 \log |\mathcal{R}| (\log |\mathcal{R}| + \epsilon^{-2} (1 + \epsilon_0)))$. This improves on previous results by roughly a factor of $|\mathcal{C}|$ or $|\mathcal{R}|$. The running times are summarized in Tables 1 and 2.

Our motivation is an application to VLSI design. In global routing instances of the (nonlinear) MIN-MAX RESOURCE SHARING PROBLEM occur naturally when dealing with today's constraints and objectives (see e. g. [7]). We incorporate a speed-up technique that drastically decreases the number of oracle calls in practice. We generalize the randomized rounding paradigm to our problem and obtain an improved bound. Finally we present experimental results for instances from current chips, with millions of customers and resources. We show that such problems can be solved efficiently.

References

- [1] Albrecht, C.: Global routing by new approximation algorithms for multicommodity flow. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems 20 (2001), 622–632

- [2] Bienstock, D., and Iyengar, G.: Concurrent Flows in $O^*(\frac{1}{\epsilon})$ iterations. In Proceedings of the 2004 Symposium on Theory of Computing, pp. 146–155, 2004.
- [3] Charikar, M., Chekuri, C., Goel, A., Guha, S., and Plotkin, S.: Approximating a finite metric by a small number of tree metrics. Proceedings of the 39th Annual IEEE Symposium on the Foundations of Computer Science (1998), pp. 379–388
- [4] Grigoriadis, M. D., and Khachiyan, L. D.: Fast approximation schemes for convex programs with many blocks and coupling constraints. SIAM Journal on Optimization, Vol. 4, No. 1, pp. 86–107, 1994.
- [5] Grigoriadis, M. D., and Khachiyan, L. D.: Coordination complexity of parallel price-directive decomposition. Mathematics of Operations Research, Vol. 21, No. 2, pp. 321–340, 1996.
- [6] Jansen, K., and Zhang, H.: Approximation algorithms for general packing problems and their application to the multicast congestion problem. Mathematical Programming 114, Ser. A, pp. 183–206, 2008.
- [7] Müller, D.: Optimizing yield in global routing. Proceedings of the IEEE International Conference on Computer-Aided Design, pp. 480–486, 2006.
- [8] Plotkin, S.A., Shmoys, D.B., and Tardos, É.: Fast approximation algorithms for fractional packing and covering problems. Mathematics of Operations Research 2 (1995), 257–301
- [9] Shahrokhi, F., and Matula, D.W.: The maximum concurrent flow problem. Journal of the ACM 37 (1990), 318–334
- [10] Vygen, J.: Near-optimum global routing with coupling, delay bounds, and power consumption. In: Integer Programming and Combinatorial Optimization; Proceedings of the 10th International IPCO Conference; LNCS 3064 (G. Nemhauser, D. Bienstock, eds.), Springer, Berlin 2004, pp. 308–324
- [11] Young, N.E.: Randomized rounding without solving the linear program. Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms (1995), pp. 170–178

The Anonymous Subgraph Problem

Andrea Bettinelli,^a Leo Liberti,^b Franco Raimondi,^c
David Savourey^b

^a*Dept. of Mathematics, Università degli Studi di Milano, Italy*
andrea.bettinelli@unimi.it

^b*LIX, École Polytechnique, F-91128 Palaiseau, France*
{liberti,savourey}@lix.polytechnique.fr

^c*Dept. of Computer Science, University College London, UK*
f.raimondi@cs.ucl.ac.uk

Key words: anonymity, anonymous routing, secret santa, graph's topology

1. Introduction

Many problems can be modeled as the search for a subgraph $S \subseteq A$ with specific properties, given a graph $G = (V, A)$. There are applications in which it is desirable to ensure also S to be *anonymous*. In this work we formalize an anonymity property for a generic family of subgraphs and the corresponding decision problem. We devise an algorithm to solve a particular case of the problem and we show that, under certain conditions, its computational complexity is polynomial. We also examine in details several specific family of subgraphs.

2. Characterization of anonymity

Given a digraph $G = (V, A)$, let $|V| = n$ and $|A| = m$. We are interested in finding if a certain family \mathcal{A} of subgraphs is *anonymous* with respect to G . By anonymous we mean that it is not possible to single out a subgraph $S \in \mathcal{A}$, nor to identify any other arc in the subgraph, given the topology of the graph and a subset C of the arcs in S . We call C a *partial view* of S . Let $PV : \mathcal{P}(A) \times \mathcal{A} \rightarrow \{0, 1\}$ be the function

$$PV(X, S) = \begin{cases} 1 & X \text{ is a partial view of } S \\ 0 & \text{otherwise} \end{cases}$$

that defines which subsets are considered a partial view of a certain subgraph.

Definition 4. (Anonymous family of subgraphs) Given a digraph $G = (V, A)$ and a function $PV : \mathcal{P}(A) \times \mathcal{A} \rightarrow \{0, 1\}$, a family of subgraphs $\mathcal{A} \subseteq \mathcal{P}(A)$ is *anonymous* in G if

$$\forall S \in \mathcal{A}, \forall C \in \{X \mid PV(X, S) = 1\}, \forall b \in S \setminus C \quad \exists T \in \mathcal{A} : C \subseteq T \wedge b \notin T.$$

We call *anonymous subgraphs* the elements of an anonymous family \mathcal{A} . It is now possible to define *Anonymous Subgraph Problem* (ASP) as the decision problem of checking if a family of subgraphs contains an anonymous family with respect to a graph.

Definition 5. (Anonymous Subgraph Problem) Given a digraph $G = (V, A)$, a function $PV : \mathcal{P}(A) \times \mathcal{A} \rightarrow \{0, 1\}$, and a family of subgraphs \mathcal{S} , is there a non empty subset \mathcal{A} of \mathcal{S} which is anonymous in G ?

Here we restrict our analysis to the case where the set of partial views of a subgraph S is $\{C \mid C \subseteq S \wedge |C| = 1\}$, i.e. only one arc of the subgraph is known. With this restriction, we obtain the following definition of anonymity:

Definition 6. Given a digraph $G = (V, A)$, a family of subgraphs $\mathcal{A} \subseteq \mathcal{P}(A)$ is *anonymous* in G if

$$\forall S \in \mathcal{A}, \forall a \neq b \in S \quad \exists T \in \mathcal{A} : a \in T \wedge b \notin T.$$

We will refer to ASP1 to denote the Anonymous Subgraph Problem where Definition 6 is used to characterize anonymity. In Section 3 we propose an algorithm to solve the ASP1 and we show under what conditions its computational complexity is polynomial in the size of the graph G , even if the family \mathcal{S} contains a combinatorial number of subgraphs.

3. Algorithm

Algorithm 1 Algorithm for solving the ASP1

```

1: FINDANONYMOUSSG( $G, \mathcal{S}, P$ ):
2: for all  $a \neq b \in P$  do
3:   if FINDSG( $G, \mathcal{S}, P \setminus \{b\}, \{a\}$ ) =  $\emptyset$  then
4:     return FINDANONYMOUSSG( $G, \mathcal{S}, P \setminus \{a\}$ )
5:   end if
6: end for
7: return FINDSG( $G, \mathcal{S}, P, \emptyset$ )

```

Algorithm 1 solves the ASP1: it returns an element of \mathcal{A} , if \mathcal{A} exists, and an

empty set otherwise. It is a recursive algorithm and at the top level P is equal to the arc set A . The algorithm is based on the following observation: if there exists two distinct arcs $a, b \in A$ such that no subset $T \in \mathcal{S}$ contains a but not b , it implies that all the subsets $S \in \mathcal{S}$ that contain a are not anonymous. Thus, we can transfer the anonymity property from the subsets to the arcs. The algorithm iteratively remove arcs from the set P of permitted arcs and uses this set as additional constraints when looking for possible subgraphs. If no subgraphs can be found satisfying the additional constraints given by P the family is not anonymous in G .

We assume the correctness of the subroutine $\text{FINDSG}(G, \mathcal{S}, P, X)$: it returns an empty set if and only if it doesn't exist a subgraph $S \in \mathcal{S} \cap \mathcal{P}(P) : x \in S \forall x \in X$.

Theorem 15. Alg. 1 correctly solves the ASP1.

Proof. First we observe that, if the algorithm returns a non empty solution, at line 7 the set \mathcal{A} of subgraphs in \mathcal{S} where every arc belongs to P is anonymous in G . Let $S \subseteq P$ be an element of \mathcal{A} , we know that $\forall a, b \in S \exists T \in \mathcal{A}$ s.t. $a \in T$ and $b \notin T$, otherwise a would have been banned from P . Assume now there is a solution to ASP1 and Alg. 1 fails. The existence of a solution implies the existence of a non empty anonymous set \mathcal{A} . If our algorithm reached line 7 with $\mathcal{A} \subseteq \mathcal{P}(P)$, then, because $\text{FINDSG}(G, \mathcal{S}, P, X)$ is correct, our algorithm would not have failed. Thus we know that the algorithm reached line 7 with $\mathcal{A} \setminus \mathcal{P}(P) \neq \emptyset$. Consider the first time an arc $a \in A$ used in at least one element of \mathcal{A} has been removed from P . At that time $\mathcal{A} \subseteq \mathcal{P}(P)$, so a would not have been removed because $\forall b \neq a \in P \exists T \in \mathcal{A}$ s.t. $a \in T$ and $b \notin T$.

Since initially $|P| = m$ and at every recursive call the cardinality of P is decreased by one, we are sure that the number of recursive calls is bounded by m . At each call the subroutine FINDSG is executed up to m^2 times. Thus, if FINDSG has computational complexity $O(\gamma)$, the worst case complexity of the overall algorithm is $O(m^3\gamma)$. In conclusion, if we are provided a polynomial algorithm to solve the subproblem, we can solve ASP1 in polynomial time.

4. Special cases and applications.

Definition 4 holds for a generic family of subsets. In real application we usually have to deal with a family \mathcal{S} characterized by specific properties. By exploiting them, we can describe \mathcal{S} implicitly and, in some cases, obtain polynomial procedures to solve FINDSG even if the cardinality of \mathcal{S} is combinatorial in the size of the graph.

We now analyze some families of subgraphs that lead to interesting applica-

tions.

Secret Santa Problem.

If the family \mathcal{S} is the set of all *Vertex Disjoint Circuit Covers* (VDCCs), we obtain the Secret Santa Problem described in [2].

The basic concept of the Secret Santa game is simple. All of the participants' names are placed into a hat. Each person then chooses one name from the box, but doesn't tell anyone which name was picked. He/she is now responsible for buying a gift for the person selected. When the Secret Santa wraps his/her gift, he/she should label it with the recipient's name but doesn't indicate whom the present is from. All the gifts are then placed in a general area for opening at a designated time.

Additional constraints are considered in the definition of the problem: it may be required that self-gifts and gifts between certain pairs of participants should be avoided. The problem can be modeled with a digraph, where vertices represent the participants and arcs the possibility of a participant giving a gift to another participant. We want to determine if the topology of the graph allows an anonymous exchange of gifts, that is nobody can discover who made a gift to whom, knowing the graph and the receiver of his gift.

The problem can be formulated as an ASP1 where \mathcal{S} is the family of all the VDCCs of the graph.

Definition 7. A *Vertex Disjoint Circuit Cover* (VDCC) for $G = (V, A)$ is a subset $S \subseteq A$ of arcs of G such that: (a) for each $v \in V$ there is a unique $u \in V$, called the predecessor of v and denoted by $\pi_S(v)$, such that $(u, v) \in S$; (b) for each $v \in V$ there is a unique $u \in V$, called the successor of v and denoted by $\sigma_S(v)$, such that $(v, u) \in S$. We denote by \mathcal{C} the set of all VDCCs in G .

In this case $\text{FINDSG}(G, \mathcal{S}, P, \{(i, j)\})$ requires to find a VDCC with restrictions on the arcs can be used. As shown in [2] it can be done in $O(n^{\frac{1}{2}}m)$ by solving an assignment problem on a bipartite graph $B = (U_1, U_2, A')$, where $U_1 = U_2 = V \setminus \{i, j\}$ and $A' = P$.

Anonymous routing.

In many contexts it is desirable to hide the identity of the users involved in a transaction on a public telecommunication network. According to the specific application, we may be interested in:

- *sender anonymity* to a node, to the receiver or to a global attacker;
- *receiver anonymity* to any node, to the sender or to a global attacker;
- *sender-receiver unlinkability* to any node or a global attacker. This means that a node may know that A sent a message and B received one, but not that A's message was actually received by B.

Several protocols to provide anonymous routing features has been proposed in the literature ([1; 4; 3]). Using these protocols every node traversed by a message has only a partial knowledge on the path the message is being routed on. Typically a node knows only the next step, or the next and the previous one.

Attacks against these protocols are usually based on traffic analysis. Thus, if the topology of the network contains “forced paths”, they can leak information to an attacker who is monitoring the traffic.

Some protocols, like Onion Routing, require the topology of the network to be known to every participant. Every time a node leave or a new node joins the protocol, the topology of the network changes. Therefore it may be useful to check the network against the presence of “forced paths”. This can be done by solving, for each pair of nodes (s, t) of the network, an instance of ASP1 where G is the graph representing the network and S is the family of all paths of length at least 2 between the two nodes. We exclude paths involving one arc because they naturally fail in providing anonymity. The subproblem $\text{FINDSG}(G, S, P \setminus \{b\}, \{(i, j)\})$ requires, in this case, to find two paths: one from s to i and one from j to t . It can be done in $O(n + m)$ using a graph traversing algorithm.

Anonymous routing protocols usually generate pseudo-random path in order to maximize the level of anonymity provided and the robustness against traffic analysis attacks. This introduces delays in the transaction (e.g. in onion routing we have to apply a layer of cryptography for each node in the path) that cannot be tolerated in certain application, i.e. when the content of the message is part of an audio or video stream, or in financial market transactions. In these situations we may want to give up some anonymity in exchange for performances. We may, for example, force the routing protocol to choose *quasi* shortest paths, instead of random ones. Again, we would like the topology of the graph to allow them to be anonymous. We can check this property in a way similar to what we have done in the previous case, but this time the family S will contain only the $s - t$ paths S whose length is not greater than α times the length of the shortest path from s to t , where $\alpha \geq 1$ is a given parameter.

References

- [1] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–88, 1981.

- [2] Leo Liberti and Franco Raimondi. The secret santa problem. In R. Fleischer and J. Xu, editors, *AAIM08 Proceedings*, pages 271–279. Springer, 2008.
- [3] M. Reiter and A. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
- [4] P F Syverson, D M Goldschlag, and M G Reed. Anonymous connections and onion routing. *In IEEE Symposium on Security and Privacy*, 1997.

Graph Theory III

Lecture Hall B

Thu 4, 08:45–10:15

The number of excellent discrete Morse functions on graphs[★]

R. Ayala,^a D. Fernández-Ternero,^a J.A. Vilches^a

^a*Dpto. de Geometría y Topología, Universidad de Sevilla, 41080, Sevilla, SPAIN*
{rdayala, desamfer, vilches}@us.es

Abstract

In [5] the number of non-homologically equivalent excellent discrete Morse functions defined on S^1 was obtained in the differentiable setting. We carried out the analogous study in the discrete setting for some kind of graphs, including S^1 , in [2]. In this paper we complete this study counting excellent discrete Morse functions defined on any infinite locally finite graph.

Key words: infinite locally finite graph, critical simplex, gradient vector field, gradient path, excellent discrete Morse function

1. Introduction

Through all this paper, we only consider infinite graphs which are locally finite. Given such a graph G , a **bridge** is an edge whose deletion increases the number of connected components of G . A graph is said to be **bridgeless** if it contains no bridges. We consider non trivial connected bridgeless graphs, that is, connected bridgeless graphs not consisting of a unique vertex.

Let B the set of all bridges of G . The **bridge components** of G are the connected components of $G - B$. For other topics of graph theory we follow [4].

We introduce here the basic notions of Discrete Morse theory [3]. A **discrete Morse function** is a function $f : G \rightarrow \mathbb{R}$ such that, for any p -simplex $\sigma \in G$:

$$(M1) \quad \text{card}\{\tau^{(p+1)} > \sigma / f(\tau) \leq f(\sigma)\} \leq 1.$$

[★] The authors are partially supported by the Plan Nacional de Investigación 2.007, Project MTM2007-65726, España, 2008.

(M2) $\text{card}\{v^{(p-1)} < \sigma/f(v) \geq f(\sigma)\} \leq 1$.

A p -simplex $\sigma \in G$ is said to be a **critical simplex** with respect to f if:

(C1) $\text{card}\{\tau^{(p+1)} > \sigma/f(\tau) \leq f(\sigma)\} = 0$.

(C2) $\text{card}\{v^{(p-1)} < \sigma/f(v) \geq f(\sigma)\} = 0$.

A value of a discrete Morse function on a critical simplex is called **critical value**.

A **ray** is a sequence of simplices:

$$v_0, e_0, v_1, e_1, \dots, v_r, e_r, v_{r+1} \dots$$

If there is a discrete Morse function f defined on G , a **decreasing ray** is a ray verifying that:

$$f(v_0) \geq f(e_0) > f(v_1) \geq f(e_1) > \dots \geq f(e_r) > f(v_{r+1}) \geq \dots$$

A **critical element** of f on G is either a critical simplex or a decreasing ray.

Given $c \in \mathbb{R}$ the **level subcomplex** $G(c)$ is the subcomplex of G consisting of all simplices τ with $f(\tau) \leq c$, as well as all of their faces, that is,

$$G(c) = \bigcup_{f(\tau) \leq c} \bigcup_{\sigma \leq \tau} \sigma$$

Theorem 1.1. [1] Let G be a graph and let f be a discrete Morse function defined on G such that the numbers $m_i(f)$ of critical i -simplices of f with $i = 0, 1$ are finite and f has no decreasing rays. Then:

- (i) $m_0(f) \geq b_0$ and $m_1(f) \geq b_1$, where b_i denotes the i -th Betti number of G with $i = 0, 1$.
- (ii) $b_0 - b_1 = m_0(f) - m_1(f)$.

Given a discrete Morse function defined on G , we say that a pair of simplices $(v < e)$ is in the **gradient vector field** induced by f if and only if $f(v) \geq f(e)$.

Given a gradient vector field V on G , a **V -path** is a sequence of simplices

$$\alpha_0^{(p)}, \beta_0^{(p+1)}, \alpha_1^{(p)}, \beta_1^{(p+1)}, \dots, \beta_r^{(p+1)}, \alpha_{r+1}^{(p)}, \dots,$$

such that, for each $i \geq 0$, the pair $(\alpha_i^{(p)} < \beta_i^{(p+1)}) \in V$ and $\beta_i^{(p+1)} > \alpha_{i+1}^{(p)} \neq \alpha_i^{(p)}$.

Given a 0-critical simplex in G , we say that any vertex w of G is **rooted** in v if there exists a finite V -path joining w and v .

Proposition 1.2. [2] Let G be an infinite graph and let f be a discrete Morse function defined on G with no decreasing rays. It holds that:

- (i) Given w any vertex of G , there is a unique 0-critical simplex on which w is rooted.
- (ii) Given any 0-critical simplex v , the set of all V -paths rooted in it is a tree called **the tree rooted in v** and denoted by T_v .
- (iii) Any two of such rooted trees are disjoint.

Theorem 1.3. [2] Under the above definitions and notations, the forest F consisting of all rooted trees in G can be obtained by removing all critical edges of f on G .

2. The number of excellent discrete Morse functions on a graph

A discrete Morse function defined on a graph G is called **excellent** if all its critical values are different.

Two excellent discrete Morse functions f and g defined on a graph G with critical values $a_0 < a_1 < \dots < a_{m-1}$ and $c_0 < c_1 < \dots < c_{m-1}$ respectively will be called **homologically equivalent** if for all $i = 0, \dots, m-1$ the level subcomplexes $G(a_i)$ and $G(c_i)$ have the same Betti numbers.

Let f be an excellent discrete Morse function defined on G with m critical simplices and critical values a_0, \dots, a_{m-1} . We denote the level subcomplexes $G(a_i)$ by G_i for all $i = 0, \dots, m-1$. The **homological sequences** of f are the two sequences $B_0, B_1 : \{0, 1, \dots, m-1\} \rightarrow \mathbb{N}$ containing the homological information of the level subcomplexes G_0, \dots, G_{m-1} , that is, $B_p(i) = b_p(G_i) = \dim(H_p(G_i))$ for each $i = 0, \dots, m-1$ and $p = 0, 1$.

Notice that the homological sequences of f satisfy:

$$B_0(0) = B_0(m-1) = b_0 = 1, \quad B_0(i) > 0, \quad |B_0(i+1) - B_0(i)| = 0 \text{ or } 1;$$

$$B_1(0) = 0, \quad B_1(m-1) = b_1, \quad B_1(i) \geq 0, \quad B_1(i+1) - B_1(i) = 0 \text{ or } 1.$$

Lemma 2.1. [2] For each $i = 0, 1, \dots, m-2$ it holds one and only one of the following identities:

$$(H1) \quad B_0(i) = B_0(i+1).$$

$$(H2) \quad B_1(i) = B_1(i+1).$$

Note that identity (H1) reveals us the creation of a new 1-cycle of G on this process and therefore it holds this identity for exactly b_1 values of i . If we remove $B_0(i+1)$ for these values of i in the sequence B_0 , we obtain a walk in $\mathbb{Z}_{>0}$ starting and ending at 1, with even length $2k$ and steps of size ± 1 . The number of such walks is the k -th Catalan number $C_k = \frac{1}{k+1} \binom{2k}{k}$.

Lemma 2.2. If G is a connected graph with at least one bridge and $b_1 < +\infty$, then $G = P_1 \cup P_2 \cup \dots \cup P_p \cup F$, where P_1, \dots, P_p are the non trivial bridge components of G , F is a forest and every tree in F intersects each P_i in at most one vertex. Moreover, if G is infinite, then F has at least an infinite tree.

Theorem 2.3. Under the notations of the above Lemma, the number of homology equivalence classes of excellent discrete Morse functions with $m = b_0 + b_1 + 2k$ critical elements on a graph G with $b_1 < +\infty$ is:

- (i) $C_k \binom{m-2}{2k}$ if G is a non trivial bridgeless graph.
- (ii) $C_k \binom{m-1}{2k}$ if G is infinite or has at least one vertex with degree one.
- (iii) $\sum_{j=0}^{k-1} C_j C_{k-j-1} \left(\binom{m-1}{2k} - \binom{2j+b_{12}+1}{2j} \binom{2(k-j)+b_{11}-2}{2(k-j)-1} \right)$ if G is finite, G has at least one bridge and the degree of any vertex of G is greater than one, where $b_{11} = \min\{b_1(P_i) : F \cap P_i \text{ is a unique vertex}\}$ and $b_{12} = b_1 - b_{11}$.

References

- [1] R. Ayala, L.M. Fernández and J.A. Vilches, Discrete Morse inequalities on infinite graphs, *The Electronic Journal of Combinatorics* **16** (1) (2009) R38, 11 pp. (electronic).
- [2] R. Ayala, L.M. Fernández, D. Fernández-Tertero and J.A. Vilches, Discrete Morse Theory on graphs, to appear in *Topology Appl.*
- [3] R. Forman, A user's guide to discrete Morse theory, *Sém. Lothar. Combin.* **48** (2002), Art. B48c, 35 pp. (electronic).
- [4] J. L. Gross, J. Yellen (ed.), Handbook of graph theory. Discrete Mathematics and its Applications. CRC Press, 2004.
- [5] L. I. Nicolaescu, Counting Morse functions of the 2-sphere, *Compositio Math.* **144** (2008), pp. 1081–1106.

Partial characterizations of circle graphs

Flavia Bonomo,^{a,b,1,2} Guillermo Durán,^{a,c,d,1,3}
Luciano N. Grippo,^{a,b,e,1} Martín D. Safe^{a,b,1}

^aCONICET, Argentina

^bDepartamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina

{fbonomo, mdsafe}@dc.uba.ar

^cDepartamento de Matemática, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina

gduran@dm.uba.ar

^dDepartamento de Ingeniería Industrial, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, Chile

^eInstituto de Ciencias, Universidad Nacional de General Sarmiento, Argentina

lgrippo@ungs.edu.ar

Key words: circle graphs, Helly circle graphs, structural characterizations.

1. Introduction

Circle graphs were introduced in [7] to solve a problem of queues and stacks posed by Knuth in [11]. A graph $G = (V, E)$ is a *circle graph* if it is the intersection graph of a family $L = \{C_v\}_{v \in V}$ of chords on a circle (i.e., for each $v, w \in V$, $vw \in E$ if and only if $v \neq w$ and $C_v \cap C_w \neq \emptyset$). L is called a *circle model* of G . In [1; 8; 12], recognition algorithms based on the fact that circle graphs are closed by split composition (cf. [4]) were presented. In [2], Bouchet proved that a graph G is a circle graph if and only if each graph that is locally equivalent to G contains none of 3 prescribed forbidden induced subgraphs. However, there are not known characterizations of circle graphs by forbidden induced subgraph that do not involve the notion of local equivalence. We present some results in this direction, providing forbidden induced subgraph characterizations of circle graphs

¹ Partially supported by ANPCyT PICT-2007-00518 and UBACyT X069 (Arg.).

² Partially supported by ANPCyT PICT-2007-00533 and UBACyT X606 (Arg.).

³ Partially supported by FONDECYT Grant 1080286 and Millennium Science Institute “Complex Engineering Systems” (Chile).

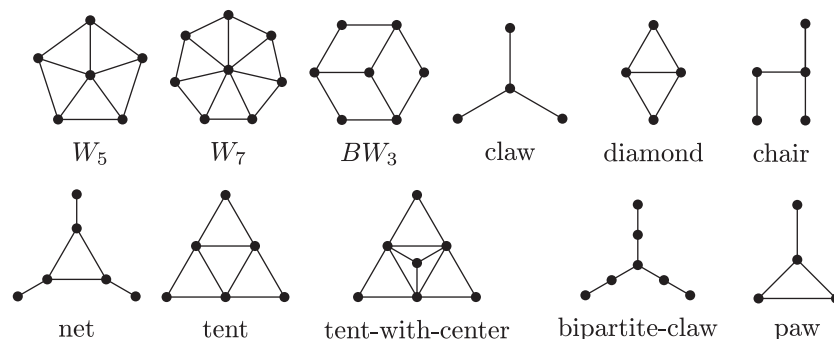


Fig. 1. Some small graphs

restricted to graphs that belong to one of the following graph classes: linear domino, $\{\text{chair, triangle}\}$ -free, P_4 -sparse, and tree-cographs.

The concept of *Helly circle graph* is due to Durán [6]. A graph belongs to this class if it has a circle model whose chords are all different and satisfy the Helly property. In [6], it is conjectured that a circle graph is a Helly circle graph if and only if it is a diamond-free graph. This conjecture was recently affirmatively settled affirmatively in [5]. Therefore, the Helly circle graph recognition problem is solvable in polynomial time. Nevertheless, to best of our knowledge, there is no characterization for the whole class of Helly circle graphs by forbidden induced subgraphs. In this work we completely characterize *unit Helly circle* graphs, which are those having a model whose chords have all the same length, are all different, and satisfy the Helly property.

2. Characterizations

The *local complement* of a graph $G = (V, E)$ with respect to a vertex $u \in V$ is the graph $G * u$ that arises from G by replacing the induced subgraph $G[N(u)]$ by its complement. Two graphs G and H are *locally equivalent* if and only if G arises from H by a sequence of local complementations.

Theorem 1. ([2]) Let G be a graph. Then, G is a circle graph if and only if no graph locally equivalent to G contains W_5 , W_7 or BW_3 as induced subgraph.

As a consequence, we can prove the following result.

Theorem 2. Let G be a graph. If G is not a circle graph, then any graph H that arises from G by edge subdivisions is not a circle graph.

Some small graphs to be referred in the sequel are depicted in Figure 1. A *triangle* is a complete with 3 vertices. A P_4 is a chordless path on 4 vertices.

A graph G is *domino* if all its vertices belong to at most two cliques. If each of its edges belongs to at most one clique, then G is a *linear domino graph*. Linear domino graphs coincide with $\{\text{claw,diamond}\}$ -free graphs [10]. A *prism* is a graph that consists of two disjoint triangles $\{a_1, a_2, a_3\}$ and $\{b_1, b_2, b_3\}$ linked by three vertex-disjoint paths P_1, P_2, P_3 , whose internal vertices have degree two and where P_i links a_i and b_i for $i = 1, 2, 3$. The graph $\overline{C_6}$ is a prism where each path has just one edge. By Theorem 1, $\overline{C_6}$ is not a circle graph. Besides, since every prism arises from $\overline{C_6}$ by edge subdivision, Theorem 2 implies that prisms are not circle graphs.

Theorem 3. Let G be a linear domino graph. Then, G is a circle graph if and only if G contains no induced prism.

The proof relies on the split decomposition of a graph into stars, completes and prime graphs (cf. [4]) and the fact that circle graphs are closed by split composition [1; 8].

Chudnovsky and Kapadia gave a polynomial-time algorithm to decide if a graph contains a theta or a prism [3] (a theta is a graph arising from $K_{2,3}$ by edge subdivision). Theorem 3 and the existence of a polynomial-time algorithm for recognizing circle graphs imply an alternative polynomial-time algorithm to find a theta or a prism in linear domino graphs, because a domino graph cannot contain a theta.

Next, we characterize those $\{\text{chair,triangle}\}$ -free graphs that are circle graphs.

Theorem 4. Let G be a $\{\text{chair,triangle}\}$ -free graph. Then, G is a circle graph if and only if G contains no induced BW_3 .

Cographs are the graphs with no chordless paths on 4 vertices; i.e., P_4 -free. It is well known that cographs are circle graphs. P_4 -sparse graphs are a natural generalization of cographs. Hoàng [9] defined a graph to be P_4 -sparse if every five vertices induce at most one P_4 . For any graph G , let G^+ denote the graph that arises from G by adding a universal vertex.

Theorem 5. Let G be a P_4 -sparse graph. Then, G is a circle graph if and only if G contains no induced net^+ , no induced tent^+ and no induced tent-with-center .

Tree-cographs are defined recursively as follows: trees are tree-cographs, the disjoint union of tree-cographs is a tree-cograph, and if H is a tree-cograph, then \overline{H} is a tree-cograph.

Theorem 6. Let G be a tree-cograph. Then, G is a circle graph if and only if G contains no induced $(\text{bipartite-claw})^+$ and no induced $\text{co-(bipartite-claw)}$.

Our last result is a complete characterization of unit Helly circle graphs. Let C_n^* denote the graph that arises from a chordless n -cycle by adding an isolated vertex.

Theorem 7. Let G be a graph. Then the following assertions are equivalent: (i) G is a unit Helly circle graph; (ii) G contains no induced paw, no induced diamond and no induced C_n^* for any $n \geq 3$; (iii) G is a chordless cycle, a complete graph, or a disjoint union of chordless paths.

The proof is of geometric nature and relies on properties of tangent lines to a circle.

References

- [1] A. Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorica*, **7**, 243–254, 1987.
- [2] A. Bouchet. Circle graphs obstructions. *J. Combin. Theory Ser. B*, **60**, 107–144, 1994.
- [3] M. Chudnovsky and R. Kapadia. Detecting a theta or a prism. *SIAM J. Discrete Math.*, **22**, 1164–1186, 2008.
- [4] W. H. Cunningham. Decomposition of directed graphs. *SIAM J. Alg. Disc. Meth.*, **3**, 214–228, 1982.
- [5] J. Daligault, D. Gonçalves, and M. Rao. Diamond-free circle graphs are Helly circle. Manuscript, 2008.
- [6] G. Durán. Some new results on circle graphs. *Mat. Contemp.*, **25**, 91–106, 2003.
- [7] S. Even and A. Itai. Queues, stacks and graphs. In A. Kohavi and A. Paz, editors, *Theory of Machines and Computations*, pages 71–86. Academic Press, New York, 1971.
- [8] C. Gabor, K. Supowit, and W. Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, **36**, 435–473, 1989.
- [9] C. T. Hoàng. *Perfect graphs*. PhD thesis, School of Computer Science, McGill University, Montreal, 1985.
- [10] T. Kloks, D. Kratsch, and H. Müller. Dominoes. *Lect. Notes Comput. Sci.*, **93**, 106–120, 1995.
- [11] D. E. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, Reading, MA, 1969.
- [12] J. P. Spinrad. Recognition of circle graphs. *J. Algorithms*, **16**, 264–282, 1994.

A Replacement Model for a Scale-Free Property of Cliques

Takeya Shigezumi,^a Yushi Uno,^b Osamu Watanabe^a

^a*Department of Mathematical and Computing Sciences, Tokyo Institute of Technology,
Tokyo 152-8552, Japan*

{Takeya.Shigezumi, watanabe}@is.titech.ac.jp

^b*Department of Mathematics and Information Sciences, Graduate School of Science,
Osaka Prefecture University, Sakai 599-8531, Japan*

uno@mi.s.osakafu-u.ac.jp

Key words: degree, isolated clique size, power-law, replacement model.

1. Introduction

It has been observed that many large graphs (networks) that express some real world relationships possess certain characteristics such as power-law degree distributions, large cluster coefficients, etc. Recently, Uno et al. [1] found another ‘scale-free’ property by investigating a graph from some real network. They observed that the size distributions of ‘isolated cliques’, cliques that can be separated easily from the other part, follows power-law. Furthermore, it keeps this property after contracting every isolated clique to one vertex; that is, the clique structure of the graph has self-similarity. (Though a different type, some self-similarity has been also studied by Song et al. [2].) In this paper we show a way to generate graphs with this recursive clique structure. Our method is to expand an initial graph for a few times so that the obtained graph has a recursive clique structure. One important point is that the basic characteristics of the initial graph are kept through this expansion process.

Preliminaries: Consider any (sufficiently large) graph G and let us fix it in the following explanation. We use V and E (or more specifically $V(G)$ and $E(G)$) to denote its set of *vertices* and *edges*, respectively, and let n denote the number of vertices in G . For a vertex v , let $N(v)$ denote the set of its neighbor vertices, namely, $N(v) = \{u \mid \{u, v\} \in E\}$, and for any $U \subset V$, we also use $N(U)$ to denote $\bigcup_{v \in U} N(v)$. The *degree* of v is defined as $\deg(v) \triangleq |N(v)|$. By “subgraph” of G , we imply its *induced subgraph*.

Isolated Clique and Contraction: A subgraph of G is called *clique* if every pair of vertices in this subgraph is adjacent. A clique C is called *c-isolated* if the number of *outgoing* edges from $V(C)$ to $V \setminus V(C)$ is less than or equal to $c|V(C)|$. Although finding large cliques in the graph is intractable, finding isolated cliques is not so hard. Furthermore, 1-isolated clique can be enumerated in linear time [3], and it is

investigated in [1]. Note that very few overlaps occur among 1-isolated cliques and they are easy to be separated. Throughout this paper, we consider 1-isolated clique and we simply call them *isolated clique*. We consider a process of *contracting an isolated clique* of G into one vertex. We use $\mathcal{C}(G)$ to denote a graph obtained from G by contracting all isolated cliques in G .

Power-law and Scale-free Property: The scale-freeness is considered as one of the basic properties characterizing real world large graphs. We say that G is ‘scale-free’ if its degree distribution follows power-law, i.e., a distribution proportional to $k^{-\gamma}$ for some constant γ . Let us make these notions more precise for our discussion. The *degree distribution* of G is a sequence $\{n_k\}_{k \geq 1}$, where n_k is the number of vertices in G with degree k . Then we say that G ’s degree distribution follows a *power-law* if $n_k = \Theta(k^{-\gamma})$ for some γ , that is, there are some constants c_1 and c_2 such that $c_1 k^{-\gamma} \leq n_k \leq c_2 k^{-\gamma}$ for all $k \geq 1$. The parameter γ is called a *power-law exponent*. In this paper we extend this notion to isolated clique size distributions. The *isolated clique size distribution* of G is a sequence $\{m_s\}_{s \geq 1}$, where m_s is the number of isolated cliques of s vertices. We say that G ’s isolated clique size follows *power-law* if the sequence $\{m_s\}_{s \geq 1}$ satisfies $m_s = \Theta(s^{-\gamma})$ for some γ .

Remark on constants. It does not make sense for discussing the above properties for any fixed finite graph G . Thus, in this paper, we will consider a family of graphs consisting of infinite number of graphs defined in a certain way and discuss power-law properties with constants c_1 and c_2 that are independent from k and the choice of a graph in the family. Thus, when claiming for example that G ’s degree distribution follows a power-law with some exponent γ , we formally imply that its degree sequence $\{n_k\}_{k \geq 1}$ satisfies $n_k = \Theta(k^{-\gamma})$ under some fixed constants c_1 and c_2 for all graphs in our assumed graph family.

Cluster Coefficient: Another basic property is on a cluster coefficient, a way to measure the density of triangles in a given graph. For any vertex v , the following ratio is called the *cluster coefficient* of v :

$$CC(v) = |\{ \{u, w\} \in E \mid u, w \in N(v) \}| / \binom{\deg(v)}{2}.$$

Then let $CC(U)$ denote the average cluster coefficient of all vertices in U (i.e., the arithmetic mean of $CC(v)$ of all vertices v in U). We say that G has a *large cluster coefficient* if $CC(V(G))$ is larger than some constant. (Here again the remark above is applicable. By ‘constant’ we mean some constant that works for all graphs in our assumed graph family.) Note that the cluster coefficient is the probability that any pair of two neighbors of some vertex have an edge (when such a pair is chosen uniformly at random); thus, on a graph G with a large cluster coefficient, it is likely that vertices adjacent to some vertex are also adjacent.

2. Model

We describe our model, that is, a way of defining graphs that possess self-similarity as observed in [1]. We give a method for expanding a graph randomly; it is designed so that a graph with our desired self-similarity is obtained with high probability by applying this expansion some number of times to a given initial graph that is defined by some other model. Our expansion method is simple. For each degree $k \geq 3$, we select each vertex v of degree k independently with probability p_k , and then replace it with a clique of size k , where each edge to a vertex u in $N(v)$ is replaced with an edge between u and one vertex v_u in the clique (see Figure 1). We introduce one parameter p and define the probability p_k as follows:

$$p_k = \begin{cases} p/(k-1), & \text{if } k \geq 3, \text{ and} \\ 0, & \text{otherwise.} \end{cases}$$

For any graph G , let $\mathcal{E}(G)$ denote a graph obtained by applying this random expansion. Note that $\mathcal{E}(G)$ is a random variable. We consider graphs obtained by applying this random expansion t times. In the following analysis, we fix one initial graph that is taken from a certain graph family, and let G^0 denote it. For example, we may assume that G^0 is generated by some known scale-free graph model. Then for a given t , let G^t denote a graph obtained by applying $\mathcal{E}(\cdot)$ for t times to G^0 . In order to simplify our discussion, we assume in this paper that G^0 has no isolated cliques. Throughout this paper, we assume that t is sufficiently large, but it is still regarded as a constant. (Some results can be generalized for the non-constant case.)

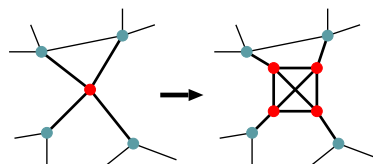


Fig. 1. An expansion of a vertex of degree four.

3. Analysis

Fix G^0 and consider randomly generated graphs G^i , $1 \leq i \leq t$. We first show that two basic properties characterizing real world large graphs are inherited in G^1, \dots, G^t . That is, if G^0 has a power-law degree distribution and/or has a large cluster coefficient, then so does G^t with high probability.

First consider the degree distribution of G^t . For each $k \geq 1$, let n_k and N_k^t denote the number of degree k vertices in G^0 and G^t , respectively. Then for any $k \geq 3$, it is easy to show the following. (Note that from our setting $p_1 = p_2 = 0$, the number of vertices of degree 1 or 2 does not change by expansion.)

Theorem 3.1. $E[N_k^t] = (1 + p)^t n_k$, and for any δ ($0 < \delta < 1$), we have

$$\Pr[|N_k^t - \mathbb{E}[N_k^t]| > \delta \cdot \mathbb{E}[N_k^t]] < 2te^{-\frac{p\delta^2 n_k}{12t^2(k-1)}}.$$

Thus, if n_k is large enough, we may assume that N_k^t 's concentration around its expectation is high. Therefore, if n_k follows power-law with exponent γ , then with high probability N_k^t also follows the power-law with the same exponent.

Next consider cluster coefficients. We analyze a cluster coefficient for vertices of each degree k ; that is, for each $k \geq 1$, we let ΣCC_k^i denote the sum of the cluster coefficients of all degree k vertices in G^i , and we analyze this quantity. Then we have the following bound for $k \geq 5$. (For small degree $k \leq 4$, a similar bound can be shown by detail case analysis.)

Theorem 3.2. $\mathbb{E}[\Sigma CC_k^t] > \alpha^t \Sigma CC_k^0 + \frac{1}{5}((1+p)^t - \alpha^t)n_k$, where $\alpha = 1 - \frac{p}{k-1}$.

From these bounds, we can derive a bound for the degree-wise cluster coefficient by simply dividing ΣCC_k^t by N_k^t . Recall here that $\mathbb{E}[N_k^t] = (1+p)^t n_k$ and that N_k^t is close to this expectation with high probability if we can assume that n_k is large enough. Hence, for example, we can show for any $k \geq 5$ that $\mathbb{E}[\text{CC}(V_k^t)] > \beta^t \text{CC}(V_k^0) + (1 - \beta^t)/5$, where V_k^i is the set of vertices of degree k in G^i and $\beta = 1 - (p/(1+p))(k/(k-1)) \simeq 1 - p$. Notice that the above bound is either close to $\text{CC}(V_k^0)$ or larger than some constant, say, $1/5$. Hence if the original degree-wise cluster coefficient $\text{CC}(V_k^0)$ is large, then $\text{CC}(V_k^t)$ is also large (provided n_k is large). On the other hand, if n_k is small for supporting some reasonable concentration on N_k^t , then the influence of vertices in V_k in $\text{CC}(V(G^t))$ can be ignored. Therefore, we can conclude that $\mathbb{E}[\text{CC}(V(G^t))]$ is either close to $\text{CC}(V(G^0))$ or larger than some constant.

Next consider an isolated clique size distribution. Let M_s^i be the number of isolated cliques of size s in G^i . Again this is a random variable except for M_s^0 , which was assumed to be 0 (i.e., no isolated cliques in G^0). Then for any $s \geq 3$, we can show the following bounds.

Theorem 3.3. $(1+p)^{t-1}pn_s/s < \mathbb{E}[M_s^t] < (1+p)^t n_s/s$.

Thus (regarding p and t as constants) we can conclude that if $\{n_k\}_{k \geq 1}$ follows power-law with some exponent γ , then on average $\{M_s^t\}_{s \geq 1}$ follows power-law with exponent $\gamma + 1$. That is, G^t has a large cluster coefficient if G^0 's degree distribution follows power-law. A concentration result similar to the one for N_k^t can be also shown.

Finally consider the self-similarity or recursive structure of G^t . We would like to see, e.g., whether G^t keeps a similar isolated clique size distribution after contracting it several times. For any $j \geq 0$, let H^j be the graph obtained from G^t by applying the contraction $\mathcal{C}(\cdot)$ for j times. That is, $H^0 = G^t$, $H^1 = \mathcal{C}(H^0)$, $H^2 = \mathcal{C}(H^1), \dots$, and so on. It should be noted here that the contraction $\mathcal{C}(\cdot)$ is not the inverse of our expansion $\mathcal{E}(\cdot)$ in general. Thus, $H^1 = G^{t-1}$ does not hold in general, and it is not at all trivial that H^j has a similar clique size distribution.

Nevertheless, for the number C_s^j of cliques of size s in H^j , we can show that the following bound for any $s \geq 3$, which supports that H^j keeps a similar clique distribution on s .

Theorem 3.4. $E[C_s^j] > (1 - e^{-p})^j E[M_s^{t-j}] > (1 - e^{-p})^j (1 + p)^{t-j-1} p n_s / s$.

References

- [1] Y. Uno, T. Kiyotani and F. Oguri, Investigating web structure by cliques and stars, RIMS Technical Report, to appear.
- [2] C. Song, S. Havlin and H.A. Makse, Self-similarity of complex networks, in *Nature*, 433, 392–395, 2005.
- [3] H. Ito, K. Iwama and T. Osumi, Linear-time enumeration of isolated cliques, in *Proc. ESA*, LNCS 3669, 119–130, 2005.

Graph Theory IV

Lecture Hall A

Thu 4, 10:30–12:30

Combinatorial optimization problems with conflict graphs

Andreas Darmann,^a Ulrich Pferschy,^b Joachim Schauer,^b
Gerhard J. Woeginger^c

^a*University of Graz, Institute of Public Economics, Universitaetsstr. 15, A-8010 Graz, Austria*

andreas.darmann@uni-graz.at

^b*University of Graz, Department of Statistics and Operations Research, Universitaetsstr. 15, A-8010 Graz, Austria*

{pferschy, joachim.schauer}@uni-graz.at

^c*Eindhoven University of Technology, Department of Mathematics and Computer Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

gwoegi@win.tue.nl

Key words: knapsack problem, minimum spanning tree, conflict graph

1. Introduction

Conflict graphs impose disjunctive constraints for pairs of jobs, items, edges or other objects in a combinatorial optimization problem. Equivalently, the feasible domain of the considered problem is restricted to stable sets in the given conflict graph. After reviewing in our presentation results from the literature for bin packing and scheduling problems with conflict graphs, we first consider the classical 0-1 knapsack problem. Adding a conflict graph makes the problem strongly NP-hard but for three special graph classes, namely trees, graphs with bounded treewidth and chordal graphs, we can develop pseudopolynomial algorithms. From these we can easily derive fully polynomial time approximation schemes (FPTAS). Secondly, we study the minimum spanning tree problem and show that the border between polynomially solvable and NP-hard is given by moving from a conflict graph containing only isolated edges to paths of length 2.

2. The Knapsack Problem with Conflict Graphs

For a formal definition of the *knapsack problem with conflict graph* (KCG), let n be the number of items, each of them with profit p_j and weight w_j , $j = 1, \dots, n$, and c the capacity of the knapsack. Let $G = (V, E)$ with $|V| = n$ be a conflict graph, where the vertices uniquely correspond to the items of the knapsack. G is not necessarily connected and may therefore contain isolated vertices.

Then KCG is determined by the following ILP formulation:

$$(KCG) \quad \max \sum_{j=1}^n p_j x_j \quad (2.1)$$

$$\text{s.t.} \quad \sum_{j=1}^n w_j x_j \leq c \quad (2.2)$$

$$x_i + x_j \leq 1 \quad \forall (i, j) \in E \quad (2.3)$$

$$x_j \in \{0, 1\} \quad j = 1, \dots, n. \quad (2.4)$$

From a graph theoretical perspective, KCG can also be seen as a generalization of the independent set problem. For every given instance of the independent set problem we can superimpose an instance of KCG by introducing trivial items for every vertex with profit and weight equal to 1 and capacity $c = n$. Therefore it follows immediately, that KCG for general graphs is strongly NP-hard (cf. [4]) and does not permit pseudo-polynomial algorithms (under $P \neq NP$). Motivated by this complexity status and our main task is to identify graph classes for which we can prove the existence of a pseudo-polynomial time and space algorithm and use them to attain fully polynomial time approximation schemes (FPTAS).

For trees as conflict graphs we introduce a dynamic programming algorithm that solves KCG in $O(nP^2)$ time using $O(\log(n)P + n)$ space where $P = \sum_{i=1}^n p_i$. If we consider any vertex $i \in T$, by the property of trees as conflict graphs, when including i into the knapsack solution, it is not allowed to include the parent vertex p of i as well as any of the k child vertices $c_1 \dots c_k$ of i . Indeed these vertices are the only vertices in T that are in conflict with i . The main idea of our algorithm is to process T in depth-first order starting at some root vertex r .

For graphs of bounded treewidth as conflict graphs (including series-parallel graphs, outerplanar graphs, Halin graphs... ([2])), we derive a dynamic programming algorithm solving KCG with a conflict graph of bounded treewidth k in $O(nP^2)$ time using $O(\log(n)P + n)$ space. For algorithmic purposes, the structure of the decomposition is restricted to four simple configurations corresponding to a *nice tree decomposition*, which can be computed from a tree-decomposition in $O(n)$ time (cf. [3]).

For every chordal graph G (graphs that do not contain induced cycles other than triangles) there exists a clique tree $T = (\mathcal{K}, \mathcal{E})$, where the maximal cliques K of G are vertices of T and for each vertex $v \in G$ all cliques K containing v induce a subtree in T . The basic idea for treating chordal graphs lies in utilizing special separation properties of the clique-tree of a chordal graph along with the fact that from every maximal clique of G at most one vertex can be added to the knapsack solution. The three relevant separation properties can be found with detailed proofs in [1]. Our algorithm can be implemented to run in $O((n + m)P^2)$ time and $O(\min \{m, n \log n\} * P + m)$ space where m denotes the number of edges of G .

All the algorithms mentioned above do admit FPTASs by scaling the profit space in a standard way (see e.g. [5]). In fact, the correctness of this scaling procedure depends only on the cardinality of the solution set, which is trivially bounded by n . The running time complexities of the resulting FPTASs differ from the above bounds in the following way: every occurrence of the factor P is replaced by $\frac{n^2}{\epsilon}$ for the scaled instances and thus the required complexity of an FPTAS is attained.

3. Minimum Spanning Trees with Conflict Graphs (*MSTCG*)

In this part of our presentation we consider an extension of the minimum spanning tree problem. In addition to the well studied problem of finding a minimum spanning tree in an undirected connected graph $G = (V, E)$ with weight function w , there exist incompatibilities for certain pairs of edges. These symmetric conflict relations are represented by means of an undirected *conflict graph* $\bar{G} = (E, \bar{E})$, where every vertex of \bar{G} corresponds uniquely to an edge $e \in E$ and edges $\bar{e} = (i, j) \in \bar{E}$ imply that the two vertices adjacent to \bar{e} cannot occur together in a solution.

First we show that *MSTCG* is already NP-hard for an easy subclass of all possible conflict graphs, namely for conflict graphs \bar{G} consisting of components that are described by three vertices (w.l.o.g. e_1, e_2 and e_3) which are connected by two edges (w.l.o.g. (e_1, e_2) and (e_2, e_3)). We call such graphs \bar{G} 3-ladder. In terms of the underlying graph G this means that in a feasible spanning tree including e_1 the edges e_2 and e_3 are necessarily excluded. Obviously this result implies that *MSTCG* is already NP-hard on paths as conflict graphs. The stated result is then shown by reducing a special variant of 3 – *SAT* to *MSTCG*.

On the other hand we also show that *MSTCG* is easy for disjunctive conflicting pairs of edges (we call them ladder). This is done by showing that the conflicting structure imposed by a ladder is a matroid. Then by the matroid intersection theorem of Edmonds (cf. [7]) by intersecting the above matroid and the graphic matroid, which describes the minimum spanning trees, the desired result follows.

4. Concluding remarks

After considering chordal graphs the natural next step would be the more general class of *perfect graphs*. This question however can be settled by a result due to Milanič and Monnot [6]. There it was shown the exact weighted independent set problem (EWIS) for perfect graphs is strongly NP-complete. But this problem can be reduced to KCG. Furthermore, motivated by the result for minimum spanning trees, it seems worthwhile to consider other classical combinatorial optimization problems that do admit polynomial algorithms and combine them with additional constraints, imposed by a conflict graph.

References

- [1] J. R. S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In A. George, J. R. Gilbert, and J. H. U. Liu, editors, *Graph Theory and Sparse Matrix Computations*, pages 1–29, New York, 1993. Springer.
- [2] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–21, 1993.
- [3] H. L. Bodlaender and A. M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.
- [4] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., Amsterdam, The Netherlands, 2004.
- [5] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.
- [6] M. Milanič and J. Monnot. *Combinatorial Optimization - Theoretical Computer Science : interfaces and Perspectives*, chapter The complexity of the exact weighted independent set problem, pages 393–432. Wiley-ISTE, 2008.
- [7] A. Schrijver. *Combinatorial Optimization, Polyhedra and efficiency*, volume B. Springer, 2003.

On the decomposition of graphs into offensive k -alliances

José M. Sigarreta,^a Ismael G. Yero,^b Sergio Bermudo,^c
Juan A. Rodríguez-Velázquez^b

^a*Faculty of Mathematics, Autonomous University of Guerrero, Carlos E. Adame 5, Col. La Garita, Acapulco, Guerrero, México*

^b*Department of Computer Engineering and Mathematics, Rovira i Virgili University, Av. Països Catalans 26, 43007 Tarragona, Spain*
ismael.gonzalez@urv.cat

^c*Department of Economy, Quantitative Methods and Economic History, Pablo de Olavide University, Carretera de Utrera Km. 1, 41013-Sevilla, Spain*

Abstract

The (global) offensive k -alliance partition number of a graph $\Gamma = (V, E)$, denoted by $(\psi_k^{go}(\Gamma)) \psi_k^o(\Gamma)$, is defined to be the maximum number of sets in a partition of V such that each set is an offensive (a global offensive) k -alliance. We obtain tight bounds on $\psi_k^o(\Gamma)$ and $\psi_k^{go}(\Gamma)$ in terms of several parameter of the graph. As a consequence of the study we show the close relationships that exist among the chromatic number of Γ and $\psi_0^{go}(\Gamma)$. Moreover, we study the particular case of partitioning the vertex set of the cartesian product of graphs into (global) offensive k -alliances.

Key words: Offensive alliances, chromatic number, cartesian product of graphs

1. Introduction

Since (defensive, offensive and powerful) alliances in graph were first introduced by P. Kristiansen, S. M. Hedetniemi and S. T. Hedetniemi [5], several authors have studied their mathematical properties [1; 2; 3; 4; 5; 6; 7]. We focus our attention in the problem of partitioning the vertex set of a graph into (global) offensive k -alliances. This problem have been previously studied, for the case of defensive k -alliances, by K. H. Shafique and R. D. Dutton [6; 7] and the particular case $k = -1$ have been studied by L. Eroh and R. Gera [2; 3] and by T. W. Haynes and J. A. Lachniet [4]. We begin by stating the terminology used. Throughout this article, $\Gamma = (V, E)$ denotes a simple graph of order $|V| = n$ and size

$|E| = m$. We denote the degree of a vertex $v \in V$ by $\delta(v)$, the minimum degree by δ and the maximum degree by Δ . For a nonempty set $X \subseteq V$, and a vertex $v \in V$, $N_X(v)$ denotes the set of neighbors v has in X and the degree of v in X will be denoted by $\delta_X(v) = |N_X(v)|$. The complement of the set S in V will be denoted by \bar{S} , moreover, the boundary of S is defined as $\partial(S) := \bigcup_{v \in S} N_{\bar{S}}(v)$. For $k \in \{2 - \Delta, \dots, \Delta\}$, a nonempty set $S \subseteq V$ is an *offensive k -alliance* in Γ if $\delta_S(v) \geq \delta_{\bar{S}}(v) + k$, $\forall v \in \partial(S)$. An offensive k -alliance S is called *global* if it is a dominating set. The (global) *offensive k -alliance number* of Γ , denoted by $(\gamma_k^o(\Gamma)) a_k^o(\Gamma)$, is defined as the minimum cardinality of any (global) offensive k -alliance in Γ . We denote by $(\psi_k^{go}(\Gamma)) \psi_k^o(\Gamma)$ the maximum number of sets in a partition of V such that each set is an offensive k -alliance. Notice that if every vertex of Γ has even degree and k is odd, then every offensive k -alliance in Γ is an offensive $(k + 1)$ -alliance and vice versa. Hence, in such a case, $a_k^o(\Gamma) = a_{k+1}^o(\Gamma)$, $\gamma_k^o(\Gamma) = \gamma_{k+1}^o(\Gamma)$, $\psi_k^o(\Gamma) = \psi_{k+1}^o(\Gamma)$ and $\psi_k^{go}(\Gamma) = \psi_{k+1}^{go}(\Gamma)$. Analogously, if every vertex of Γ has odd degree and k is even, then every offensive k -alliance in Γ is an offensive $(k + 1)$ -alliance and vice versa. Hence, in such a case, $a_k^o(\Gamma) = a_{k+1}^o(\Gamma)$, $\gamma_k^o(\Gamma) = \gamma_{k+1}^o(\Gamma)$, $\psi_k^o(\Gamma) = \psi_{k+1}^o(\Gamma)$ and $\psi_k^{go}(\Gamma) = \psi_{k+1}^{go}(\Gamma)$. We say that a graph Γ is partitionable into (global) offensive k -alliances if $(\psi_k^{go}(\Gamma) \geq 2) \psi_k^o(\Gamma) \geq 2$.

2. Results

Proposition 1. For any graph Γ without isolated vertices, there exists $k \in \{0, \dots, \delta\}$ such that Γ is partitionable into global offensive k -alliances.

Corollary 2. Any graph without isolated vertices is partitionable into global offensive 0-alliances.

Theorem 3. If a graph is partitionable into $r \geq 3$ global offensive k -alliances, then $k \leq 3 - r$.

From Theorem 3 we have that if a graph is partitionable into $r \geq 3$ global offensive k -alliances, then $k \leq 0$, so we obtain the following interesting consequence.

Corollary 4. If Γ is partitionable into global offensive k -alliances for $k \geq 1$, then $\psi_k^{go}(\Gamma) = 2$.

From Corollary 2 we have that any graph without isolated vertices is partitionable into global offensive 0-alliances. Therefore, the above result leads to the following consequence.

Corollary 5. For any graph without isolated vertices, $2 \leq \psi_0^{go}(\Gamma) \leq 3$.

An example of graph where $\psi_0^{go}(\Gamma) = 2$ is the complete graph and an example of graph where $\psi_0^{go}(\Gamma) = 3$ is the cycle graph C_{3t} , $t \geq 1$.

Now we are going to show the relationship that exists among the chromatic number of Γ , $\chi(\Gamma)$, and $\psi_0^{go}(\Gamma)$.

Theorem 6. Any set belonging to a partition of a graph into $r \geq 3$ global offensive k -alliances is a $(-k)$ -dependent set.

Corollary 7. If $\psi_0^{go}(\Gamma) = 3$, then $\chi(\Gamma) \leq 3$.

A trivial example where $\psi_0^{go}(\Gamma) = 3 = \chi(\Gamma)$ is the cycle graph $\Gamma = C_3$. In the case of the cycle graph $\Gamma = C_6$ it is satisfied $\psi_0^{go}(\Gamma) = 3$ and $\chi(\Gamma) = 2$.

Remark 2.1. If Γ is a non bipartite graph and $\psi_0^{go}(\Gamma) = 3$, then $\chi(\Gamma) = 3$.

An example of graph where $\chi(\Gamma) > 3$ and $\psi_0^{go}(\Gamma) = 2$ is the complete graph $\Gamma = K_n$, with $n \geq 4$.

Corollary 8. For any graph Γ without isolated vertices and chromatic number greater than 3, $\psi_0^{go}(\Gamma) = 2$.

Theorem 9. For any graph Γ without isolated vertices containing a vertex of odd degree, $\psi_0^{go}(\Gamma) = 2$.

Theorem 10. If a graph Γ is partitionable into global offensive k -alliances, then $\psi_k^{go}(\Gamma) \leq \left\lfloor \frac{2m-n(k-4)}{2n} \right\rfloor$.

The above bound is attained, for instance, for the cycle graph C_{3t} , where $\psi_0^{go}(C_{3t}) = 3$.

Theorem 11. Let $C_{(r,k)}^{go}(\Gamma)$ be the minimum number of edges having its endpoints in different sets of a partition of Γ into $r \geq 2$ global offensive k -alliances, then $C_{(r,k)}^{go}(\Gamma) \geq \left\lceil \frac{(r-1)(2m+nk)}{4} \right\rceil$. Moreover, if $r \geq 3$, then $C_{(r,k)}^{go}(\Gamma) \leq \left\lfloor \frac{(r-1)(2m-nk)}{4(r-2)} \right\rfloor$.

From the above result we have that if $\psi_k^{go}(\Gamma) \geq 3$, then $\psi_k^{go}(\Gamma) \leq \left\lfloor \frac{6m+nk}{2m+nk} \right\rfloor$. Notice also that, for $k \leq \delta$, $2 \leq \left\lfloor \frac{6m+nk}{2m+nk} \right\rfloor$, so we obtain the following bound.

Corollary 12. For any graph Γ of order n and size m , $\psi_k^{go}(\Gamma) \leq \left\lfloor \frac{6m+nk}{2m+nk} \right\rfloor$.

The above bound is attained, for instance, for the cycle graph $\Gamma = C_{3t}$, where $\psi_0^{go}(\Gamma) = 3$.

Theorem 13. [1] Let $\Gamma_i = (V_i, E_i)$ be a graph of minimum degree δ_i and maximum degree Δ_i , $i \in \{1, 2\}$. If S_i is an offensive k_i -alliance in Γ_i , $i \in \{1, 2\}$, then, for $k = \min\{k_2 - \Delta_1, k_1 - \Delta_2\}$, $S_1 \times S_2$ is an offensive k -alliance in $\Gamma_1 \times \Gamma_2$.

From the above result we deduce that, a partition Π_i of Γ_i into r_i offensive k_i -alliances, $i \in \{1, 2\}$, induces a partition of $\Gamma_1 \times \Gamma_2$ into $r_1 r_2$ offensive k -alliances,

with $k = \min\{k_2 - \Delta_1, k_1 - \Delta_2\}$. So, we obtain the following result.

Corollary 14. For any graph Γ_i of order n_i and maximum degree Δ_i , $i \in \{1, 2\}$, and for every $k \leq \min\{k_1 - \Delta_2, k_2 - \Delta_1\}$, $\psi_k^o(\Gamma_1 \times \Gamma_2) \geq \psi_{k_1}^o(\Gamma_1)\psi_{k_2}^o(\Gamma_2)$.

Example of equality in the above result is $\psi_{-3}^o(C_4 \times K_4) = 4 = \psi_0^o(C_4)\psi_1^o(K_4)$.

Theorem 15. Let $\Gamma_i = (V_i, E_i)$ be a graph of order n_i and let Π_i be a partition of Γ_i into r_i global offensive k_i -alliances, $i \in \{1, 2\}$. If $x_i = \min_{X \in \Pi_i} \{|X|\}$, $y_i = \max_{X \in \Pi_i} \{|X|\}$ and $k \leq \min\{k_1, k_2\}$. Then $\gamma_k^o(\Gamma_1 \times \Gamma_2) \leq \min\{n_2x_1, n_1x_2\}$, and $\psi_k^{go}(\Gamma_1 \times \Gamma_2) \geq \max\{\psi_{k_1}^{go}(\Gamma_1), \psi_{k_2}^{go}(\Gamma_2)\}$.

Corollary 16. If a graph Γ_i of order n_i is partitionable into global offensive k_i -alliances, $i \in \{1, 2\}$, then for $k \leq \min\{k_1, k_2\}$,

$$\gamma_k^{go}(\Gamma_1 \times \Gamma_2) \leq \frac{n_1n_2}{\max_{i \in \{1,2\}} \{\psi_{k_i}^{go}(\Gamma_i)\}}.$$

Example of equality is $\gamma_1^o(C_4 \times K_2) = \frac{4 \cdot 2}{\max\{\psi_1^{go}(C_4), \psi_1^{go}(K_2)\}} = 4$.

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

References

- [1] S. Bermudo, J. A. Rodríguez-Velázquez, J. M. Sigarreta and I. G. Yero, On global offensive k -alliances in graphs. Submitted.
- [2] L. Eroh, R. Gera, Global alliance partition in trees. *J. Combin. Math. Combin. Comput.* **66** (2008) 161-169.
- [3] L. Eroh, R. Gera, Alliance partition number in graphs. Accepted.
- [4] T. W. Haynes and J. A. Lachniet, The alliance partition number of grid graphs. *AKCE Int. J. Graphs Comb.* **4** (1) (2007) 51-59.
- [5] P. Kristiansen, S. M. Hedetniemi and S. T. Hedetniemi, Alliances in graphs. *J. Combin. Math. Combin. Comput.* **48** (2004) 157-177.
- [6] K. H. Shafique, Partitioning a Graph in Alliances and its Application to Data Clustering. Ph. D. Thesis, 2004.
- [7] K. H. Shafique and R. D. Dutton, On satisfactory partitioning of graphs. *Congr. Numer.* **154** (2002) 183-194.

Increasing the Edge Connectivity by One in $\mathcal{O}(\lambda_G n^2 \log^{(*)} n)$ Expected Time

Michael Brinkmeier

Technische Universität Ilmenau
Faculty of Computer Science and Automation
Institute for Theoretical Computer Science
mbrinkme@tu-ilmenau.de

Key words: Undirected graphs, edge connectivity, minimum cut, polynomial time algorithm, edge connectivity augmentation, extreme set, maximum adjacency order

In some situations an undirected multigraph has to be ‘enriched’ by a minimum number of additional edges, such that there exists at least a given number k of edge-disjoint paths between every pair of vertices. Due to the duality of maximum flows and minimum cuts, this corresponds to the increase of the edge connectivity to the target value k by a minimum number of additional edges. Finding this *minimum k -augmenting set* of additional edges is called the *edge connectivity augmentation problem* (ECA). In this paper we will concentrate on the case $k = \lambda_G + 1$, ie. the problem of *edge connectivity augmentation by one* (ECA1).

Several strongly and pseudo-polynomial algorithms for ECA are known up to date. Basically these can be split into two groups. The algorithms of the first type use the process of *edge splitting*. Based on the results of Cai and Sun [3], Frank [5] described such a strongly polynomial algorithm requiring time $\mathcal{O}(n^5)$ on a graph with n vertices and m edges. Gabow [8] obtained $\mathcal{O}(n^2 m \log(n^2/m))$. Nagamochi and Ibaraki [13; 14] described an $\mathcal{O}(nm \log n + n^2 \log^2 n)$ algorithm.

The other group of algorithms does not use edge splitting explicitly. Many of them iteratively solve ECA1, ie. they increase the edge connectivity of the graph one by one. This approach usually results in pseudo-polynomial runtimes, which depend on the number of necessary steps, ie. the difference $k - \lambda_G$. One of the eldest is described in [15] and requires $\mathcal{O}(kLn^4(kn + m))$ time with $L = \min\{k, n\}$. Naor, Gusfield and Martel [12] described an $\mathcal{O}(\delta^2 nm + \delta^3 n^2 + n \cdot M(G))$ algorithm, where δ is the number of required steps and $M(G)$ the time required by a maximum flow algorithm on the graph G . In [6; 7] Gabow describes an algorithm requiring $\mathcal{O}(m + k^2 n \log n)$ time for simple graphs.

Benczúr and Karger [1] describe a nondeterministic algorithm mixing both approaches. They use a probabilistic algorithm for the construction of all *extreme* sets as defined in [15] of vertices and then use these to implicitly construct an edge splitting, increasing the edge connectivity to $k - 1$. For the last step they suggest a *cactus based** algorithm, like those in [12] or [9]. The resulting algorithm constructs a minimum k -augmenting set in time $\mathcal{O}(n^2 \log^5 n)$ with high probability. In [10; 11] Nagamochi describes an algorithm, which allows the deterministic computation of all extreme sets in an undirected, weighted graph in $\mathcal{O}(mn + n^2 \log n)$ time, resulting in an algorithm for ECA in the fashion of Benczúr and Karger, requiring $\mathcal{O}(mn + n^2 \log n)$ time and $\mathcal{O}(m + n)$ space.

In this paper we present a new algorithm for the solution of ECA1. We show that its runtime is bounded by $\mathcal{O}(\lambda_G n^2 \log n)$ and that its expected runtime is $\mathcal{O}(\lambda_G n^2 \log^{(*)} n)$. Furthermore, due to a quite conservative estimation, the average runtime of the algorithm may be significantly lower. The algorithm may either be used to solve ECA iteratively, or as the last step in Benczúr and Karger’s algorithm, avoiding the construction of the cactus.

We assume, that the edge connectivity λ_G of the underlying integer weighted graph is already known (see eg. [2]). Our algorithm consists of two main components. The first component is an algorithm for the computation of all λ_G -*extreme sets*, which are the minimal minimum cuts of G . As we show, this can be achieved in time $\mathcal{O}(\lambda_G n^2)$ by using *Lax-Adjacency Orders* as introduced in [2].

It is a simple observation, that the increase of the edge connectivity by one, requires that for every λ_G -extreme at least one new edge has to leave it. This leads to a lower bound of $\lceil \frac{l}{2} \rceil$ for the number of required additional edges, where l is the number of λ_G -extreme sets. As proven in [5], this lower bound is in fact the minimum number of edges required to solve ECA1. In other words, an optimal solution consists of $\lfloor \frac{l}{2} \rfloor$ disjoint pairs of λ_G -extreme sets and edges between the two components of each pair. If l is odd, then one additional edge leaving the remaining set has to be added.

Due to the minimality of the λ_G -extreme sets and the fact that the intersection of two minimum cuts is again a minimum cut, every minimum cut contains at least one λ_G -extreme set. We call a pair (X, Y) of λ_G -extreme sets illegal, if there exists a minimum cut Z , that contains X and Y and no other λ_G -extreme set. In this situation the addition of an edge between X and Y would require that an additional edge leaves Z , implying that the pair cannot be part of an optimal solution.

Basically our algorithm chooses an arbitrary *pairing* of the λ_G -extreme sets and then detects illegal pairs by constructing all λ_G -extreme sets in the graph, in which edges between the pairs were added. The edges induced by legal pairs remain

* *Cactus-based* means that the algorithm requires the construction of the cactus representation of all minimum cuts in the multigraph G , as described in [4; 6]

in the solution, while the illegal pairs are split again, and the process is repeated until no illegal pair was found.

As we observed, each λ_G -extreme set can be a member of at most two illegal pairs. If known illegal pairs are avoided in subsequent rounds, it can be shown that at most $\mathcal{O}(\log n)$ rounds are required, leading to a total runtime of $\mathcal{O}(\lambda_g n^2 \log n)$. A more thorough analysis reveals, that a random choice of the pairings is expected to require only $\mathcal{O}(\log^{(*)} n)$ rounds, leading to an expected runtime of $\mathcal{O}(\lambda_G n^2 \log^{(*)} n)$.

References

- [1] András A. Benczúr and David R. Karger. Augmenting undirected edge connectivity in $\tilde{o}(n^2)$ time. *J. Algorithms*, 37(1):2–36, 2000.
- [2] Michael Brinkmeier. A simple and fast min-cut algorithm. *Theory of Computing Systems*, 41:369–380, 2007.
- [3] Guo-Ray Cai and Yu-Geng Sun. The minimum augmentation of any graph to a k -edge-connected graph. *Networks*, 19:151–172, 1989.
- [4] E.A. Dinic, A.V. Karzanov, and M.V. Lomonosov. On the structure of a family of minimal weighted cuts in a graph. *Studies in Discrete Optimization*, pages 290–306, 1976.
- [5] András Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM J. Discrete Math.*, 5(1):25–53, 1992.
- [6] Harold N. Gabow. Applications of a poset representation to edge connectivity and graph rigidity. In *FOCS*, pages 812–821. IEEE, 1991.
- [7] Harold N. Gabow. Applications of a poset representation to edge connectivity and graph rigidity. Technical Report CU-CS-545-91, University of Colorado, 1991.
- [8] Harold N. Gabow. Efficient splitting off algorithms for graphs. In *STOC*, pages 696–705, 1994.
- [9] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *J. ACM*, 43(4):601–640, 1996.
- [10] Hiroshi Nagamochi. Computing extreme sets in graphs and its applications. In *Proceedings of the 3rd Hungarian-Japanese Symp. on Disc. Math. and Its Appl.*, pages 349–357, Tokyo, 2003.
- [11] Hiroshi Nagamochi. Graph algorithms for network connectivity problems. *Journal of the Operations Research Society of Japan*, 47(4):199–223, 2004.
- [12] Dalit Naor, Dan Gusfield, and Charles U. Martel. A fast algorithm for optimally increasing the edge-connectivity. In *FOCS*, volume II, pages 698–707. IEEE, 1990.
- [13] Hiroshi Nagamochi and Toshihide Ibaraki. Deterministic $\tilde{O}(nm)$ time edge-

- splitting in undirected graphs. In *STOC*, pages 64–73, 1996.
- [14] Hiroshi Nagamochi and Toshihide Ibaraki. Graph connectivity and its augmentation: applications of MA orderings. *Discrete Applied Mathematics*, 123(1-3):447–472, 2002.
 - [15] Toshimasa Watanabe and Akira Nakamura. Edge-connectivity augmentation problems. *J. Comput. Syst. Sci.*, 35(1):96–144, 1987.

Enumerating all finite sets of minimal prime extensions of graphs

V. Giakoumakis,^a C.B. Ould El Mounir^a

^a*MIS, Université d'Amiens, France*

Vassilis.Giakoumakis@u-picardie.fr

Abstract

In [2] V. Giakoumakis and S. Olariu characterized all classes of graphs whose set of minimal prime extensions is finite. In this extended abstract we propose an efficient algorithm that enumerates all minimal prime extensions of these classes of graphs.

Key words: modules in graphs, modular decomposition, minimal prime, extension, enumeration algorithm.

1. Motivation, notation and terminology

For terms not defined here the reader is referred to [1]. All considered graphs are finite, without loops nor multiple edges. Let G be a graph, the set of its vertices will be noted by $V(G)$ while the set of its edges will be noted by $E(G)$. An edgeless (resp. complete) graph of n vertices is denoted O_n (resp. K_n) while $[W]$ will denote the subgraph of G induced by $W \subseteq V(G)$. A chordless path (or chain) of k vertices will be denoted P_k and a chordless cycle of k vertices will be denoted by C_k . A *bull* is a graph formed from a P_4 and a vertex x which is adjacent to the middle vertices and misses the two extremities of this P_4 . The vertex x will be called the *top* vertex of the bull. A *diamond* (resp. *paw*) is a graph formed from a P_3 (resp. a $\overline{P_3}$) and a universal vertex with respect to the set of vertices of this P_3 (resp. $\overline{P_3}$). A set $M \subseteq V(G)$ is called a *module* if every vertex of G outside M is adjacent to all vertices of M or to none of them. The empty set, $V(G)$ and the singletons are *trivial* modules and whenever G has only trivial modules is called *prime* or *indecomposable*. A non trivial module M is also called *homogeneous set*. The graph G' is a *minimal prime extension* of G if G' is prime, it contains an induced subgraph isomorphic to G and is minimal with respect to set inclusion and primality. $Ext(G)$ will denote the set of minimal prime extensions of G . It is well known that $Ext(G)$ is not necessarily a finite set.

Finding characterizations of the set of minimal prime extensions of classes of graphs could lead to efficient optimization algorithms (see in [2] for details) and this has motivated many researching works in this direction. In ([4]) is proved that if G is a P_4 -homogeneous graph (i.e. every non trivial module of G induces a subgraph of a P_4) then $Ext(G)$ is a finite set.

The open problem of giving necessary and sufficient conditions for the finiteness of $Ext(G)$ was recently solved in [2] where it is proved that $Ext(G)$ is a finite set iff G is either a P_4 -homogeneous graph or a $2P_4$ -homogeneous graph whose definition is given below:

Definition 1.1 Let G be a connected graph which is not P_4 -homogeneous such that \overline{G} contains exactly two connected components C_1 and C_2 . Then, G and \overline{G} are said to be $2P_4$ -homogeneous graphs if $[C_1]$ and $[C_2]$ are subgraphs of a chordless chain and one of the following conditions holds:

- $[C_1]$ is a singleton and $G' = G \setminus u$ is a P_4 -homogeneous graph
- $[C_1]$ is isomorphic to a $P_1 \cup P_3$ or to a $P_1 \cup P_4$ or to an O_3 or to an $O_2 \cup P_2$ and $[C_2]$ is isomorphic to a P_4 , \overline{P}_3 or a \overline{P}_2 .

Using as framework the theoretical results in [2] and [4] we present in this abstract an efficient and easily programming method which enumerates the set of all minimal prime extensions in the finite case. In this way we unify also several previous researching works where this set is obtained by examining separately each particular case of the graphs under consideration.

2. Enumeration of $Ext(G)$ for a P_4 -homogeneous graph G

We shall first give the structure of the modular decomposition tree $T(G)$ of a P_4 -homogeneous graph G . We shall classify G in to 3 classes as follows:

Definition 2.1. Let G be a connected P_4 -homogeneous graph, let $T(G)$ be its corresponding modular decomposition tree and let $r(T)$ be the root of T . Then

- (i) G is of type 1 if $r(T)$ is an N -node and the subgraph of G corresponding to each son of $r(T)$ is isomorphic to a subgraph of a P_4
- (ii) G is of type 2 if $r(T)$ is an S -node having two sons and the subgraph of G corresponding to each son of $r(T)$ is isomorphic to either a P_4 or to a \overline{P}_3 or to \overline{P}_2 or to a P_1 .
- (iii) G is of type 3 if G is isomorphic to a C_3 or to a diamond or to a paw.

Theorem 2.2 Let G be a connected graph and let $T(G)$ be its corresponding

modular decomposition tree. If G is P_4 -homogeneous then G is of type 1, 2 or 3.

We recall that since the construction of the modular decomposition tree of a graph can be obtained in linear time on the size this graph (see [1]), Theorem 2.2 implies that the recognition of a P_4 -homogeneous graph G can be obtained in linear time on the size of G . The modular decomposition tree of a P_4 -homogeneous graph will be used in the enumeration algorithm presented below. Before, we give some definitions and we recall some known results.

Definition 2.3 ([3]) Let G be an induced subgraph of a graph H , and let W be a homogeneous set of G . We define a W -pseudopath in H as a sequence $R = (u_1, u_2, \dots, u_t)$, $t \geq 1$, of pairwise distinct vertices of $V(H) \setminus V(G)$ satisfying the following conditions:

- (i) u_1 is partial with respect to W .
- (ii) $\forall i = 2, \dots, t$ either u_i est adjacent to u_{i-1} and indifferent with respect to $W \cup \{u_1, \dots, u_{i-2}\}$ or u_i is total with respect to $W \cup \{u_1, \dots, u_{i-2}\}$ and not adjacent to u_{i-1} (when $i = 2$, $\{u_1, u_2, \dots, u_t\} = \emptyset$).
- (iii) $\forall i = 2, \dots, t-1$, u_i is total with respect to $N(W)$ and indifferent with respect to $V(G) - N(W)$ and either u_t is not adjacent to a vertex of $N(W)$ or u_t is adjacent to a vertex of $V(G) - N(W)$.

From ([3]) we know that for every homogeneous set W of a graph G there is a W -pseudopath with respect to any induced copy of G in $Ext(G)$.

Definition 2.4 Let W be a non trivial module of a graph G and let x be a partial vertex with respect to W . We shall say that x is a *strong partial* vertex for W if W does not contain an homogeneous set in the graph induced by $V(G) \cup \{x\}$. If x is the first vertex of a W -pseudopath P , P will be called a *strong* pseudopath.

Definition 2.5 A graph G will be called (P_4, bull) -homogeneous graph if every homogeneous set of G induces either a subgraph of a P_4 or is isomorphic to a bull.

From ([4]) we know that for every homogeneous set W of a P_4 -homogeneous graph G there is in $Ext(G)$ a strong W -pseudopath $P = x_1$ or a pseudopath $P = x_1, x_2$. The latter case occurs whenever $[W \cup \{x_1\}]$ is isomorphic to a bull and $W \cup \{x_1\}$ forms a non trivial module in the graph induced by $V(G) \cup \{x_1\}$. In this case x_2 is a strong $W \cup \{x_1\}$ -pseudopath such that x_2 either is only adjacent to the top vertex x_1 of $[W \cup x_1]$ and misses all vertices of W or x_2 is not adjacent to x_1 and is total with respect to W .

We are now in position to present our enumeration algorithm. We shall enumerate the set of minimal prime extensions of a connected P_4 -homogeneous graph G of type 1 or 2. A graph G of type 3 is a particular case of a P_4 -homogeneous graphs and $Ext(G)$ is already known by previous researching works. Although we could

adapt our algorithm in order to enumerate also $Ext(G)$ in this case, we prefer to invite the reader to see [2] for the related references. Finally, whenever G is disconnected, the set of minimal prime extensions \mathcal{H} of G will be the set of the complementary graphs of \mathcal{H} (see [2]).

The enumeration algorithm of $Ext(G)$

Input: A connected P_4 -homogeneous graph Q of type 1 or 2

Output: The set of minimal prime extensions \mathcal{H} of Q

- (i) Consider the set $\mathcal{F}(Q) = \{G^1, \dots, G^t\}$ of $(P_4, bull)$ -homogeneous graphs containing Q as induced subgraph
- (ii) Let G^i be a graph of $\mathcal{F}(Q)$ and let U_1, \dots, U_k be the set of the maximal non trivial modules of G^i that can be computed from the modular decomposition tree of G^i .
- (iii) $Ext_1(G^i)$ is the set of all graphs obtained by adding to G^i the set $X = \{x_1, \dots, x_k\}$ of new vertices such that:
 - (a) each x_i form a strong U_i - pseudopath of length 1 and $[X]$ is edgeless
 - (b) there is no edge between x_i and U_j , $i \neq j$ and $i, j = 1 \dots k$
- (iv) $Ext_2(G^i)$ is be the set of all graphs obtained by considering every graph of $Ext_1(G^i)$ and then identifying in all possible ways the vertices of X of (the neighborhood of the vertex resulted from the identification of the two vertices x and y is $N(x) \cup N(y)$). $Ext_3(G^i)$ is the set of all graphs obtained by adding edges in all possible ways between the vertices of $V(G_j) \cap X$, of every graph G_j of $Ext_2(G^i)$. Finally, $Ext_4(G^i)$ is the set of all graphs obtained by adding edges in all possible ways between the vertices of $V(G_l) \cap X$ of every graph G_l of $Ext_3(G^i)$ and the non trivial modules of G_l in the following manner: for $x \in V(G_l) \cap X$ and for an homogeneous set U of G_l such that x is indifferent with respect to U , we add all edges between x and the vertices of U if $[U]$ is not isomorphic to a P_3 or to its complement; if $[U]$ is isomorphic to a $P_3 = abc$ or to its complement we add either all edges between x and $\{a, b, c\}$ or the edge xb or the edges xa and xc .
- (v) The set of minimal prime extensions \mathcal{H} of Q is obtained from the union of the sets $Ext_1(G^i) \cup Ext_2(G^i) \cup Ext_3(G^i) \cup Ext_4(G^i)$, $i = 1, \dots, t$.

3. Enumerating $Ext(G)$ whenever G is a $2P_4$ - homogeneous graph

The space limitations of this extended abstract do not allow us to give the details for obtaining $Ext(G)$ in all cases that may occur for a $2P_4$ -homogeneous graph G . We shall only present here a general method for enumerating $Ext(G)$.

We first enumerate $Ext([C_2])$ (since $[C_2]$ is a P_4 -homogeneous graph, $Ext([C_2])$

can be found as exposed in previous section). Let H be a graph of $Ext([C_2])$ and let A be a set of new vertices such that $[A]$ is isomorphic to $[C_1]$. Then add edges between A and $V(H)$ as follows:

- 1 A is partial with respect to $V(H)$ and the graph induced by $V(H) \cup A$ is a minimal prime extension of G . Let F_1 be the set of graphs obtained in this manner.
- 2 A is total with respect to $V(H)$. Hence the graph H' induced by $V(H) \cup A$ contains two maximal modules, A and $V(H)$. Then add to H' a strong B -pseudopath and a strong $V(H)$ -pseudopath which is $2K_2$ -free and containing at most 9 vertices. Let F_2 be the set of graphs obtained with this manner.
- 3 $Ext(G)$ is a subset of $F_1 \cup F_2$.

Remark. Following the structure of the $2P_4$ -homogeneous graph G under consideration, the general method presented above can be consequently adapted in order to enumerate with precision the corresponding set of graphs $F_1 \cup F_2$.

References

- [1] A. Brandstädt, V.B. Lee and J. Spinrad, *Graph classes: a survey*, SIAM Monographs on Discrete Mathematics and Applications, 1999.
- [2] V. Giakoumakis, Stephan Olariu, *All minimal prime extensions of hereditary classes of graphs*, Theor. Comput. Sci. **370**(1-3): 74-93 (2007)
- [3] I. Zverovich, *Extension of hereditary classes with substitutions*, Discrete Applied Mathematics **128**, (2-3), (2003) 487-509
- [4] I. Zverovich, *A finiteness theorem for primal extensions*, Discrete Mathematics, **293**, (1), (2005), 103-116.

Networks II

Lecture Hall B

Thu 4, 10:30–12:30

On planar and directed multicuts with few source-sink pairs

Cédric Bentz^a

^aLRI, Univ. Paris-Sud and CNRS, 91405 ORSAY Cedex
cedric.bentz@lri.fr

Key words: Multicuts, Graph algorithms, **NP**-hardness, **APX**-hardness

1. Introduction

Assume we are given a n -vertex m -edge (di)graph $G = (V, E)$, a *weight function* $c : E \rightarrow \mathbb{N}^*$ and a list \mathcal{N} of pairs (source s_i , sink s'_i) of *terminal* vertices. The *minimum multicut problem*, or **MINMC**, consists in selecting a minimum weight set of edges (or arcs) whose removal leaves no (directed) path from s_i to s'_i for each i . The *minimum multiterminal cut problem* (**MINMTC**) is a particular minimum multicut problem in which, given a set of r vertices $\{t_1, \dots, t_r\}$, the source-sink pairs are (t_i, t_j) for $i \neq j$.

For $|\mathcal{N}| = 1$, **MINMC** is equivalent to the classical *minimum cut problem*, and therefore is polynomial-time solvable both in directed and in undirected graphs. However, **MINMC** (resp. **MINMTC**) becomes **NP**-hard, and even **APX**-hard, as soon as $|\mathcal{N}| = 3$ (resp. $r = 3$) in undirected graphs [7] (the cases $r = 2$ and $|\mathcal{N}| = 2$ being tractable [12]), and as soon as $|\mathcal{N}| = 2$ (resp. $r = 2$) in digraphs [10]. For an arbitrary number of source-sink pairs, **MINMC** is **APX**-hard even in unweighted stars [9]. Moreover, **MINMC** is polynomial-time solvable in directed trees (the constraint matrix being totally unimodular) and **MINMTC** is polynomial-time solvable in directed acyclic graphs [6].

It is generally believed that **MINMC** is not significantly simpler in directed acyclic graphs. In [2], it was proved that **MINMC** is **NP**-hard even in unweighted directed acyclic graphs (or **DAG**) having a very special structure (namely, their underlying undirected graph is a bipartite cactus of bounded path-width and of maximum degree three), but its **APX**-hardness remained open, as well as its complexity when $|\mathcal{N}|$ is fixed. Moreover, **MINMTC** is known to be **NP**-hard in planar graphs, where it becomes tractable if r is fixed [7], and, in [1], **MINMC** has been shown

to be polynomial-time solvable in planar graphs where all the terminals lie on the outer face, if $|\mathcal{N}|$ is fixed.

In this extended abstract, we first show that MINMC is **APX**-hard in DAG, even if $|\mathcal{N}| = 3$. The proof is surprisingly simple, and only relies on the **APX**-hardness of VERTEX COVER in bounded-degree undirected graphs. Then, we improve the result in [1], by giving a nearly linear-time algorithm for MINMC when the graph is planar, $|\mathcal{N}|$ is fixed, and the terminals lie on the outer face.

2. **APX**-hardness proof for DAG

The structure of our proof is simple: we will give an approximation-preserving reduction from VERTEX COVER in graphs of maximum degree 3, which is known to be **APX**-hard [11], to MINMC in unweighted DAG with $|\mathcal{N}| = 3$. This will immediately implies the **APX**-hardness of the latter problem.

As in [2], we consider an instance of VERTEX COVER in graphs of maximum degree 3, and transform this (undirected) graph G into a DAG, by first numbering the vertices arbitrarily, and then orienting the edges so that this numbering defines a topological order. Then, we replace each vertex v_i by an arc (v'_i, v''_i) and any arc of the form (v_i, v_j) by an arc (v''_i, v'_j) . Let G' be this new digraph (note that G' is also a DAG, and has maximum degree 4). To finish the reduction, we add six new vertices $s_1, s_2, s_3, s'_1, s'_2, s'_3$, and, following the topological order of the v'_i in G' , we link, for each pair (v'_i, v'_j) such that (v_i, v_j) is an arc of G , vertex s_h to v'_i and v'_j to s'_h by two new arcs, where $h \in \{1, 2, 3\}$ is the smallest index such that there is no arc from v''_i to s'_h yet.

Since G is of maximum degree 3, it is not difficult to show that:

Claim 1. Such an index h always exists, so the construction is always possible.

This yields a MINMC instance where the terminal pairs are (s_i, s'_i) for $i = 1, 2, 3$, and which, as can be easily seen, is such that: for each integer S , there is a vertex cover of size S in G iff there is an arc multicut of size S in G' . So:

Theorem 16. MINMC is **APX**-hard in unweighted DAG, even when $|\mathcal{N}| = 3$.

An interesting open problem would be to settle the case where there are only two source-sink pairs (recall that MINMC is then **APX**-hard in general digraphs [10]). Moreover, our result is best possible (up to constant factors), in the sense that there exists a trivial 3-approximation algorithm for MINMC when $|\mathcal{N}| = 3$ (for each i , compute a minimum simple cut between s_i and s'_i , then take as a multicut the union of these three cuts). It also matches the best inapproximability result known for this problem in *unrestricted digraphs* that is based on the assumption that $\mathbf{P} \neq \mathbf{NP}$

[10] (the $\Omega\left(\frac{\log n}{\log \log n}\right)$ inapproximability bound of Chuzhoy and Khanna [5] being based on a stronger complexity assumption, and the $\Omega(1)$ inapproximability bound of Chawla et al. [3] being based on a different one, namely the *Unique Games Conjecture*).

3. An efficient algorithm for planar graphs

In [1], a polynomial algorithm for MINMC in planar graphs with fixed $|\mathcal{N}|$ and all the terminals lying on the outer face was given, but it was not an FPT algorithm [8]. Here we give the sketch of an FPT algorithm for this case.

Recall that the algorithm in [1] was based on a refinement of an idea from [7]: when we remove the edges of any optimal solution to a MINMC instance, the terminals are clustered in $p \leq 2|\mathcal{N}|$ connected components. So, after having “guessed” the right clustering of the terminals (which is done by brute force enumeration on the set of terminals), we can obtain an equivalent MINMTC instance by merging, for each cluster, all the terminals of this cluster into a single terminal. If the graph remains planar after this operation, then we can use the algorithm given in [7]; this was the main idea used in [1]. However, if, in the MINMTC instance we obtain, all the terminals were lying on the outer face, then we could use the much more efficient algorithm given in [4], which runs in $O(n \log n)$ time. The basic idea of our new algorithm is thus to call this fast algorithm several times, in order to split up the graph into several (connected) components, each one of them being, in turn, a new MINMTC instance on which we can apply the same algorithm once again. So, we follow a *divide-and-conquer* approach, and, unlike in [1], solve several planar MINMTC instances (whose terminals lie on the outer face) in order to solve the MINMTC instance (whose terminals does not necessarily lie on the outer face) associated with the optimal clustering of the initial MINMC instance. But how should we split up the graph? The next lemma is our starting point:

Lemma 5. Assume we are given a MINMC instance \mathcal{M} in an undirected 2-vertex-connected planar graph G where the terminals lie on the outer face, and an optimal clustering of the terminals for this instance. Let us denote by $\mathcal{C}_1, \dots, \mathcal{C}_q$ the clusters of this clustering which are not included in other clusters (a cluster \mathcal{C} being included in another cluster \mathcal{C}' if any path on the outer face linking, clockwise, two terminals of \mathcal{C} is included in some path on the outer face linking, clockwise, two terminals of \mathcal{C}'). Then, the edges of any optimal solution for the MINMTC instance obtained in G by considering only the terminals in $\mathcal{C}_1, \dots, \mathcal{C}_q$ and merging, for each i , the terminals in \mathcal{C}_i into a single terminal, are part of an optimal solution for \mathcal{M} .

The ideas used in the proof of this lemma are rather simple: start from an optimal multicut, and replace the edges lying between the connected components associated with the clusters \mathcal{C}_i by an optimal solution of the MINMTC instance

described in the lemma. It is a reasonably easy task to show that the new solution is also an optimal multicut. Hence, starting from G , we can recursively define a MINMTC instance by taking the \mathcal{C}_i 's as terminals, solve it (by the algorithm given in [4], since the terminals lie on the outer face), and then apply this approach on each one of the connected components obtained by removing the edges of this optimal multiterminal cut. The algorithm given in [4] runs in $O(n \log n)$ time, and, for a given clustering (there are $O(1)$ possible clusterings when $|\mathcal{N}|$ is fixed), we call it $O(|\mathcal{N}|)$ times: therefore, for fixed $|\mathcal{N}|$, our –FPT– algorithm also runs in $O(n \log n)$ time. An interesting open problem would be to determine whether a linear-time algorithm exists.

References

- [1] Bentz, C.: A simple algorithm for multicuts in planar graphs with outer terminals. *Discrete Applied Mathematics* **157** (2009) 1959–1964.
- [2] Bentz, C.: On the complexity of the multicut problem in bounded tree-width graphs and digraphs. *Discrete Applied Mathematics* **156** (2008) 1908–1917.
- [3] Chawla, S., Krauthgamer, R., Kumar, R., Rabani, Y., Sivakumar, D.: On the hardness of approximating multicut and sparsest-cut. *Proc. CCC* (2005).
- [4] Chen, D.Z., Wu, X.: Efficient algorithms for k -terminal cuts on planar graphs. *Algorithmica* **38** (2004) 299–316.
- [5] Chuzhoy, J., Khanna, S.: Hardness of Cut Problems in Directed Graphs. *Proc. STOC* (2006) 527–536.
- [6] Costa, M.-C., Létocart, L., Roupin, F.: Minimal multicut and maximal integer multiflow: a survey. *European J. of Operational Research* **162** (2005) 55–69.
- [7] Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiterminal cuts. *SIAM J. Comput.* **23** (1994) 864–894.
- [8] Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer, NY (1999).
- [9] Garg, N., Vazirani, V.V., Yannakakis, M.: Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica* **18** (1997) 3–20.
- [10] Garg, N., Vazirani, V.V., Yannakakis, M.: Multiway cuts in node weighted graphs. *Journal of Algorithms* **50** (2004) 49–61.
- [11] Papadimitriou, C., Yannakakis, M.: Optimization, approximation, and complexity classes. *J. Comput. and System Sciences* **43** (1991) 425–440.
- [12] Yannakakis, M., Kanellakis, P., Cosmadakis, S., Papadimitriou, C.: Cutting and partitioning a graph after a fixed pattern. *Proc. ICALP* (1983) 712–722.

Maintenance resources allocation on power distribution networks with a multi-objective framework

Fábio Luiz Usberti,^a José Federico Vizcaino González,^a
Christiano Lyra Filho,^a Celso Cavellucci^a

^aFEEC/UNICAMP, C.P. 6102, 13083-970 Campinas SP, Brazil
fusberti@yahoo.com

Key words: Network Optimization, Integer Non-Linear Problem, Pareto Frontier

Introduction. Power distribution companies are incumbent to transport electrical energy in order to attend all clients in a given network, subject to specified quality and reliability levels. The occurrence of network components failure is the main factor which compromise power systems reliability. Therefore, maintenance actions like repairs and component replacements are needed to reestablish the network healthy activity. These maintenance actions can be classified as preventive, when executed before the component failure, or corrective, otherwise. Given the increasing demand of reliable (uninterrupt) power supply and more severe quality inspections imposed by regulations entities, it becomes mandatory to rationalize investments on distribution network maintenance. This is done by first defining the relationship between maintenance and reliability, and then achieving the network reliability target through the lowest maintenance cost possible, or alternatively, seeking the most reliable network under maintenance resources constraints. This work proposes a multi-objective approach to tackle the Maintenance Resources Allocation Problem (MRAP), i.e., we have considered optimizing simultaneously both objectives: maintenance cost and network reliability, and so providing power distribution companies with a set of non-dominated (Pareto optimum) solutions to access the companies' decisions on maintenance investments.

Problem Definition. Most power distribution networks operate with a radial configuration, which means that, using a graph terminology, the network can be represented as a tree $T(V, E)$ rooted at a substation that provides a unique path from the substation to each load point (or node) v . Each node attends a given number of clients and contains a set of electrical equipments subject to failure. The occurrence of any equipment failure will determine power supply interruption of the corresponding node and recursively of all his offsprings. The MRAP is defined

in a time horizon of $t = 1, \dots, T$ years. In the following, we give some notation used in this work:

- $x_e^t = 1$ if equipment e receives maintenance in year t , 0 otherwise.
- $\delta_e^t =$ failure rate of equipment e in year t .
- $N_v =$ number of clients affected (including offsprings) by failure on node v .
- $p_e =$ preventive maintenance cost for equipment e .
- $c_e =$ corrective maintenance cost for equipment e .
- $m_{e0} =$ failure multiplier for equipment e on lack of maintenance.
- $m_{e1} =$ failure multiplier for equipment e when maintenance is executed.
- $m_e^t =$ failure multiplier applied on equipment e failure rate in year t .

We define the nature in which the network equipments deteriorate or improve along the time horizon, whether if they receive or not maintenance, as failure rate model. In this model, the equipments failure rates are updated through the failure rate multipliers, according to the actions applied (0.1). The maintenance cost is expressed in terms of corrective maintenance cost (0.2) and preventive maintenance cost (0.3). The network reliability is given through the SAIFI (System Average Interruption Frequency Index) (0.4).

$$\delta_e^t = \begin{cases} \delta_e^{t-1} m_{e1} & \text{if equipment } e \text{ receives maintenance} \\ \delta_e^{t-1} m_{e0} & \text{otherwise} \end{cases} \quad (0.1)$$

$$C_c^t = \sum_{v \in V} \sum_{e \in v} c_e \delta_e^t \quad (0.2)$$

$$C_p^t = \sum_{v \in V} \sum_{e \in v} p_e x_e^t \quad (0.3)$$

$$SAIFI^t = \frac{\sum_{v \in V} (N_v \sum_{e \in v} \delta_e^t)}{\sum_{v \in V} N_v} \quad (0.4)$$

The bi-objective integer non-linear mathematical model for MRAP is given:

$$MIN \quad CT = \sum_t \frac{1}{(1+i)^t} (C_c^t + C_p^t) \quad (0.5)$$

$$MIN \quad SAIFI = \max_t (SAIFI^t) \quad (0.6)$$

st

$$m_e^t = x_e^t m_{e1} + (1 - x_e^t) m_{e0} \quad \forall e, \forall t \quad (0.7)$$

$$\delta_e^t = \delta_e^{t-1} m_e^t \quad \forall e, \forall t \quad (0.8)$$

$$x_e^t \in \{0, 1\} \quad \forall e, \forall t \quad (0.9)$$

The objective function (0.5) minimizes the maintenance total cost, adjusted by the present value, given an interest rate i . The objective function (0.6) minimizes the maximum $SAIFI^t$ obtained through all the time horizon. Constraints (0.7) guarantee that the failure multiplier m_e^t applied to each equipment e and year t must

be one of two possible values, m_{e0} or m_{e1} ; (0.8) determine the failure rate of each equipment along the time period, depending on the maintenance applied; finally, (0.9) correspond the binary constraints on the decision variables.

Solving Strategy. To solve the MRAP model, we use a traditional scalarization technique, the ε -Constraint Method [2], where only one of the objectives, maintenance cost, is minimized, while the SAIFI is transformed into a constraint (0.10). To obtain the set of non-dominated solutions, the previous model should be solved p times under distinct values of ε_k , where $\varepsilon_1 = SAIFI_{max}$ and $\varepsilon_p = SAIFI_{min}$. These SAIFI extreme values can be calculated by considering $x_e^t = 0$ and $x_e^t = 1$ ($\forall t, \forall e$), respectively. To distribute the p values of ε_k uniformly between the SAIFI extreme values, they are computed by (0.11).

$$\max_t(SAIFI^t) \leq \varepsilon_k \quad (0.10)$$

$$\varepsilon_k = SAIFI_{min} + \frac{(SAIFI_{max} - SAIFI_{min})(k - 1)}{p - 1} \quad k = 1, \dots, p \quad (0.11)$$

Given a power distribution network, the Pareto frontier is thus obtained after p iterations of the ε -constraint method. To solve each iteration, an efficient genetic algorithm specifically developed for MRAP [1] is executed.

Composition of Pareto Frontiers. In general, a power distribution company is expected to have not one, but several distribution networks. In theory, all these networks could be considered as a single instance and then solved by the procedure previously described. Nevertheless, this would result in an overwhelming computational effort. This work introduces a divide-and-conquer technique to surmount this problem: it first solves each network individually (division phase) and then composes all Pareto frontiers into a single one (conquer phase). This last phase introduces a new combinatorial problem, which we call Composition of Pareto Frontiers Problem (CPFP).

The input of the CPFP is a set of n Pareto frontiers and a vector NC^i ($i = 1, \dots, n$) which represents the number of clients attended by network i . For the sake of simplicity, we are considering that each frontier has the same number p of non-dominated solutions. The j^{th} solution from the i^{th} Pareto frontier $S_j^i = (CT_j^i, SAIFI_j^i)$ is represented by the pair of objectives. In the CPFP we select one solution from each Pareto frontier (i.e., some j for each $i = \{1, \dots, n\}$), in order to produce a composite solution $S^C = (CT^C, SAIFI^C)$ which we desire to be non-dominated. Supposing we have chosen the n solutions as $(S_{j_1}^1, S_{j_2}^2, \dots, S_{j_n}^n)$, then the composite solution can be determined by (0.12).

$$S^C = \left(\sum_{i=1}^n CT_{j_i}^i, \frac{\sum_{i=1}^n NC^i SAIFI_{j_i}^i}{\sum_{i=1}^n NC^i} \right) \quad (0.12)$$

A naive strategy to determine a set of p well distributed non-dominated composite solutions could be choosing all possible solution combinations of the Pareto frontiers and then filtering p well distributed non-dominated compositions. That would lead to p^n composite calculations, which is excessive considering that power distribution companies may have, for example, more than $n = 50$ networks, and that a good Pareto frontier should have at least $p = 20$ solutions. A better way to achieve this is to compound the Pareto frontiers two-by-two, always preserving p well distributed non-dominated compositions in the resulting frontier; the process is repeated until only one frontier remains. This will reduce the calculations to $(n - 1)p^2$, which is perfectly acceptable.

Study Cases. The methodology described above was applied to a fictional small-scaled group of three distribution networks (15600 clients, 105 equipments), based on [3], and to real large-scale group of five distribution networks (48247 clients, 6314 equipments). The results confirm the suitability of the strategy to find good quality non-dominated solutions for the MRAP, and then composing them into one Pareto frontier.

Conclusions. This work proposes a multi-objective approach to solve MRAP: a hard, non-linear, combinatorial problem which concerns major power distribution companies. The purpose of the methodology is to help decisions on investments applied to network maintenance, giving the companies decision makers proper information about the best trade-offs between maintenance investments and their feedback into system reliability.

Acknowledgment. The authors acknowledge the support from the Brazilian National Council for Scientific and Technological Development (CNPq).

References

- [1] P. A. Reis (2007), "Reliability based optimization of maintenance plans for power distribution systems", Master's Thesis, FEEC-UNICAMP, Campinas - SP (in Portuguese).
- [2] M. Ehrgott (1995), "Multicriteria Optimization", Springer, pp. 97-123.
- [3] A. Sittithumwat and F. Soudi and K. Tomsovic (2004), "Optimal allocation of distribution maintenance resources with limited information", *Electric Power Systems Research*, Vol. 68, pp. 208-220.

Column Generation for the Multicommodity Min-cost Flow Over Time Problem

Enrico Grande,^a Pitu B. Mirchandani,^b Andrea Pacifici^a

^a*Dip. di Ingegneria dell'Impresa, Università degli Studi di Roma "Tor Vergata", Italy*
{grande,pacifici}@disp.uniroma2.it

^b*Systems and Industrial Engineering Dept., University of Arizona, USA*
pitu@sie.arizona.edu

Key words: flows over time, column generation, exact algorithm.

1. Introduction

In most networks flows models, the time dimension is not explicitly considered. This assumption is unrealistic in several applications such as road and air traffic management, or water distribution. In *flows-over-time* models, the flow is allowed to vary over time and it requires a positive amount of time to travel through an arc. Reviews of applications and fundamental theory results are reported in the surveys [2; 5; 8].

The notion of flows over time (or dynamic flows) is introduced by Ford and Fulkerson. In [6; 7] they give efficient solution algorithms for the *maximum flow over time* problem: Given a capacitated network $G = (V, E)$, source and destination nodes, and arcs traversal times, find a flow over time maximizing the amount of flow reaching the destination node within a given time horizon T . Capacity constraints are expressed as upper bounds on the flow rates on the arcs. A related problem, polynomially solvable, is the *quickest (s, t) -flow*: Find a dynamic flow such that a certain demand is shipped and the time horizon T is minimized (see Burkard *et al.* [12]). Gale [4] introduce the *earliest arrival flows*, i.e., flows such that the amount reaching the destination node is maximal for all times t , $0 \leq t \leq T$. Pseudo-polynomial algorithms for finding such flows—which are based on the successive shortest path algorithm—have been devised by Wilkinson and Minieka [13; 11].

If we consider arc-dependent costs and we look for *minimum cost dynamic flows* satisfying a demand within a given time horizon T , things get harder. Klinz and Woeginger [3] show that the single source, single destination case is NP-hard

even for series-parallel graphs. Fleischer and Skutella in [10] show that the min-cost single commodity problem never requires the flow to wait at intermediate nodes. The *multicommodity* setting is, of course, NP-hard as well, as showed by Hall, Hippler and Skutella [1]. On the positive side, Fleischer and Skutella [9] provide an approximation algorithm for the quickest multicommodity dynamic problem.

In this paper we present an exact algorithm for the Multicommodity Flow over Time Problem (MFTP) which is based on a column generation approach for a path-based linear programming model. The column generation subproblem is shown to be binary NP-hard and we devise a pseudo-polynomial dynamic programming algorithm for its solution. The results of a preliminary computational study is also reported.

2. A linear programming model for MFTP

We use the following notation. $G = (V, E)$ is a digraph. K is a set of commodities to be served within the time horizon (or makespan) T : Each commodity $k \in K$ is identified by the triple $(s_k, t_k, b_k) \in (V \times V \times \mathbb{R}_+)$, i.e. source, destination and demand of commodity. P_k is the set of paths that one can use to serve commodity $k \in K$. We assume that $\bigcap_{k \in K} P_k = \emptyset^*$ and let $P = \bigcup_{k \in K} P_k$. The capacity of arc $e \in E$, expressed as a rate (unit of flow/time), is u_e , c_e its cost, and t_e its traversal time. Given a path $p \in P$, we let $c_p = \sum_{e \in p} c_e$ be the cost and t_p the traversal time of p . Moreover, if $e \in p$, we let $t_{e,p}$ be the traversal time of path $p \in P$ up to (and including) arc $e \in E$. (Clearly, if e' is the last arc of path p , then $t_p = t_{e',p}$.) We also use an incidence vector $\delta_{e,p} = 1$ [0] if $e \in p$ [$e \in E \setminus p$].

In the following linear programming problem, decision variable $f_p(t)$ represents the amount of flow starting from the origin of path p at time t , $p \in P$, $t = 1, \dots, T$.

$$\begin{aligned}
 & \min \sum_{p \in P} \sum_{t \in \{1, \dots, T\}} c_p f_p(t) \\
 (\mathcal{P}) \quad & \text{s.t.} \quad \sum_{p \in P} f_p(t - t_{e,p}) \delta_{e,p} \leq u_e, \quad \forall t \in \{1, \dots, T\}, \forall e \in E \\
 & \quad \sum_{t \in \{1, \dots, T\}} \sum_{p \in P_k} f_p(t) \geq b_k, \quad \forall k \in K \\
 & \quad f_p(t) \geq 0, \quad \forall p \in P, 1 \leq t \leq T - t_p
 \end{aligned} \tag{2.1}$$

* Note that, of course, one arc may be part of two or more paths serving the same or different commodities.

The first family of capacity constraints considers the times taken by the flows traveling on different paths until arc e . More precisely, a unit of flow reaching *the head* v of arc $e = (u, v)$ at time t , using path $p \in P_k$, left its origin s_k at time $t - t_{e,p}$. The second family of constraints imposes the demand to be fulfilled. Linear program (2.1) is non-compact: With respect to the input size, there are exponentially many variables. Furthermore the number of constraints is pseudo-polynomial since it depends on the time horizon T . It is therefore worthwhile to devise a column generation approach in order to provide an exact solution for \mathcal{P} .

3. Column generation subproblem

Suppose $\bar{P}_k \subset P_k$ is the restricted set of paths for commodity $k \in K$ used so far in the primal. If we associate variables $w_e(t)$, $(t, e) \in \{1, \dots, T\} \times E$, to primal capacity constraints and variables σ^k , $k \in K$, to demand constraints, the column generation subproblem consists of finding a path $p \in P_k \setminus \bar{P}_k$ such that $c_p + \sum_{e \in p} w_e(t + t_{e,p}) < \sigma^k$ for all $1 \leq t \leq T$, $k \in K$, and $e \in E$.

Fix $t \in \{1, \dots, T\}$ and let $t_{u,p}$ be the *arrival time* at node u along path p and observe that $t + t_{(u,v),p} = t_{u,p} + t_{(u,v)}$. Note that, due to the structure of the original (primal) problem, we do not have to consider waiting times at intermediate nodes. Hence, we search for a path $p \in P_k \setminus \bar{P}_k$ such that, for all $e \in E$:

$$\sum_{(u,v) \in p} c_{u,v} + w_{u,v}(t_{u,p} + t_{(u,v)}) < \sigma^k$$

This is in fact a special shortest-path problem: *Find a minimum cost path in a network where the cost of any arc (u, v) is time-dependent and arc traversal times are constant values.* Unfortunately, the following negative result holds:

Theorem. The column generation subproblem is binary NP-hard.

However, we devise a *dynamic programming* solving the column generation subproblem that runs in (pseudo-polynomial) time $O(|V|^3 T)$. It is worthwhile to point out that the dynamic programming algorithm looks for the minimum cost path from each node to one destination, for each $t \in \{1, \dots, T\}$.

4. Preliminary Results

Preliminary computational results, performed on a standard PC, show that the approach presented here is promising. It is reasonably effective against increasing networks size and efficient in terms of CPU times. We are able to optimally solve

in few seconds instances of the problem with $T \leq 100$, $k \leq 10$ and hundreds of nodes.

References

- [1] M. S. Alex Hall, Steffen Hippler. Multicommodity flows over time: Efficient algorithms and complexity. *Theoretical Computer Science*, 379:387–404, 2007.
- [2] J. E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20:1–66, 1989.
- [3] G. J. W. B. Klinz. Minimum-cost dynamic flows: the series parallel case. *Networks*, 43:153–162, 2004.
- [4] D. Gale. Transient flows in networks. *Michigan Mathematical Journal*, 6:50–63, 1959.
- [5] H. Hamacher and S. A. Tjandra. Mathematical modelling of evacuation problems: A state of the art. In *Pedestrian and Evacuation Dynamics*, 2002.
- [6] L. R. F. Jr. and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6(3):419–433, 1958.
- [7] L. R. F. Jr. and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
- [8] B. Kotnyek. An annotated overview of dynamic network flows. Technical report, INRIA, 2003.
- [9] M. S. L. Fleischer. Quickest flows over time. *SIAM Journal on Computing*, 36(6):1600–1630, 2007.
- [10] M. S. Lisa Fleisher. Minimum cost flows over time without intermediate storage. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 66–75, 2003.
- [11] E. Minieka. Maximal, lexicographic and dynamic network flows. *Operations Research*, 21:517–527, 1973.
- [12] B. K. Rainer E. Burkard, Karin Dlaska. The quickest flow problem. *Methods and Models of Operations Research*, 37:31–58, 1993.
- [13] W. L. Wilkinson. An algorithm for universal maximal dynamic flows in a network. *Operations Research*, 19:255–266, 1971.

Inferring Update Sequences in Boolean Gene Regulatory Networks

Fabien Tarissan,^a Camilo La Rota^b

^a*Complex System Institute (ISC) & CNRS, Palaiseau, France*
tarissan@lix.polytechnique.fr

^b*Complex System Institute (IXXI), Lyon, France*
camilo.larota@ens-lyon.fr

Key words: Mathematical programming, Inverse problems, Gene regulatory networks reconstruction

1. Introduction

This paper employs mathematical programming and mixed integer linear programming techniques for solving a problem arising in the study of genetic regulatory networks. More precisely, we solve the inverse problem consisting in the determination of the sequence of updates in the digraph representing the gene regulatory network (GRN) of *Arabidopsis thaliana* in such a way that the generated gene activity is as close as possible to the observed data.

Differences among cells of different tissues depend on the specific set of genes that are active in each tissue. Therefore, one usually assumes that the different steady states of a GRN dynamics correspond to the different possible cell fates ([7]). This leads to explain the changes observed during the development of the organisms by the fact that perturbations on specific elements of the network make the system switch from one steady state to another. Some hypothesis can be made about these perturbations, which are then treated as initial conditions for the new tissue being formed. However, an important unknown is (are) the update sequence(s) of the gene activity that let the system evolve from a given set of initial conditions to the set of steady states. Indeed, different update sequences determine different sets of basins of attraction of the GRNs. However, the steady states remain the same under any sequence.

Usually, a specific update sequence is assumed to rule the dynamics of the GRNs [1; 3]. The present study differs from this approach in that we sought to *infer* the update sequence from the biological observations. It also differs from our previous paper as we focus here on asynchronous sequences whereas in [6] the updates were synchronous.

2. The problem

Given a directed graph $G = (V, E)$, a discrete set T of time instants (which we suppose to be an initial contiguous proper subset of \mathbb{N}) and the following functions:

- a function $\alpha : E \rightarrow \{+1, -1\}$ called the *arc sign function*;
- a function $\omega : E \mapsto \mathbb{R}_+$ called the *arc weight function*;
- a function $\chi : V \times T \mapsto \{0, 1\}$ called the *gene state function*;
- a function $\iota : V \mapsto \{0, 1\}$ called the *initial configuration*;
- a function $\theta : V \mapsto \mathbb{R}$ called the *threshold function*;
- a function $\gamma : V \times T \mapsto \{0, 1\}$ called the *updating function*.

A *gene regulatory network* (GRN) is a 8-tuple $(G, T, \alpha, \omega, \theta, \chi, \iota, \gamma)$ such that:

$$\forall v \in V \quad \chi(v, 0) = \iota(v) \quad (2.1)$$

$$\forall v \in V, t \in T \setminus \{0\} \quad \chi(v, t) = \begin{cases} H(v, t-1) & \text{if } \gamma(v, t) = 1 \\ \chi(v, t-1) & \text{otherwise} \end{cases} \quad (2.2)$$

where H is the *Heaviside* function defined for $v \in V$ and $t \in T$ by

$$H(v, t) = \begin{cases} 1 & \text{if } \sum_{u \in \delta^-(v)} \alpha(u, v) \omega(u, v) \chi(u, t) \geq \theta(v) \\ 0 & \text{otherwise,} \end{cases} \quad (2.3)$$

with $\delta^-(v) = \{u \in V \mid (u, v) \in E\}$ for all $v \in V$. Eqns. (2.1)-(2.2)-(2.3) together are called the *evolution rules* of the GRN. For any particular $t \in T$, $\chi(\cdot, t) : V \rightarrow \{0, 1\}$ is called a *configuration*. Since the evolution rules relate a configuration at time t with a configuration at time $t-1$, $\chi(\cdot, t)$ is called a *fixed configuration* (or fixed point) if it remains invariant under the application of one complete cycle of updates encoded by γ . Furthermore, as long as the evolution rules are purely deterministic (as is modelled above), a fixed point of a GRN is determined by its initial configuration.

In this paper we deal with an inverse problem related to the estimation of update sequence in GRNs. More precisely, we address the following.

UPDATE SEQUENCE ESTIMATION IN GRNS (USEGRN). Given a digraph G , a time instant set T , an arc sign function α , an arc weight function ω , a threshold function θ and a set I of initial configurations, find an update function γ with the property that for all $\iota \in I$ there exists a gene activation function χ such that $(G, T, \alpha, \omega, \theta, \chi, \iota, \gamma)$ are GRNs whose fixed points are at a minimum distance to observed data.

In other words, we attempt to estimate the sequence of updates in a GRN from the knowledge of the digraph topology in such a way that (a) the GRN evolution rules

are consistent with respect to a certain set of initial configurations and (b) the fixed points induced by the estimated values are as close as possible to the observed ones. As the reader might notice, the problem strongly depends on the modelling of the update sequence encoded by γ . In [3], the authors proposed to describe such a sequence by means of *periods* and *delays* parameters for each gene. Assuming p_v and d_v to be such values for gene v , we can reformulate Equation 2.2 in the previous modellisation according to the following relation:

$$\forall t \in T \quad \gamma(v, t) = 1 \iff \exists n \in \mathbb{N} \text{ s.t. } t = np_v + d_v$$

3. The mathematical programming formulation

The methodology we shall follow is that of modelling the USEGRN by means of a mathematical programming formulation:

$$\left. \begin{array}{l} \min_x f(x) \\ \text{subject to } g(x) \leq 0, \end{array} \right\}$$

where $x \in \mathbb{R}^n$ are the *decision variables* and $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the *objective function* to be minimized subject to a set of constraints $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ which may also include variable ranges or integrality constraints on the variables.

The primary concern in solving the USEGRN is thus modellistic rather than algorithmic. One of the foremost difficulties is that of employing a static modelling paradigm — such as mathematical programming — in order to describe a problem whose very definition depends on time. Another important difficulty resides in describing the necessary and sufficient conditions for a configuration to be a fixed point in a mathematical form. We solve this difficulty by introducing two decision variables: a binary variable s stating that the network has been stable for at least two successive time steps; a binary variable y that will indicate the first time the network is stable. The last difficulty concerns the proper modelling of the update sequence as proposed in [3]. The solution relies on the use of two binary variables π and δ for each gene and indexed over the possible values for the periods and the delays. Then, $\pi_v(p)$ (resp. $\delta_v(d)$) is set to 1 if the period (resp. delay) of v is p (resp. d). We provide above such a formulation:

- *Sets*: V of genes in the network, E of edges in the network, T of time instants, P of periods values, D of delay values and R of regions.
- *Parameters*:
 - $\iota : R \times V \mapsto \{0, 1\}$ is the initial configuration of the network (vector of boolean values affected to the genes) for each region.
 - $\alpha : A \rightarrow \{+1, -1\}$ is the sign of the arc weights;
 - $w : V \mapsto \mathbb{R}_+$ is the arc weight function;
 - $\theta : V \mapsto \mathbb{R}$ is the threshold function;
 - $\phi : V \times R \mapsto \{0, 1\}$ is the targeted fixed configuration for region r .
- *Variables*:

- for all $r \in R, v \in V, t \in T, x_{r,v}^t \in \{0, 1\}$ is the activation state of gene v at time t in region r ;
 - for all $r \in R, v \in V, t \in T, h_{r,v}^t \in \{0, 1\}$ is the projection of state of gene v at time t in region r according to Heaviside function;
 - $s : R \times T \mapsto \{0, 1\}$ is a decision variable indicating that the network is stable during at least two successive time steps in region r .
 - $y : R \times T \mapsto \{0, 1\}$ is a decision variable that indicates the first time the network reaches a stable state in region r .
 - for all $v \in V, p \in P, \pi_{v,p} \in \{0, 1\}$ is a decision variable that indicates that the periodicity of gene v is p .
 - for all $v \in V, d \in D, \delta_{v,d} \in \{0, 1\}$ is a decision variable that indicates that the delay of gene v is d .
- *Objective function:*

$$\min \sum_{r \in R} \sum_{t \in T \setminus \{1\}} \left((y_r^{t-1} - y_r^t) \sum_{v \in V} |x_{r,v}^t - \phi_{r,v}| \right).$$

• *Constraints:*

- Heaviside function computation rule (for all $t \in T \setminus \{1\}, v \in V, r \in R$):

$$\theta_v h_{r,v}^t - |V|(1 - h_{r,v}^t) \leq \sum_{u \in \delta^-(v)} \alpha_{uv} w_{uv} x_{r,u}^{t-1} \leq (\theta_v - 1)(1 - h_{r,v}^t) + |V| h_{r,v}^t$$

- state transition rules (for all $r \in R, v \in V, p \in P, d \in D$):

$$\begin{aligned} x_{r,v}^0 &= \iota_{r,v} \\ \forall t \in T \setminus \{1\} \text{ s.t. } t \neq np + d & \quad \pi_{v,p} \delta_{v,d} x_{r,v}^t = \pi_{v,p} \delta_{v,d} x_{r,v}^{t-1} \\ \forall t \in T \setminus \{1\} \text{ s.t. } t = np + d & \quad \pi_{v,p} \delta_{v,d} x_{r,v}^t = \pi_{v,p} \delta_{v,d} h_{r,v}^{t-1} \\ & \quad \pi_{v,p} \delta_{v,d} d \leq p \end{aligned}$$

- fixed point conditions (for all $r \in R, t \in T \setminus \{1\}$):

$$\begin{aligned} \sum_{v \in V} |x_{r,v}^t - x_{r,v}^{t-1}| &\leq \|V\| s_r^t & y_r^t f_r^t &= 0 & (1 - y_r^t) &\leq f_r^t \\ \sum_{v \in V} |x_{r,v}^t - x_{r,v}^{t-1}| &\geq s_r^t & \sum_{u > t} s_r^u &= f_r^t & (|P| + |D|)^2 &\leq \sum_{\tau \in T} y_r^\tau \end{aligned}$$

4. Reformulations and solutions

The above problem is a nonconvex Mixed-Integer Non-Linear Problem that can be reformulated exactly to a Mixed-Integer Linear Problem using the techniques proposed in [5]. After standard mathematical manipulations, all the nonlinearities reduce to product terms of binary and/or integer variables, which can be

reformulated by adding new auxiliary variables and constraints as follows:

xy terms (x, y : binary)	xz terms (x : binary, z : integer)
$\eta \geq 0$	$\zeta \geq z^L x$
$\eta \leq y$	$\zeta \leq z + (z^L + z^U)(1 - x)$
$\eta \leq x$	$\zeta \leq z^U x$
$\eta \geq x + y - 1$	$\zeta \geq z - (z^L + z^U)(1 - x)$

where z^L and z^U stand for the boundaries of z and η and ζ are the new variables that replace the products in the equations.

We solved to optimality a few real-life instances from the GRN of *Arabidopsis thaliana* using AMPL [2] to model the problem and CPLEX [4] to solve it. The size of the GRNs involved were such that CPLEX obtained the optimal solution in a matter of minutes.

5. Acknowledgements

This work was supported by EU FP6 FET Project MORPHEX. We are deeply grateful to L. Liberti (LIX) for his feedback on the mathematical formulation.

References

- [1] J. Demongeot, A. Elena, and S. Sené. Robustness in Regulatory Networks: a Multi-Disciplinary Approach. *Acta Biotheoretica*, 56(1-2):27–49, 2008.
- [2] R. Fourer and D. Gay. *The AMPL Book*. Duxbury Press, Pacific Grove, 2002.
- [3] Carlos Gershenson. Classification of random boolean networks. In Abbas Standish and Bedau, editors, *Artificial Life VIII, Proceedings of the Eighth International Conference on Artificial Life*, pages 1–8. MIT Press, 2002.
- [4] ILOG. *ILOG CPLEX 10.1 User's Manual*. ILOG S.A., Gentilly, France, 2006.
- [5] L. Liberti, S. Cafieri, and F. Tarissan. Reformulations in mathematical programming: A computational approach. In A. Abraham, A.-E. Hassanien, and P. Suarry, editors, *Global Optimization: Theoretical Foundations and Application*, Studies in Computational Intelligence. Springer, New York, to appear.
- [6] F. Tarissan, C. La Rota, and L. Liberti. Network reconstruction: a mathematical programming approach. In *Proceedings of the European Conference of Complex Systems (ECCS'08)*, to appear.
- [7] R Thomas and M Kaufman. Multistationarity, the basis of cell differentiation and memory. i. structural conditions of multistationarity and other nontrivial behavior. *Chaos*, 11(1):170–179, Mar 2001.

Bioinformatics

Lecture Hall A

Thu 4, 14:00–15:30

NANOCONES

A classification result in chemistry

Gunnar Brinkmann, Nico Van Cleemput

*Applied Mathematics and Computer Science, Ghent University,
Krijgslaan 281 S9, 9000 Ghent, Belgium*

{ Gunnar.Brinkmann, Nicolas.VanCleemput}@UGent.be

Key words: Nanocone, Enumeration, Classification

1. Introduction

Nanococones are carbon networks conceptually situated in between graphite and the famous fullerene nanotubes. Graphite is a planar carbon network where each atom has three neighbours and the faces formed are all hexagons. Fullerene nanotubes are discussed in two forms: once the finite, closed version where except for hexagons you have 12 pentagons and once the one-side infinite version where 6 pentagons bend the molecule so that an infinite tube with constant diameter is formed. A nanocone lies in the middle of these: next to hexagons it has between 1 and 5 pentagons, so that neither the flat shape of graphite nor the constant diameter tube of the nanotubes can be formed. Recently the attention of the chemical world in nanococones has strongly increased. Figure 1 shows an overview of these types of carbon networks.

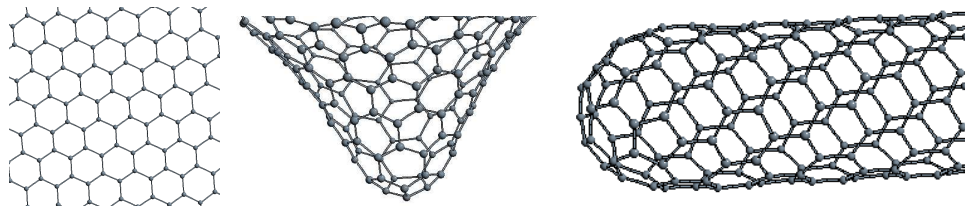


Fig. 1. graphite - nanocone - nanotube

The structure of graphite is uniquely determined, but for nanotubes and nanococones an infinite variety of possibilities exist. There already exist fast algorithms to generate fullerene nanotubes (see [3]) that are e.g. used to detect energetically possible nanotubes. In this talk we describe a generator for nanococones.

2. Patches

For computer generation of these structures we first need to describe them in a finite way. We describe the infinite molecule by a unique finite structure from

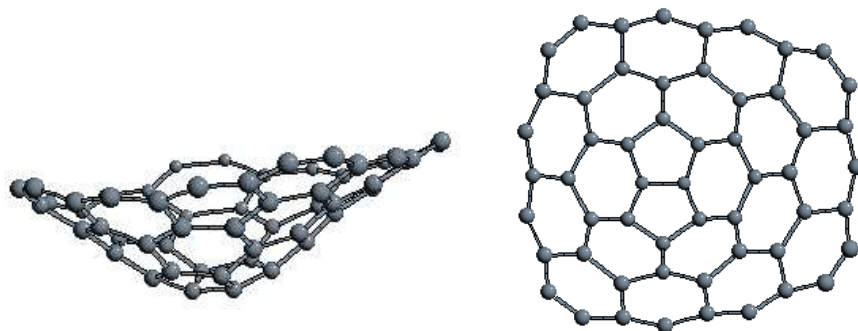


Fig. 2. Two views of a patch with two pentagons. which the cone can be reconstructed. The aim of this talk is to describe this step and give an idea of the algorithm to generate these finite representations.

A finite and 2-connected piece of a cone that contains all the pentagons is called a patch. All the vertices (atoms) in a cone have degree 3, so the vertices along the boundary of a patch will have degree 2 or 3. It can be easily shown that if the boundary of a patch doesn't contain any consecutive threes, then the number of neighbouring twos is equal to $6 - p$, where p is the number of pentagons in the patch.

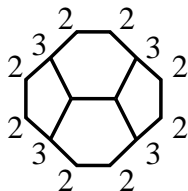


Fig. 3. A cone patch with boundary $(2(23)^1)^4$ and four neighbouring twos.

We can interpret patches without consecutive threes as polygons where the consecutive twos are the corners, and the lengths of the sides are determined by their number of threes.

3. Classification


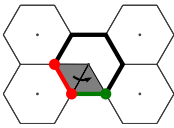

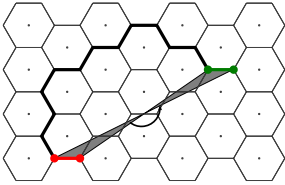

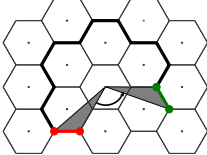

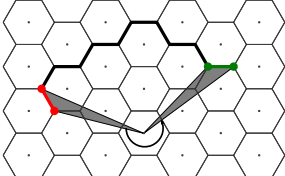

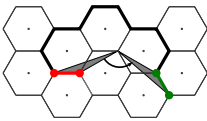

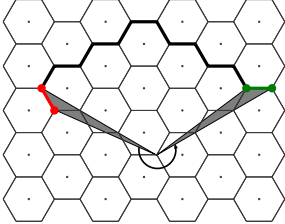

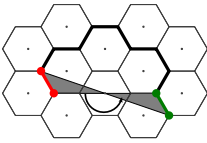

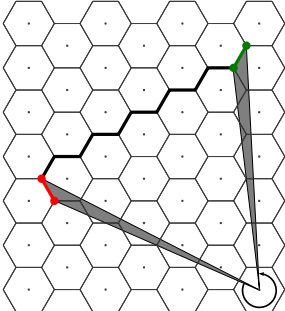
Definition 1. A **symmetric patch** is a patch that has a boundary of the form $(2(23)^k)^{6-p}$, with $1 \leq p \leq 5$.

A **nearsymmetric patch** is a patch that has a boundary of the form $(2(23)^{k-1})(2(23)^k)^{6-p-1}$, with $1 < p < 5$.

So in a symmetric patch all sides have an equal length and in a nearsymmetric patch all sides except one have an equal length and that one side is just one shorter than the others.

Theorem 3.1. All cones with 1 or 5 pentagons contain a symmetric patch, and all cones with 2, 3 or 4 pentagons contain a symmetric or a nearsymmetric patch.

Table 14. The complete classification of cone patches.

 $2^5 = (2(23)^0)^5$		 $2(23)^1(2(23)^2)^2$	
 $(2(23)^1)^4$		 $(2(23)^2)^2$	
 $2(23)^0(2(23)^1)^3$		 $2(23)^2 2(23)^3$	
 $(2(23)^1)^3$		 $2(23)^5$	

This result was first established in [5]. Here we sketch a proof that is not only shorter but can also easily be generalized to other periodic structures.

We interpret a nanocone as a disordered graphite lattice. Choosing a path around all disordering pentagons in the cone (described by right and left turns) and repeating the first edge at the end, and then following this path in the graphite lattice, the first and last edge in the resulting path don't agree anymore. It can be shown that the first edge and the last edge can be mapped onto each other by a **symmetry of the lattice** which is in fact a rotation by $p * 60^\circ$. This method to classify disordered patches was invented in [4], extended in [2] and in [1] it was shown that (under the circumstances described here), two disorders of the same tiling are isomorphic – except for a finite region – if and only if these symmetries are equivalent. Two rotations are said to be equivalent when they rotate among the same angle, and the centers of rotation are equivalent under a symmetry of the tiling.

In our case there are only a limited number of possibilities for these symmetries. They are all rotations and are depicted in Table 14. The patches in Table 14 are patches that correspond to these symmetries.

It is easily proven that adding or removing layers of hexagons does not change the type of the boundary, i.e. whether the boundary is symmetric or nonsymmetric.

So together with the theorem of Balke, this proves that all cones are equivalent to one of the cones obtainable from the patches in Table 14.

It also follows from Table 14 that no cone contains a symmetric and a nonsymmetric patch which both contain all the pentagons in the cone, because such boundaries correspond to different automorphisms.

Choosing the boundary that exists due to Theorem 3.1 in a shortest way leads to a unique patch that fully describes the nancone.

In fact this classification even leads to a unique patch, so we have the following theorem.

Theorem 3.2. There is a 1-1 correspondence between the set of symmetric and nonsymmetric patches and the set of nancones.

An algorithm to generate these patches will be sketched in the talk. It was implemented and tested against an independent algorithm to verify the results.

References

- [1] Ludwig Balke, *Classification of Disordered Tilings*, Annals of Combinatorics **1**, 1997, pp.297–311.
- [2] G. Brinkmann, *Zur mathematischen Behandlung gestrter Pflasterungen*, Ph.D. thesis, University of Bielefeld, 1990.
- [3] G. Brinkmann, U. von Nathusius and A.H.R. Palser, *A constructive enumeration of nanotube caps.*, *Discrete Applied Mathematics* **116**, 2002, pp. 55–71.
- [4] A.W.M. Dress, *On the Classification of Local Disorder in Globally Regular Spatial Patterns.*, *Temporal Order, Synergetics 29* Springer, 1985, pp. 61–66.
- [5] D.J. Klein, *Topo-combinatoric categorization of quasi-local graphitic defects*, Phys. Chem. Chem. Phys. **4**, 2002, pp.2099–2110.

The Molecular Distance Geometry Problem Applied to Protein Conformations

A. Mucherino,^a C. Lavor,^b N. Maculan^c

^a*LIX, École Polytechnique, Palaiseau, France*
mucherino@lix.polytechnique.fr

^b*Dept. of Applied Mathematics (IMECC-UNICAMP), State University of Campinas,
Campinas-SP, Brazil*
clavor@ime.unicamp.br

^c*Federal University of Rio de Janeiro (COPPE-UFRJ), C.P. 68511, 21945-970, Rio de
Janeiro - RJ, Brazil*
maculan@cos.ufrj.br

Key words: distance geometry, protein molecules, branch and prune

Molecules are sets of atoms that bond to each other forming particular three-dimensional structures, which can reveal important features of the molecules. One of the most used approaches to discover these structures is based on the Nuclear Magnetic Resonance (NMR). This is an experimental technique which is able to detect the distances between particular pairs of atoms of the molecule. Once a subset of distances between atoms has been obtained, the problem of identifying the coordinates of the considered atoms is known as the MOLECULAR DISTANCE GEOMETRY PROBLEM (MDGP) [3].

Many researchers worked on this problem and proposed different approaches. The most common approach is to formulate the MDGP as a continuous global optimization problem, in which the function to be minimized is a penalty function monitoring how much the known distances are violated in possible conformations of the atoms of the molecule. One of the most used objective functions is the Largest Distance Error (LDE):

$$LDE(\{x_1, x_2, \dots, x_n\}) = \frac{1}{|m|} \sum_{\{i,j\}} \frac{||x_i - x_j|| - d_{ij}}{d_{ij}},$$

where $\{x_1, x_2, \dots, x_n\}$ represents a conformation, d_{ij} is the known distance between the atom x_i and the atom x_j and m is the total number of known distances. If the subset of given distances is feasible, then the value of the LDE function in correspondence with a solution is 0.

We are studying a particular subclass of instances of the MDGP for which a combinatorial formulation can be supplied. Let $G = (V, E, d)$ be a weighted undirected graph, where V represents the set of atoms, edges in E indicate that the distances between the connected atoms are known, and the weights d represent the numeric value of the distances. As shown in [4; 5; 6; 7], if the following two assumptions are satisfied

- (i) E contains all cliques on quadruplets of consecutive atoms,
- (ii) consecutive vertices cannot represent perfectly aligned atoms,

for a given order in G , then the MDGP can be formulated as a combinatorial problem. We refer to this problem as the DISCRETIZABLE MOLECULAR DISTANCE GEOMETRY PROBLEM (DMDGP).

The DMDGP is NP-complete [4]. Moreover, it is interesting to note that the assumption (i) in the definition of the DMDGP is the tightest possible for the problem to be NP-complete. If E contains all cliques on *quintuplets*, and not only quadruplets, of consecutive atoms, then G is a *trilateration graph*. By [2], the MDGP associated to a trilateration graph can be solved in polynomial time. Therefore, graphs having cliques on sequences of 4 consecutive vertices correspond to an NP-complete problem, whereas graphs with cliques on sequences of 5 consecutive vertices bring to easy-to-solve problems. There is a sort of *barrier* after which the DMDGP becomes simple to solve. This is very interesting, because, when data from biology are considered, the corresponding DMDGP approaches to this barrier, but it is not able to go beyond and to be classified as an easy-to-solve problem.

We are particularly interested in *protein molecules*. Proteins are formed by smaller molecules called *amino acids*, that bond to each other by forming a sort of chain. Because of their particular structure, the MDGP related to protein molecules can be formulated as a combinatorial problem, because the aforementioned assumptions are satisfied, in most of the cases. Instances used to test the performances of approaches to the MDGP are usually artificially generated from the known conformations of some proteins. In [1; 8], for example, the used instances are generated by computing the distances between all the possible pairs of atoms of the molecule, and by keeping only the distances smaller than 6Å. This simulates instances obtained by NMR, because this technique is able to detect only distances which are not larger than 6Å. Currently, our attention is focused on protein backbones only, and therefore, in previous works, the considered atoms are limited to the backbone atoms only. In particular, the sequence of atoms N–C_α–C, defining the backbone of a protein, is considered.

A BRANCH & PRUNE (BP) algorithm [4] is used for solving the combinatorial problem efficiently (Algorithm 1). It is based on the exploration of a binary tree containing solutions to the problem. During the search, branches of the binary tree are pruned as soon as they are discovered to be infeasible. This pruning phase helps

Algorithm 1 The BP algorithm.

```
0: BP( $i, n, d$ )
  for ( $k = 1, 2$ ) do
    compute the  $k^{th}$  atomic position for the  $i^{th}$  atom:  $x_i$ ;
    check the feasibility of the atomic position  $x_i$ :
    if ( $|\|x_i - x_j\| - d_{ij}| < \varepsilon, \forall j < i$ ) then
      the atomic position  $x_i$  is feasible;
      if ( $i = n$ ) then
        a solution is found;
      else
        BP( $i + 1, n, d$ );
      end if
    else
      the atomic position  $x_i$  is pruned;
    end if
  end for
```

in reducing the binary tree quickly, so that an exhaustive search of the remaining branches is not computationally expensive. The computational experiments presented in [4; 6; 7] showed that the combinatorial approach can provide much more accurate solutions to the problem.

Our final aim is to be able to solve instances containing real data (i.e. data obtained by NMR) by the combinatorial approach. This is not trivial. Indeed, the artificially generated instances used in the experiments are still far from instances obtained by NMR. Indeed, the NMR is able to identify distances between hydrogen atoms only. Therefore, instances generated as explained above simulate real data only because of the rule of the 6Å threshold, but not for the kind of considered atoms.

Let us suppose that the sequence of atoms N–C $_{\alpha}$ –C (defining the protein backbones) and all the hydrogens H bonded to such atoms are considered. Let $G = (V, E, d)$ be the associated weighted undirected graph. Since only hydrogens are detected by NMR, there is an edge between two vertices only if both the vertices refer to hydrogens, or if one vertex represents a hydrogen and the other one is the carbon or the nitrogen bonded to it (bond lengths are known *a priori*). It follows that the subgraph G_H , such that $G \supset G_H = (V_H, E_H, d_H)$ and containing all the vertices in V to which at least two edges are associated, refers to hydrogen atoms only. Given an order on the vertices in V_H for which the assumptions (i) and (ii) are satisfied, the MDGP can be formulated as a DMDGP, and solved by the BP algorithm. From a chemical point of view, however, the solutions provided by BP are incomplete in this case, because they provide the coordinates of the hydrogen atoms only, whereas the sequence N–C $_{\alpha}$ –C is of interest. The problem is to find the coordinates of the atoms associated to the vertices of $G - G_H$ by exploiting the coordinates of the atoms associated to the vertices of G_H , and some distances

between pairs of vertices (v, v_H) , where $v \in V - V_H$ and $v_H \in V_H$. Suitable strategies for solving this problem are currently under investigation.

Acknowledgments

The authors would like to thank the Brazilian research agencies FAPESP and CNPq, the French research agency CNRS and École Polytechnique, for financial support. The authors also thank Leo Liberti for the fruitful comments to this work.

References

- [1] P. Biswas, K.-C. Toh, and Y. Ye, *A Distributed SDP Approach for Large-Scale Noisy Anchor-Free Graph Realization with Applications to Molecular Conformation*, SIAM Journal on Scientific Computing **30**, 1251–1277, 2008.
- [2] T. Eren, D.K. Goldenberg, W. Whiteley, Y.R. Yang, A.S. Morse, B.D.O. Anderson and P.N. Belhumeur, *Rigidity, Computation, and Randomization in Network Localization*, IEEE Infocom Proceedings, 2673–2684, 2004.
- [3] T.F. Havel, *Distance Geometry*, D.M. Grant and R.K. Harris (Eds.), Encyclopedia of Nuclear Magnetic Resonance, Wiley, New York, 1701-1710, 1995.
- [4] C. Lavor, L. Liberti, and N. Maculan, *Discretizable Molecular Distance Geometry Problem*, Tech. Rep. q-bio.BM/0608012, arXiv, 2006.
- [5] C. Lavor, L. Liberti, A. Mucherino, and N. Maculan, *Recent Results on the Discretizable Molecular Distance Geometry Problem*, Proceedings of the Conference ROADEF09, Nancy, France, February 10–12, 2009.
- [6] C. Lavor, L. Liberti, A. Mucherino and N. Maculan, *On a Discretizable Subclass of Instances of the Molecular Distance Geometry Problem*, Proceedings of the Conference SAC09, Honolulu, Hawaii, March 8–12, 2009.
- [7] L. Liberti, C. Lavor, and N. Maculan, *A Branch-and-Prune Algorithm for the Molecular Distance Geometry Problem*, International Transactions in Operational Research **15** (1), 1–17, 2008.
- [8] D. Wu and Z. Wu, *An Updated Geometric Build-Up Algorithm for Solving the Molecular Distance Geometry Problem with Sparse Distance Data*, Journal of Global Optimization **37**, 661–673, 2007.

Protein Threading

Guillaume Collet,^a Rumen Andonov,^b Nikola Yanev,^c
Jean-François Gibrat^b

^a*Symbiose team, IRISA, Campus de Baulieu, 35042 Rennes, France*

^b*INRA, Unité Mathématique, Informatique et Génome UR1077, F-78352 Jouy-en-Josas
Cedex, France*

^c*Faculty of Mathematics and Informatics, University of Sofia, 1164 Sofia, 5 James
Bourchier Blvd., Bulgaria*

Key words: Integer programming, combinatorial optimization, protein threading problem,
protein structure alignment

1. Introduction

The most important *in silico* methods, to exploit the amount of new genomic data, are based on the concept of homology. The principle of homology-based analysis is to identify a homology relationship between a new protein and a protein whose function is known. For remote homologs, sequence alignment methods fail. In such a case one aligns the sequence of a new protein with the 3D structures of known proteins. Such methods are called fold recognition methods or threading methods.

Lathrop & Smith [1] were the first to propose an algorithm based on a branch & bound technique providing the global alignment with the optimal score and to prove the problem to be NP-Hard. Since then, other methods have been developed that improved the efficiency of the sequence – structure global alignment algorithm ([2; 3; 4]).

This paper describes a new algorithm that expands upon algorithms proposed in previous works ([3; 4]) to allow implementation of *local* sequence – structure alignments. This allows threading methods to cover the whole spectrum of alignment types needed to analyze homologous proteins.

Our definition of alignments is based on the definition of the Protein Threading Problem (PTP) given in [1].

2. Outline of the Protein Threading Problem

Query Sequence and Structure Template: A query sequence is a string of length N over the 20-letter amino acid alphabet. A structure template is an ordered set M of m blocks which correspond to the secondary structure elements (SSEs). Block k has a fixed length of L_k amino acids. Let $I \subseteq \{(k, l) \mid 1 \leq k < l \leq m\}$ be the set of blocks interactions.

Alignment: An alignment of a structure template with a query sequence corresponds to positioning blocks of the template along the sequence. A *global* alignment requires that all blocks are aligned, preserve their order, and do not overlap. This alignment has been modeled by mixed integer programming (MIP) approach in [2; 3]. In this paper, we extend the model presented in [3].

3. Local alignments : towards better PTP models

Global alignment assumes that all blocks are aligned with the query sequence. However, it sometimes happens that some members of a protein family do not share exactly the same number of SSEs. An alignment which permits to omit blocks is called a *local* alignment. To solve this local alignment, we propose two models: (1) A compact model (CM) where we modify constraints to omit blocks. (2) An extended model (EM) where we add dummy positions for each block. When a dummy position is chosen, the block is omitted. These models are described very briefly below. For more details, the interested reader can refer to our research report [5].

3.1 Compact model

We define a digraph $G(V, A)$ with vertex set V and arc set A . Each vertex $(i, k) \in V$ represents block k at position i along the sequence. A block k can take $n_k = N - L_k + 1$ positions along the query sequence. A cost C_{ik} (resp. D_{ikjl}) is associated to each vertex (i, k) (resp each arc $((i, k), (j, l))$). Let y_{ik} (resp. z_{ikjl}) be binary variables associated with vertices (resp. arcs). Based on these notations, we obtain the following model:

$$\max \sum_{k=1}^m \sum_{i=1}^{n_k} C_{ik} y_{ik} + \sum_{((i,k),(j,l)) \in A} D_{ikjl} z_{ikjl} \quad (3.1)$$

Subject to:

$$y_{ik} \in \{0, 1\}, \quad k \in M, i \in [1, n] \quad (3.2)$$

$$0 \leq z_{ikjl} \leq 1, \quad ((i, k), (j, l)) \in A \quad (3.3)$$

$$\sum_{i=1}^{n_k} y_{ik} \leq 1, \quad k \in M \quad (3.4)$$

$$\sum_{j=i+L_k}^{n_l} z_{ikjl} - y_{ik} \leq 0, \quad (k, l) \in I, i \in [1, n_k] \quad (3.5)$$

$$\sum_{i=1}^{\min(j-L_k, n_k)} z_{ikjl} - y_{jl} \leq 0, \quad (k, l) \in I, j \in [1, n_l] \quad (3.6)$$

$$y_{ik} + \sum_{j=1}^{\min(n_k, i+L_k-1)} y_{jl} \leq 1, \quad 1 \leq k \leq l \leq m, i \in [1, n_k] \quad (3.7)$$

$$\sum_{i=1}^{n_k} y_{ik} + \sum_{i=1}^{n_l} y_{il} - \sum_{j=L_k+1}^{n_l} \sum_{i=1}^{j-L_k} z_{ikjl} \leq 1, \quad (k, l) \in I \quad (3.8)$$

Constraints (3.4) allow a block be aligned or not. Constraints (3.5) and (3.6) allow an arc, leaving (resp. entering) an activated vertex, be activated or not. Constraints (3.7) preserve the order of blocks. Finally, constraints (3.8) coerce the activation of an arc if its vertices are activated.

3.2 Extended Model

Denote by $d_{ik}, i \in [1, N], k \in [1, m]$ a variable which we call dummy variables. The objective function is given by (3.1). This model uses constraints (3.2), (3.3), (3.5), (3.6) and (3.8). Additional constraints are the following:

$$d_{ik} \in \{0, 1\} \quad k \in M, i \in [1, n_k] \quad (3.9)$$

$$\sum_{i=1}^{n_k} y_{ik} + \sum_{i=1}^N d_{ik} = 1 \quad k \in M \quad (3.10)$$

$$\sum_{i=1}^j d_{ik} + \sum_{i=1}^{\min(j, n_k)} y_{ik} - \sum_{i=1}^j d_{i, k-1} - \sum_{i=1}^{j-L_{k-1}} y_{i, (k-1)} \leq 0 \quad k \in [2, m], j \in N \quad (3.11)$$

Constraints (3.10) state that exactly one vertex (either real or dummy), must be activated in a column. Constraints (3.11) preserve the order of the blocks.

4. Results

Two indicators have been used, computation time and the relative gap (RG) between the solution of the relaxed problem (LP) and the optimal solution (OPT):

$RG = \frac{LP-OPT}{OPT}$. RG is a good indicator of the efficiency of the model since the smaller RG , the easier for the branch & bound algorithm to find the solution.

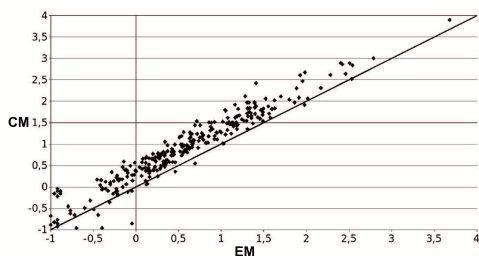


Fig. 1. Comparison of the computation times obtained by EM and CM. Each point represents an alignment. Times are expressed in seconds and are plotted using a base 10 logarithm scale.

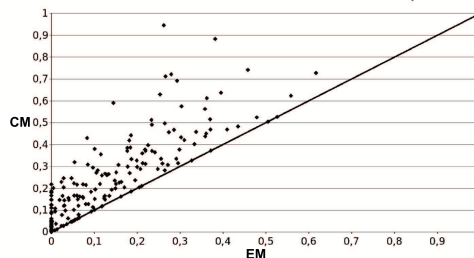


Fig. 2. Comparisons of relative gaps ($\frac{LP-OPT}{OPT}$) between models EM and CM. Each point is a sequence – structure alignment.

Figure 1 shows that EM is faster than CM for 99% of the instances. Moreover, Figure 2 shows that EM always gives a smaller RG than CM. It must be noted that LP relaxation directly gives the integer solution in 41% of the cases for the CM model and 52% of the cases for the EM model.

References

- [1] R.H. Lathrop and T.F. Smith. Global optimum protein threading with gapped alignment and empirical pair potentials. *Journal of Molecular Biology* 255:641-665 (1996).
- [2] J. Xu, et al. RAPTOR: optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology* 1(1):95-118 (2003).
- [3] R. Andonov, et al. Protein Threading Problem: From Mathematical Models to Parallel Implementations. *INFORMS Journal on Computing* 16(4):393:405 (2004).
- [4] R. Andonov, et al. Recent Advances in Solving the Protein Threading Problem. *Grids for Bioinformatics and Computational Biology*, E-G. Talbi and A. Zomaya, editors. Chapter 14, pages 325-356. Wiley-Interscience (2007)
- [5] Collet G., et al. Flexible Alignments for Protein Threading. *INRIA Research Report*, RR-6808 (2009).

Clustering

Lecture Hall B

Thu 4, 14:00–15:30

Cardinality Constrained Graph Partitioning into Cliques with Submodular Costs

J.R. Correa,^a N. Megow,^b R. Raman,^b K. Suchan^c

^a*Departamento de Ingeniería Industrial, Universidad de Chile, Chile*
correa@dim.uchile.cl

^b*Max-Planck-Institut für Informatik, Saarbrücken, Germany*
{nmegow, rraman}@mpi-inf.mpg.de

^c*Universidad Adolfo Ibáñez, Facultad de Ingeniería y Ciencias, Chile*
karol.suchan@uai.cl

Key words: graph partitioning, cardinality constraint, submodular functions

1. Introduction

We consider the problem of partitioning a graph into cliques of bounded cardinality. The goal is to find a partition that minimizes the sum of clique costs where the cost of a clique is given by a set function on the nodes. We present a general algorithmic solution based on solving the problem variant without the cardinality constraint. We yield constant factor approximations depending on the solvability of this relaxation for a large class of submodular cost functions. We give optimal algorithms for special graph classes.

More formally, we are given a simple graph $G = (V, E)$, a set function $f : 2^V \rightarrow \mathbb{R}^+$, and a bound $B \in \mathbb{Z}^+$. The problem is to find a partition of the graph G into cliques K_1, \dots, K_ℓ (the value of ℓ is not part of the input) of size at most B , that is, $|K_i| \leq B, i = 1, \dots, \ell$, such that the objective function $\sum_{i=1}^{\ell} f(K_i)$ is minimized. We denote our problem as *partition into cliques of bounded size* $\text{PCliq}(G, f, B)$.

Let V be a finite set. The function $f : 2^V \rightarrow \mathbb{R}$ is called *submodular* if for all subsets $A, B \subseteq V$ holds $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. We consider non-negative submodular functions that satisfy the following *exchange properties*. For all subsets $A, B \subseteq V$ with $f(A) \geq f(B)$ and elements $u, v \in V \setminus (A \cup B)$ with $f(u) \geq f(v)$ holds

$$f(A + u) + f(B + v) \leq f(A + v) + f(B + u). \quad (\text{E1})$$

Moreover, for all sets $A, B \subseteq V$ such $f(A) \geq f(B)$ and all elements $u \in V \setminus (A \cup B)$, holds that

$$f(A + u) \geq f(B + u). \quad (\text{E2})$$

This class of set functions contains well-known cost functions such as maximum function [7], chromatic entropy [2], probabilistic coloring [3], etc.

The problem $\text{PCliq}(G, f, B)$ is generally \mathcal{NP} -hard because it contains the classic \mathcal{NP} -hard problems *partition into cliques* (or *clique cover*) and *graph coloring* [5]. If the bound on the clique size B equals 2 then the problem corresponds to a maximum cardinality matching problem in G and can be solved optimally in polynomial time.

Graph partitioning and coloring problems (without cardinality constraints) are among the fundamental problems in combinatorial optimization. Various results are known for particular cost functions and graph classes. Recently, also generalized set functions have been considered in this context. Gijswijt, Jost, and Queyranne [6] introduce *value-polymatroidal* set functions which are *polymatroid rank functions* (i.e. nondecreasing, submodular, and $f(\emptyset) = 0$) that satisfy a slightly weaker exchange property than we require above. For every $A, B \subseteq V$ such that $f(A) \geq f(B)$ and every $u \in V \setminus (A \cup B)$, holds that $f(A + u) + f(B) \leq f(A) + f(B + u)$. They consider the problem $\text{PCliq}(G, f, \infty)$ and derive polynomial time algorithms for interval graphs and circular arc graphs. Fukunaga, Halldorsson, and Nagamochi [4] consider monotone concave cost functions and provide a general algorithmic scheme which yields a factor 4 approximation for perfect graphs. They also give a result for general graphs depending on the solvability of the *maximum independent set* problem on the complement of the graph.

Graph partitioning (or coloring) with a constraint on the clique size has been addresses rarely so far. Bodlaender and Jansen [1] investigated a special case of $\text{PCliq}(G, f, B)$ with the objective to minimize the number of cliques, that is, $f(S) = 1$ for all $S \subseteq 2^V$. They show that the decision variant of the problem on co-graphs is \mathcal{NP} -complete whereas it is polynomial time solvable on split graphs, bipartite graphs and interval graphs.

2. Our Results

2.1 Optimal algorithm for complete graphs and proper interval graphs

Consider the problem $\text{PCliq}(K, f, B)$ on a complete graph K . The following simple algorithm solves the problem optimally.

Algorithm PARTK

Order elements $u \in K$ in non-increasing order of $f(u)$ and group them greedily into sub-cliques of size B beginning with the elements of largest value.

Theorem 2.1. PARTK solves the problem $\text{PCliq}(K, f, B)$ on a clique K for submodular functions with exchange property optimally in polynomial time.

The exchange properties are indeed substantial to the result above. The problem of partitioning a complete graph is generally \mathcal{NP} -hard for submodular functions, and even for polymatroid rank functions.

Theorem 2.2. The problem $\text{PCliq}(K, f, B)$ on a clique K is NP-hard even if we restrict f to polymatroid rank functions, that is, non-decreasing, submodular functions with $f(\emptyset) = 0$.

Proof 1. Reduction from *graph partitioning* with unit node weights [5].

Additionally, we devise a dynamic program that solves the problem optimal for another special graph class.

Theorem 2.3. The problem $\text{PCliq}(G, f, B)$ on a proper interval graph G can be solved optimally in pseudopolynomial time for submodular functions with exchange property. If the number of distinct values for single elements $f(u)$ for all $u \in V$ is bounded, then this algorithm runs in polynomial time.

2.2 An $(c + 1)$ -approximation for general graphs

Our algorithmic framework is based on solving two relaxation of the given problem $\text{PCliq}(G, f, B)$. One relaxation concerns ignoring the graph structure, i.e., $\text{PCliq}(K, f, B)$ for K being a complete graph as considered above. In the other relaxation we assume that the cardinality of the cliques is not bounded, or $B \geq |V|$. We denote it as $\text{PCliq}(G, f, \infty)$. Optimal values for both relaxations considered individually may give arbitrarily bad lower bounds on an optimum solution. Still, we derive constant factor approximation guarantees when combining them. Consider the following polynomial time algorithm.

Algorithm PART

- (i) Solve the relaxation $\text{PCliq}(G, f, \infty)$ without cardinality restrictions.
- (ii) For all cliques K_i : Solve the problem $\text{PCliq}(K_i, f, B)$.

Theorem 2.4. Let f be a submodular function with exchange property. If there exists a c -approximation algorithm for the problem $\text{PCliq}(G, f, \infty)$ without cardinality constraint, then PART is a $(c + 1)$ -approximation for $\text{PCliq}(G, f, B)$.

Sketch of Proof: Let K_1, \dots, K_ℓ be the solution of the relaxed problem

PCLIQ(G, f, ∞). For each of the cliques K_i let $K_i^1, \dots, K_i^{\ell_i}$ denote the partition into subcliques (Step 2). Assume that all sets are indexed such that $f(K_i^j) \geq f(K_i^{j+1})$ for $j = 1, \dots, \ell_i - 1$. Then the value of the algorithm's solution is

$$\begin{aligned} \text{PART} &= \sum_{i=1}^{\ell} \sum_{j=1}^{\ell_i} f(K_i^j) = \sum_{i=1}^{\ell} f(K_i^1) + \sum_{i=1}^{\ell} \sum_{j=2}^{\ell_i} f(K_i^j) \\ &\leq c \text{OPT}(G, f, \infty) + \sum_{i=1}^{\ell} \sum_{j=2}^{\ell_i} f(K_i^j). \end{aligned}$$

The main effort lies in proving $\sum_{i=1}^{\ell} \sum_{j=2}^{\ell_i} f(K_i^j) \leq \text{OPT}(K, f, B)$. We first observe: For two sets $A, B \subseteq V$ with $|A| \geq |B|$ holds that if each element $v \in B$ can be mapped to a distinctive element in $u \in A$ such that $f(v) \leq f(u)$, then $f(A) \geq f(B)$. This fact combined with Algorithm PARTK and Theorem 2.1 allows us to apply a charging scheme where we map the elements of the cliques K_i^j with $i > 1$ to the optimal solution.

Since the set functions we consider are value-polymatroidal, can employ the optimal algorithm in [6] and yield a quite general result for interval graphs which are of particular interest in applications.

Corollary 2. There is a factor 2 approximation algorithm for PCLIQ(G, f, B) on interval graphs and circular arc graphs.

References

- [1] H. L. Bodlaender and K. Jansen. Restrictions of graph partition problems. Part I. *Theoretical Computer Science*, 148(1):93–109, 1995.
- [2] J. Cardinal, S. Fiorini, and G. Joret. Minimum entropy coloring. *J. Comb. Optim.*, 16(4):361–377, 2008.
- [3] F. D. Croce, B. Escoffier, C. Murat, and V. T. Paschos. Probabilistic coloring of bipartite and split graphs. In *ICCSA (4)*, volume 3483 of *Lecture Notes in Computer Science*, pages 202–211, 2005.
- [4] T. Fukunaga, M. M. Halldórsson, and H. Nagamochi. Robust cost colorings. In *SODA'08*, pages 1204–1212, 2008.
- [5] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.
- [6] D. Gijswijt, V. Jost, and M. Queyranne. Clique partitioning of interval graphs with submodular costs on the cliques. *RAIRO Oper. Res.*, 41:275–287, 2007.
- [7] S. V. Pemmaraju and R. Raman. Approximation algorithms for the max-coloring problem. In *ICALP'05*, volume 3580 of *Lecture Notes in Computer Science*, pages 1064–1075, 2005.

A simple MAX-CUT algorithm for planar graphs [★]

F. Liers, ^a G. Pardella ^a

^a*Institut für Informatik, Universität zu Köln, Pohligstraße 1, D-50969 Köln, Germany*
{liers,pardella}@informatik.uni-koeln.de

Key words: max-cut, planar graph, combinatorial algorithm, Kasteleyn city

Graph partitioning problems have many relevant real-world applications, e.g., VIA minimization in the layout of electronic circuits [1], or in physics of disordered systems [2]. In its most basic version, the task is to partition the nodes of a graph into two disjoint sets such that the weight of edges connecting the two sets is either minimum or maximum (assuming uniform weights in case the graph is unweighted). For general edge weights, the MAX-CUT problem is NP-hard. When restricting to certain graph classes, polynomial-time solution algorithms are known. This is true especially for planar graphs which are the subject of this extended abstract. In 1990 Shih, Wu, and Kuo [7] presented a mixed MAX-CUT algorithm for arbitrary weighted planar graphs. They solve the problem in time bounded by $O(|V|^{\frac{3}{2}} \log |V|)$ which is the best worst-case running time known to date. First, the dual graph is constructed which is then expanded in such a way that (optimum) matchings in the latter correspond to (optimum) cuts in the former. In this work, we follow this general algorithmic scheme which leads to an algorithm with the same worst-case running time $O(|V|^{\frac{3}{2}} \log |V|)$. However, in our procedure the expanded dual graph has a simpler structure and contains a considerably smaller number of both nodes and edges. As the bulk of the running time is spent in the matching computation and the latter scales with the size of the graph, our algorithm is much faster in practice. Our new MAX-CUT algorithm for arbitrary weighted planar graphs is a generalization of the methods proposed in [8; 6] which are based on the work of Kasteleyn [3] from the 1960s. This extended abstract bases upon a technical report [4] in which we both describe the algorithm in detail and present experimental results.

[★] Financial support from the German Science Foundation is acknowledged under contract Li 1675/1-1

1. The MAX-CUT Algorithm

In the following we assume we are given a planar embedding of G . At first, we calculate its dual graph $G_D = (V_D, E_D)$, where the weight of a dual edge is chosen as $w(\tilde{e}) = w(e)$ if $\tilde{e} \in E_D$ is the dual edge crossed by $e \in E$. Subsequently, we split all dual nodes $\tilde{v} \in V_D$ with degree $\deg(\tilde{v}) > 4$ into $\lfloor (\deg(\tilde{v}) - 1)/2 \rfloor$ nodes and connect the copies by a path of new edges receiving zero weight. Let *split nodes* denote nodes created by a splitting operation. Edges incident to the original node are equally distributed among the split nodes such that the degree of each node is at most four. We denote the resulting graph by $G_t = (V_t, E_t)$. It is easy to see that after the splitting operations, no node in G_t has a degree smaller than three or larger than four. The connectedness of G and G_D yields connectedness of G_t . Next, we expand each node in G_t to a K_4 subgraph (a so-called Kasteleyn city). Newly generated edges again receive weight zero, while edge weights from G_t are assigned to the corresponding edges in the expanded graph. We denote the resulting graph by G_E . Next, we calculate a minimum-weight perfect matching M

Algorithm 1 MAX-CUT algorithm for planar graphs

Require: Embedding of a simple, connected planar graph G

Ensure: MAX-CUT $\delta(Q)$ of G

1. Build dual graph G_D
 2. Split each node $v \in G_D$ with $\deg(v) > 4$ and call resulting graph G_t
 3. Expand each node $v \in G_t$ to a K_4 and call resulting graph G_E
 4. Compute a minimum-weight perfect matching M in G_E
 5. Shrink back all artificial nodes and edges while keeping track of matched dual edges
 6. Matched dual edges in G_D induce optimum Eulerian subgraphs and thus optimum max-cut $\delta(Q)$ of G
 7. **return** $\delta(Q)$
-

in G_E . Subsequently, we undo the expansion, i.e., shrink back all K_4 subgraphs and all (possibly created) split nodes, while keeping track of matched dual edges. Consider the subgraph of G_D induced by the matching edges that are still present in the dual after shrinking. Each node in this subgraph has even degree. Therefore, it is a minimum weight Eulerian graph. It is well known that there is a one-to-one correspondence between Eulerian subgraphs in the dual and cuts in its primal graph, see also Alg. 1 for a compact statement of the algorithm.

In order to prove correctness of the algorithm, we need to show that there always exists a perfect matching M in the expanded graph G_E . Then, we need to prove that the constructed perfect matching in G_E induces a subgraph in the dual in which all node degrees are even. We first show the latter. We call edges not contained in a K_4 *outgoing* and count the number of matched outgoing edges on some K_4 for an arbitrary perfect matching in G_E . Clearly, any possible perfect matching in a K_4 subgraph leads to either zero, two or four outgoing matching

edges. An odd number of outgoing matching edges always leaves an odd number of K_4 nodes unmatched, which contradicts the matching's perfectness. Shrinking back the artificial nodes to the corresponding split nodes does not affect the number of outgoing matching edges. Consequently, after having collapsed all split nodes back to its dual nodes, each dual node has an even number of adjacent matching edges, too. Hence the matching induced subgraph is Eulerian and therefore defines a cut $\delta(Q)$ in the original graph G . Let $w(M)$ be the weight of a minimum-weight perfect matching in G_E , which is the same weight of the Eulerian subgraph, therefore the weight of the induced Eulerian subgraph is minimum, and thus the weight of the cut $\delta(Q)$, too. We summarize this in the next theorem.

Theorem 17. The algorithm described above computes a MIN-CUT (or MAX-CUT) in an arbitrarily weighted planar graph.

The proof that the expanded graph G_E indeed has a perfect matching M is based on the following observations. G_E is connected and has an even number of nodes. A trivial perfect matching exists as in each K_4 all nodes can be covered by matching edges contained in the K_4 . Therefore, a perfect matching in G_E always exists. There also always exists another perfect matching in which not only artificial edges contained in the K_4 subgraphs are matched. This is reasonable due to the structure of the dual graph G_D , as any two adjacent nodes in G_D are connected by at least one simple cycle that is preserved during the expansion step. A possible nontrivial perfect matching in G_t may consist of those cycle edges and additionally of some artificial edges in each K_4 subgraph on the cycle. For all other Kasteleyn cities, edges contained in the K_4 can be matched.

For establishing running time bounds, we start with a given embedding of a planar graph. The geometric dual can be constructed in time $O(|V|)$. Furthermore, the described expansion of the dual graph can be done in time linear in $|V|$, and only $O(|V|)$ new nodes (edges) are created. Next, the most time consuming step is performed - the calculation of a minimum-weight perfect matching. Using a maximum-weight matching algorithm by Lipton and Tarjan [5], based on the planar separator theorem, together with an appropriate weight-function, the calculation of the minimum-weight perfect matching needs time $O(|V|^{\frac{3}{2}} \log |V|)$. Finally, all nodes blown up in the expansion are shrunk back in time $O(|V|)$. With these considerations, we state the following theorem.

Theorem 18. Using the method described above, a MIN-CUT (or MAX-CUT) in a planar graph can be determined in time bounded by $O(|V|^{\frac{3}{2}} \log |V|)$.

Space may become a crucial factor especially for large input graphs. Our method is less space demanding than the construction of [7]. This is important as the matching part depends on the graph size and is the bottleneck in the algorithm. The latter implies that our algorithm is faster in practice. Let F denote the set of faces of a maximum planar graph. Our method constructs a graph G_E with

at most $|V_E| = 4|F| = 4(2|V| - 4)$ nodes, and $6|F| + |E_D| = 15|V| - 30$ edges. The procedure by [7] generates for each dual node a “star” subgraph of seven nodes and nine edges. Thus yields an expanded dual graph with $7(2|V| - 4)$ nodes and $21|V| - 42$ edges. These bounds are sharp as the first step of Shih, Wu, and Kuo is always a triangulation of the graph. Our procedure, in comparison, computes a matching on a much smaller and sparser graph, even in the case the graph is a triangulation. Moreover, the practical running time of the method stated in [7] might increase, as the matching induced even-degree edge set may be empty in which case an additional $O(|V|)$ time step is needed to compute a nontrivial even-degree edge set.

It turns out that with our implementation of the presented algorithm we can compute MAX-CUTS in planar graphs with up to 10^6 nodes within reasonable time.

References

- [1] BARAHONA, F., GRÖTSCHEL, M., JÜNGER, M., AND REINELT, G. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research* 36, 3 (1988), 493–513.
- [2] HARTMANN, A. K., AND RIEGER, H. *Optimization Algorithms in Physics*. Wiley-VCH, 2002.
- [3] KASTELEYN, P. W. Dimer statistics and phase transitions. *Journal of Mathematical Physics* 4, 2 (1963), 287–293.
- [4] LIERS, F., AND PARDELLA, G. A simple max-cut algorithm for planar graphs. Tech. rep., Combinatorial Optimization in Physics (COPhy), Sep. 2008, <http://www.zaik.uni-koeln.de/~paper/preprints.html?show=zaik2008-579>.
- [5] LIPTON, R. J., AND TARJAN, R. E. Applications of a planar separator theorem. *SIAM J. Comput.* 9, 3 (1980), 615–627.
- [6] PARDELLA, G., AND LIERS, F. Exact ground states of large two-dimensional planar Ising spin glasses. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)* 78, 5 (2008), 056705.
- [7] SHIH, W. K., WU, S., AND KUO, Y. S. Unifying maximum cut and minimum cut of a planar graph. *IEEE Trans. Comput.* 39, 5 (1990), 694–697.
- [8] THOMAS, C. K., AND MIDDLETON, A. A. Matching Kasteleyn cities for spin glass ground states. *Physical Review B (Condensed Matter and Materials Physics)* 76, 22 (2007), 220406(R).

***k*-hyperplane clustering problem: column generation and a metaheuristic**

Edoardo Amaldi,^a Stefano Coniglio,^a Kanika Dhyani^b

^a*Dipartimento di Elettronica e Informazione, Politecnico di Milano
Piazza L. da Vinci 32, 22133, Milano
{amaldi, coniglio}@elet.polimi.it*

^b*Neptunus, Politecnico di Milano
Via G. Durando 10, 20158, Milano
dhyani@elet.polimi.it*

Key words: hyperplane clustering, column generation, metaheuristic

1. Introduction

In the *k*-Hyperplane Clustering problem (*k*-HC), given a set of m points $P = \{\underline{a}_1, \dots, \underline{a}_m\}$ in \mathbb{R}^n , we have to determine k hyperplanes $H_j = \{\underline{a} \in \mathbb{R}^n \mid \underline{w}_j^T \underline{a} = \gamma_j, \underline{w}_j \in \mathbb{R}^n, \gamma_j \in \mathbb{R}\}$, with $1 \leq j \leq k$, and assign each point to a single H_j , thus partitioning P into k *h-clusters*, so as to minimize the sum-of-squared 2-norm orthogonal distances d_{ij} from each point to the corresponding hyperplane, where $d_{ij} = \frac{|\underline{w}_j^T \underline{a}_i - \gamma_j|}{\|\underline{w}_j\|_2}$.

k-HC naturally arises in many areas such as data mining [3], operations research [7], line detection in digital images [2] and piecewise linear model fitting [4]. *k*-HC is \mathcal{NP} -hard, since it is \mathcal{NP} -complete to decide whether k lines can fit m points in \mathbb{R}^2 with zero error [7]. In [3] Bradley and Mangasarian propose a heuristic for *k*-HC which extends the classical *k*-means algorithm to the hyperplane case.

The bottleneck version of *k*-HC in which, given a maximum deviation tolerance ϵ , k is minimized, has been studied in [5]; it is closely related to the MIN-PFS problem of partitioning an infeasible linear system into a minimum number of feasible subsystems [2]. Variants of *k*-HC where linear subspaces of different dimensions are looked for are also considered, e.g., in [1].

In this work we propose a column generation algorithm and an efficient metaheuristic.

2. Column generation algorithm (CG)

We consider the following set covering master problem (MP):

$$\min \sum_{s \in \mathcal{S}} d_s y_s \quad (2.1)$$

$$s.t. \sum_{s \in \mathcal{S}} \tilde{x}_{is} y_s \geq 1 \quad 1 \leq i \leq m \quad (2.2)$$

$$\sum_{s \in \mathcal{S}} y_s \leq k \quad (2.3)$$

$$y_s \in \{0, 1\} \quad s \in \mathcal{S},$$

where \mathcal{S} is the set of (exponentially many) feasible h-clusters and, for each $s \in \mathcal{S}$, the variable y_s equals 1 if the h-cluster s is in the solution and 0 otherwise. The parameter d_s is the sum-of-squared 2-norm orthogonal distances to the hyperplane H_s of the points contained in the h-cluster s and the parameter \tilde{x}_{is} equals 1 if the point \underline{a}_i is contained in h-cluster s , and 0 otherwise.

We tackle this formulation with a column generation approach. Let $\mathcal{S}' \subset \mathcal{S}$ be the initial pool of columns. Let π_i and μ be the dual variables of constraints (2.2) and (2.3), corresponding to an optimal solution of (MP) when restricted to the only columns in \mathcal{S}' and with the integrality constraints relaxed. The column $s \notin \mathcal{S}'$ with the largest negative reduced cost $\bar{c}_{s'}$, with $\bar{c}_{s'} = d_{s'} - \sum_{i=1}^m \pi_i x_{is'}$, can be obtained by solving the following nonlinear 2-norm pricing problem (PP):

$$\min \sum_{i=1}^m ((\underline{w}^T \underline{a}_i - \gamma)^2 - \pi_i) x_i - \mu \quad (2.4)$$

$$s.t. \|\underline{w}\|_2 \geq 1 \quad (2.5)$$

$$x_i \in \{0, 1\} \quad 1 \leq i \leq m \quad (2.6)$$

$$\underline{w} \in \mathbb{R}^n, \gamma \in \mathbb{R},$$

where (\underline{w}, γ) are the parameters of $H_{s'}$ and the binary variable x_i is equal to 1 if the point \underline{a}_i is assigned to $H_{s'}$ and 0 otherwise. Note that x_i takes integer values in any optimal solution and hence (2.6) can be relaxed into $x_i \in [0, 1]$. (PP) is nonconvex due to (2.5) and can be solved to local optimality with state-of-the-art nonlinear programming solvers such as SNOPT.

Since $\sqrt{n} \|\underline{w}\|_2 \geq \|\underline{w}\|_1$, substituting $\|\underline{w}\|_1 \geq 1$ for (2.5) yields $\|\underline{w}\|_2 \geq \frac{1}{\sqrt{n}}$ and thus a relaxation of k -HC. This 1-norm constraint can be linearized with standard techniques, see [5]. Given any 1-norm solution, a corresponding feasible 2-norm solution can be obtained by fixing the point-to-hyperplane assignment and recomputing the hyperplane parameters in the closed-form proposed in [3].

To speed up convergence, a dual-stabilization technique [8] is used. At each iteration t that is a multiple of the frequency parameter f , the current dual vector

$\underline{\pi}_t$ is replaced by a convex combination $\underline{\pi}'_t$ of $\underline{\pi}_t$ and the previous dual vector $\underline{\pi}_{t-1}$, namely $\underline{\pi}'_t := \eta \underline{\pi}_t + (1 - \eta) \underline{\pi}_{t-1}$ with $\eta \in (0, 1)$.

3. Point-Reassignment metaheuristic (PR)

Our Point-Reassignment metaheuristic (PR) relies on a simple criterion to identify, at each iteration, points which are likely to be *ill-assigned* in the current solution, based on the distance ratio $\frac{d_{ij}}{\min_{j' \neq j} d_{ij'}}$.

Starting from a randomly generated solution, at each iteration the set I of possibly ill-assigned points is identified as follows. Let m_j , with $1 \leq j \leq k$, be the number of points currently assigned to hyperplane H_j and rank them w.r.t. the ratio $\frac{d_{ij}}{\min_{j' \neq j} d_{ij'}}$. Indeed, points with large ratio have larger distance w.r.t. H_j and are close to another hyperplane $H_{j'}$, hence being more likely to be ill-assigned. Given a control parameter α (“temperature”), the set I then contains the $\alpha \cdot m_j$ points of each cluster with the largest ratio.

A move consists in assigning each point \underline{a}_i in I to the closest hyperplane which differs from the current one \underline{a}_i is assigned to, and in assigning the points in $P \setminus I$ to the closest H_j , if it is not already the case. The hyperplane parameters are then recomputed in the closed-form described in [3]. To avoid cycling and try to escape from local minima, we adopt two Tabu Search features (see e.g. [6]): a list of forbidden moves of length l and a partial aspiration criterion.

Since early solutions are expected to have a larger number of ill-assigned points, the control parameter α is initially set to a large enough value α_0 and is then progressively decreased to stabilize the search process, thus progressively reducing the variability that is introduced at each iteration. More precisely, α is updated as $\alpha_t = \alpha_0 \rho^t$, where t is the index of the current iteration and $\rho \in (0, 1)$ determines the speed at which α is driven to 0. When $\alpha = 0$, I becomes empty and, after all points are assigned to the clusters H_j , the algorithm terminates in a local minimum. The best solution found is stored and returned.

4. Computational results and conclusions

We compare CG and PR with a multi-start version of Bradley and Mangasarian’s algorithm (BM, [3]) on a set of challenging, randomly generated instances [4]. CG is implemented in AMPL using SNOPT and CPLEX as solvers. PR and BM are implemented in C++. Tests are run on an Intel Xeon machine, with 2.8 GHz and 2 GB RAM, equipped with Linux and gcc 4.1.2.

CG is tested on 8 instances with $m = 20 - 70$, $n = 2 - 3$ and $k = 3 - 6$. η is set to 0.7 and is reduced to 0.4 when 90% of the dual variables become zero. The frequency f is set to 5. Because of the nonlinearities in the 2-norm pricing problem, CG with 2-norm gets often stuck in local minima and leads to poor quality solutions. CG with 1-norm finds solutions with very small objective function values, but since the 1-norm pricing problem is a non-trivial mixed-integer linear program, the overall computation time scales poorly with the size of the instances.

PR is tested on 95 instances with $m = 100 - 2500$, $n = 2 - 6$ and $k = 3 - 8$. Parameters are set to $\alpha_0 = 0.9$, $\rho = 0.6$, $l = 2$. We compare the best solutions found by running PR and BM for a fixed amount of time and restarting them from randomly generated solutions each time a local minimum is found. The time limit is set to 120 and 180 seconds for instances with up to, respectively, 750 points and 2500 points. PR finds better results in 89 cases out of 95 (with strictly better solutions in 59 cases). On average, BM yields solutions worse than those found by PR by a factor of 25%. Neglecting the 30 instances for which both algorithms find solutions of the same value (since they may be optimal), the factor amounts to 35.5%.

References

- [1] P. K. Agarwal and C. M. Procopiuc, *Approximation algorithms for the projective clustering*, Journal of Algorithms **46** (2003), no. 2, 115–139.
- [2] E. Amaldi and M. Mattavelli, *The MIN PFS problem and piecewise linear model estimation*, Discrete Appl. Math. **118** (2002), 115–143.
- [3] P.S. Bradley and O.L. Mangasarian, *k-plane clustering*, J. of Global Optimization **16** (2000), no. 1, 23–32.
- [4] S. Coniglio and F. Italiano, *Hyperplane clustering and piecewise linear model fitting*, Master’s thesis, DEI, Politecnico di Milano, 2007.
- [5] K. Dhyani, *Optimization models and algorithms for the hyperplane clustering problem*, Ph.D. thesis, Politecnico di Milano, 2009.
- [6] F. Glover and M. Laguna, *Tabu search*, Kluwer Academic Publishers, 1997.
- [7] N. Megiddo and A. Tamir, *On the complexity of locating linear facilities in the plane*, Operations Research Letters **1** (1982), 194–197.
- [8] A. Pessoa, E. Uchoa, M. Poggi de Aragão, and R. Rodrigues, *Algorithms over arc-time indexed formulations for single and parallel machine scheduling problems*, Tech. Report 8, Universidade Federal Fluminense, 2008.

Games

Lecture Hall A

Thu 4, 15:45–16:15

A System-theoretic Model for Cooperation and Allocation Mechanisms

Ulrich Faigle, Jan Voss

Universität zu Köln

Zentrum für Angewandte Informatik

Weyertal 80, D-50931 Köln, Germany {faigle,voss}@zpr.uni-koeln.de

Key words: Cooperation, system, allocation mechanism, value, randomization, symmetry, core, Weber set

1. Introduction

The classical model of cooperative games assumes that arbitrary subsets of agents can join to form feasible coalitions and create values in a given economic context. The main problem is: How should a commonly generated value be distributed among the agents?

Weber [13] developed a model of so-called probabilistic values (including the Shapley value, the Banzhaf value etc.) for the classical model.

However many real problems are not covered by the classical model. The notions of cooperation and allocation have often a more dynamic flavor. In the past there were many approaches that aimed for a suitable generalization of the classical model. For example Kalai and Samet [10] studied games with block structure, i.e., certain critical coalitions partition the set of agents. Models for cooperative games under precedence constraints were developed by Derks and Gilles [8] and Faigle and Kern [9]. Bilbao *et al.* [1; 2; 3; 4; 5; 6] have studied models for cooperative games with underlying combinatorial coalition structures such as convex geometries, antimatroids or matroids. In all these generalizations, analogues of Shapley's [12] classical value and possibly also the core are sought.

Our present research wants to take a first step towards viewing cooperation and allocation as dynamic processes and thus to approach cooperative games system theoretically. Our model includes all the above mentioned models as special cases. Also the classical results can be shown to extend to this wider context.

2. Cooperation Systems and Cooperative Games

A *cooperation system* is a quadruple $\Gamma = (N, V, A, \mathcal{A})$, where N is a finite set of agents, V a finite set of *states of cooperation* and A a finite set of feasible transitions $x \rightarrow y$ between states, which we assume to be partitioned into pairwise disjoint blocks A_i , indexed by the agents $i \in N$. We denote the latter partition by $\mathcal{A} = \{A_i | i \in N\}$ and think of the block $A_i \in \mathcal{A}$ as the set of transitions that are governed by the agent i : Intuitively, i can take the “action” $(x \rightarrow y) \in A_i$ and transform the current state x of cooperation into the state $y \in V$.

By identifying $(x \rightarrow y)$ with the pair $xy \in V \times V$, we obtain $G = (V, A)$ as the (directed) transition graph of Γ with vertex set V and arc set A . For simplicity of exposition, we assume throughout:

- (Γ_0) There is one unique *initial state* $S \in V$ (i.e. $s^- := \{u \in V | us \in A\} = \emptyset$).
- (Γ_1) G is acyclic.

This means every $x \in V$ can be reached by a directed path that starts in s . Moreover each such path extends to a path that ends in a sink t (i.e. $t^+ := \{u \in V | tu \in A\} = \emptyset$). A source-sink path is called a *cooperation instance* and we denote by \mathcal{P} the set of all cooperation instances.

A *cooperative game* is a pair (Γ, v) , where $v : V \rightarrow \mathbb{R}$ is a valuation of the states of cooperation (the so-called *characteristic function* of the game) with the property that $v(s) = 0$. The vector space of all cooperative Games on Γ is denoted by $\mathcal{V}(= \mathcal{V}(\Gamma))$.

A *selector* is an operator $X \mapsto \sigma(X)$ on the subsets of N such that $\sigma(X) \subseteq N \setminus X$ for all $X \subseteq N$. We show how cooperative games with selectors fit in our model. All mentioned generalizations of the classical model are such games with selectors. All cooperative games with selector suffice the following *singular action property* in words of cooperation instances:

- (SA) $|P \cap A_i| \leq 1$ for all $P \in \mathcal{P}$ and $i \in N$.

3. Allocation and Symmetries

We define an *allocation mechanism* to be a computational scheme for allocating payoffs to the individual agents in the context of a cooperative game (Γ, v) . We make some axiomatic assumption:

- (A_0) The null game should yield zero payoffs.
- (A_1) The allocation to the agent i should only depend on his action set A_i and

should be linear in v .

Let $\partial : \mathcal{V} \rightarrow \mathbb{R}^A$, $\partial_{xy}(v) := v(y) - v(x)$ be the *marginal operator*. Since ∂ is a monomorphism, it is an isomorphism on $\partial(\mathcal{V})$. A linear allocation mechanism is therefore described by a vector $\alpha \in \mathbb{R}^A$ that determines the individual values for $i \in N$:

$$\phi_i^\alpha(v) := \sum_{xy \in A_i} \alpha_{xy} \partial_{xy}(v).$$

We call the linear functional $v \mapsto \phi^\alpha(v) := \alpha^T \partial(v)$ the group value associated with the allocation mechanism α .

Together with the two assumptions (A_0) and (A_1) we develop a theory of linear allocation mechanisms strongly inspired by the theory of Weber [13] for the classical case. We define a Shapley allocation mechanism in our model and show that it is a generalization of the Shapley values proposed in the above mentioned models and expose it to be the unique mechanism of maximal entropy in some sense. Moreover we define λ -mechanisms as a generalization of weighted Shapley values studied by Shapley [11; 12] and Kalai and Samet [10].

Laxly speaking a symmetry of Γ is a permutation ρ of V which leaves the structure of Γ invariant, i.e.:

- (S_0) ρ is a graph automorphism of G . (Note that this is equivalent for ρ to leave invariant \mathcal{P})
- (S_1) ρ respects \mathcal{A} , i.e.: $\rho(A_i) \in \mathcal{A}$ for all $i \in N$.

In some sense the group of symmetries of Γ acts on the set of λ -values. A symmetry ρ of Γ is called an automorphism of a game v , if

$$v(\rho(x)) = v(x) \text{ for all } x \in V.$$

We extend a classical result of Owen and Carreras [7] to our model and show under the assumption (SA) that the automorphisms of a game v are exactly the symmetries of Γ which stabilize all λ -values. Sadly this statement may become false, if (SA) is dropped.

4. Core and Weber Sets

Finally we discuss the concept of a core of a cooperative game and its relation to the Weber set. For this discussion we restrict ourselves to cooperation systems that arise from selection structures. We propose a core concept as well as marginal vectors for our model as a generalization of the classical case.

The convex hull of all marginal vectors is classically called the Weber set of a

game. Weber [13] showed that in the classical model the core is always a subset of the Weber set. We extend this idea to games with selection structures and show how our results lead to Webers result as a special case.

References

- [1] E. Algaba, J.M Bilbao, R. van den Brink and A. Jiménez-Losada: *Cooperative games on antimatroids*. *Discr. Mathematics* 282 (2004), 1-15.
- [2] J.M. Bilbao, *Cooperative Games on Combinatorial Structures*. Kluwer Academic Publishers, 2000.
- [3] J.M. Bilbao, E. Lebrón and N. Jiménez: *The core of games on convex geometries*. *Europ. J. Operational Research* 119 (1999), 365-372.
- [4] J.M. Bilbao, T.S.H. Driessen, N. Jiménez Losada E. Lebrón: *The Shapley value for games on matroids*. *Math. Meth. Oper. Res.* 53 (2001), 333-348.
- [5] J.M. Bilbao and P. Edelman: *The Shapley value on convex geometries*. *Discrete Appl. Math.* 103 (2000), 33-40.
- [6] J.M. Bilbao, N. Jiménez, E. Lebrón and J.J López: *The marginal operators for games on convex geometries*. *Intern. Game Theory Review* 8 (2006), 141-151.
- [7] F. Carreras and G. Owen: *Automorphisms and weighted values*. *Intern. J. Game Theory* 26 (1997), 1-10.
- [8] J. Derks and R.P. Gilles: *Hierarchical organization structures and constraints in coalition formation*. *Intern. J. Game Theory* 24 (1995), 147-163.
- [9] U. Faigle and W. Kern: *The Shapley value for cooperative games under precedence constraints*. *Intern. J. Game Theory* 21 (1992), 249-266.
- [10] E. Kalai and D. Samet: *On weighted Shapley values*. *Int. Journ. Game Theory* 16 (1987), 205-222.
- [11] L.S. Shapley: *Additive and non-additive set functions*. PhD Thesis, Department of Mathematics, Princeton University Press, 1953.
- [12] L.S. Shapley: *A value for n -person games*. In: *Contributions to the Theory of Games*, H.W. Kuhn and A.W. Tucker eds., *Ann. Math. Studies* 28, Princeton University Press, 1953, 307-317.
- [13] R.J. Weber: *Probabilistic values for games*. In: A.E. Roth (ed.), *The Shapley Value*, Cambridge University Press, Cambridge, 1988, 101-120.

Matrices

Lecture Hall B

Thu 4, 15:45–16:15

Distribution of permanent of matrices with restricted entries over finite fields[★]

Le Anh Vinh

Mathematics Department, Harvard University, Cambridge, MA, 02138, USA

Key words: permanent, matrices over finite fields

1. Motivation

Throughout this paper, let $q = p^r$ where p is an odd prime and r is a positive integer. Let \mathbf{F}_q be a finite field of q elements. The prime base field \mathbf{F}_p of \mathbf{F}_q may then be naturally identified with \mathbf{Z}_p . Let M be an $n \times n$ matrices, two basic parameters of M are its determinant

$$\text{Det}(M) := \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^n a_{i\sigma(i)},$$

and its permanent

$$\text{Per}(M) := \sum_{\sigma \in S_n} \prod_{i=1}^n a_{i\sigma(i)}.$$

The distribution of the determinant of matrices with entries in a finite field \mathbf{F}_q has been studied by various researchers. Suppose that the ground field \mathbf{F}_q is fixed and $M = M_n$ is a random $n \times n$ matrices with entries chosen independently from \mathbf{F}_q . If the entries are chosen uniformly from \mathbf{F}_q , then it is well known that

$$\Pr(M_n \text{ is nonsingular}) \rightarrow \prod_{i \geq 1} (1 - q^{-i}) \text{ as } n \rightarrow \infty. \quad (1.1)$$

It is interesting that (1.1) is quite robust. Specifically, J. Kahn and J. Komlós [4] proved a strong necessary and sufficient condition for (1.1).

Theorem 19. ([4]) Let M_n be a random $n \times n$ matrix with entries chosen according to some fixed non-degenerate probability distribution μ on \mathbf{F}_q . Then (1.1) holds if and only if the support of μ is not contained in any proper affine subfield of \mathbf{F}_q .

[★] This paper was not actually presented at the conference, as the author did not attend (nor communicate his cancellation). The talk was replaced by the communication: S. Margulies, *Computing Infeasibility Certificates for Combinatorial Problems via Hilbert's Nullstellensatz*.

An extension of the uniform limit is to random matrices with μ depending on n was considered by Kovalenko, Leviskaya and Savchuk [5]. They proved that the standard limit (1.1) under the condition that the entries m_{ij} of M are independent and $\Pr(m_{ij} = \alpha) > (\log n + \omega(1))/n$ for all $\alpha \in \mathbf{F}_q$. The behavior of the nullity of M_n for $1 - \mu(0)$ close to $\log n/n$ and $\mu(\alpha) = (1 - \mu(0))/(q - 1)$ for $\alpha \neq 0$ was also studied by Blömer, Karp and Welzl [2].

Another direction is to fix the dimension of matrices. For an integer number n and a subset $\mathcal{E} \subseteq \mathbf{F}_q^n$, let $M_n(\mathcal{E})$ denote the set of $n \times n$ matrices with rows in \mathcal{E} . For any $t \in \mathbf{F}_q$, let $N_n(\mathcal{E}; t)$ be the number of $n \times n$ matrices in $M_n(\mathcal{E})$ having determinant t . Ahmadi and Shparlinski [1] studied some natural classes of matrices over finite fields \mathbf{F}_p of p elements with components in a given subinterval $[-H, H] \subseteq [-(p-1)/2, (p-1)/2]$. They showed that

$$D_n([-H, H]^n; t) = (1 + o(1)) \frac{(2H+1)^{n^2}}{p} \quad (1.2)$$

if $t \in \mathbf{F}_q^*$ and $H \gg q^{3/4}$. In the case $n = 2$, the lower bound can be improved to $H \gg q^{1/2+\varepsilon}$ for any constant $\varepsilon > 0$.

Covert et al. [3] studied this problem in a more general setting. A subset $\mathcal{E} \subseteq \mathbf{F}_q^n$ is called a product-like set if $|\mathcal{H}_d \cap \mathcal{E}| \lesssim |\mathcal{E}|^{d/n}$ for any d -dimensional subspace $\mathcal{H}_d \subset \mathbf{F}_q^n$. Covert et al. [3] showed that

$$D_3(\mathcal{E}; t) = (1 + o(1)) \frac{|\mathcal{E}|^3}{q},$$

if $t \in \mathbf{F}_q^*$ and $\mathcal{E} \subset \mathbf{F}_q^3$ is a product-like set of cardinality $|\mathcal{E}| \gg q^{15/8}$. Using the geometry incidence machinery developed in [3], and some properties of non-singular matrices, the author [7] obtained the following result for higher dimensional cases ($d \geq 4$):

$$D_n(\mathcal{A}^n; t) = (1 + o(1)) \frac{|\mathcal{A}|^{n^2}}{q},$$

if $t \in \mathbf{F}_q^*$ and $\mathcal{A} \subseteq \mathbf{F}_q$ of cardinality $|\mathcal{A}| \gg q^{\frac{d}{2d-1}}$.

On the other hand, little has been known about the permanent. The only known uniform limit similar to (1.1) for the permanent is due to Lyapkov and Sevast'yanov [6]. They proved that the permanent of a random $n \times m$ matrix M_{nm} with elements from \mathbf{F}_p and independent rows has the limit distribution of the form

$$\lim_{n \rightarrow \infty} \Pr(\text{Per}(M_{nm}) = k) = \rho_m \delta_{k0} + (1 - \rho_m)/p, \quad k \in \mathbf{F}_p,$$

where δ_{k0} is Kronecker's symbol. The purpose of this paper is to study the distribution of the permanent when the dimension of matrices is fixed. For any

$t \in \mathbf{F}_q$ and $\mathcal{E} \subset \mathbf{F}_q^d$, let $P_n(\mathcal{E}; t)$ be the number of $n \times n$ matrices with rows in \mathcal{E} having determinant t . We are also interested in the set of all permanents, $P_n(\mathcal{E}) = \{\text{Per}(M) : M \in M_n(\mathcal{E})\}$.

2. Statement of results

The main result of this paper is that, if \mathcal{E} is a sufficient large subset of \mathbf{F}_q^n then $P_n(\mathcal{E})$ covers \mathbf{F}_q . More precisely, our main result is the following theorem.

Theorem 20. Suppose that q is an odd prime power and $\gcd(q, n) = 1$.

a) If $\mathcal{E} \cap (\mathbf{F}_q^*)^n \neq \emptyset$, and $|\mathcal{E}| > cq^{\frac{n+1}{2}}$, then $\mathbf{F}_q^* \subseteq P_n(\mathcal{E})$.

b) If $\mathcal{E} \subset \mathbf{F}_q^n$ of cardinality $|\mathcal{E}| > nq^{n-1}$, then $\mathbf{F}_q^* \subseteq P_n(\mathcal{E})$.

\mathcal{A} is a sufficient large subset of \mathbf{F}_q then $P_n(\mathcal{A}^n)$ covers \mathbf{F}_q . More precisely, our main result is the following theorem.

Theorem 21. Suppose that q is an odd prime power and $\gcd(q, n) = 1$.

a) If $\mathcal{E} \cap (\mathbf{F}_q^*)^n \neq \emptyset$, and $|\mathcal{E}| > cq^{\frac{n+1}{2}}$, then $\mathbf{F}_q^* \subseteq P_n(\mathcal{E})$.

b) If $\mathcal{E} \subset \mathbf{F}_q^n$ of cardinality $|\mathcal{E}| > nq^{n-1}$, then $\mathbf{F}_q^* \subseteq P_n(\mathcal{E})$.

c) If $\mathcal{A} \subset \mathbf{F}_q$ of cardinality $|\mathcal{A}| \gg q^{\frac{1}{2} + \frac{1}{2n}}$, then $\mathbf{F}_q^* \subseteq P_n(\mathcal{A}^n)$.

b) If $\mathcal{A} \subseteq \mathbf{F}_q$ of cardinality $|\mathcal{A}| \gg q^{2/3}$, then for each $t \in \mathbf{F}_q^*$

$$P_3(\mathcal{A}^3; t) = (1 + o(1)) \frac{|\mathcal{E}|^3}{q}.$$

Note that the bound in Part b) of Theorem 20 is tight up to a factor of n . For example, $|\{x_1 = 0\}| = q^{n-1}$ and $P_n(\{x_1 = 0\}) = 0$. When \mathcal{E} is a product-like set, we can get a positive proportion of the permanents under a weaker assumption.

Theorem 22. Suppose that q is an odd prime power and $\gcd(q, n) = 1$. If $\mathcal{E} \subset \mathbf{F}_q^n$ is a product-like set of cardinality $|\mathcal{E}| \gg q^{\frac{n^2}{2(n-1)}}$, then

$$|P_n(\mathcal{E})| \geq (1 - o(1))q.$$

In the special case $\mathcal{E} = \mathcal{A} \times \dots \times \mathcal{A}$, we have the following corollary.

Corollary 1. Suppose that q is an odd prime power and $\gcd(q, n) = 1$. If $\mathcal{A} \subseteq \mathbf{F}_q$ of cardinality $|\mathcal{A}| \gg q^{\frac{1}{2} + \frac{1}{2(n-1)}}$, then $|P_n(\mathcal{A}^n)| \geq (1 - o(1))q$.

Furthermore, if we restrict our study to matrices over a finite field \mathbf{F}_p of p elements (p is a prime) with components in a given interval, we obtain a stronger result.

Theorem 23. Suppose that $q = p$ is a prime, and entries of M are chosen from a given interval $I := [a + 1, a + b] \subseteq \mathbf{F}_p$, where

$$\frac{b}{p^{1/2} \log p} \rightarrow \infty, \quad p \rightarrow \infty,$$

then

$$P_n(\mathcal{I}^n; t) = (1 + o(1)) \frac{b^{n^2}}{p}$$

for any $t \in \mathbf{F}_p$.

Throughout the abstract, the implied constants in the symbols ‘ o ’ and ‘ \gg ’ may depend on integer parameter n . We recall that the notation $U \gg V$ is equivalent to the assertion that the inequality $U \gg c|V|$ holds for some constant $c > 0$.

References

- [1] O. Ahmadi and I. E. Shparlinski, Distribution of matrices with restricted entries over finite fields, *Inda. Mathem.* **18**(3) (2007), 327–337.
- [2] J. Blömer, R. Karp, and E. Welzl, The rank of sparse random matrices over finite fields, *Random Structures and Algorithms* **10** (1997) 407–419.
- [3] D. Covert, D. Hart, A. Iosevich, D. Koh, and M. Rudnev, Generalized incidence theorems, homogeneous forms and sum-product estimates in finite fields, *European Journal of Combinatorics*, to appear.
- [4] J. Kahn and J. Komlós, Singularity probabilities for random matrices over finite fields, *Combinatorics, Probability and Computing* **10** (2001), 137–157.
- [5] I. N. Kovalenko, A. A. Leviskaya, and M. N. Savchuk, *Selected Problems in Probabilistic Combinatorics* (1986), Naukova Dumka, Kiev. In Russian.
- [6] L. A. Lyapkov and B. A. Sevast’yanov, The limiting probability distribution of a permanent of a random matrix in a $GF(p)$ field, *Diskr. Matem.*, **8**(2) (1996), 3–13.
- [7] L. A. Vinh, On the distribution of determinants of matrices with restricted entries over finite fields, *Journal of Combinatorics and Number Theory*, to appear.

Plenary Session III

Lecture Hall A

Thu 4, 16:45-17:30

On the boundary of tractability for nonlinear discrete optimization

Jon Lee

IBM TJ Watson Research Center, PO Box 218, Yorktown Heights, NY 10598
jonlee@us.ibm.com

Key words: mixed integer nonlinear programming, nonlinear discrete optimization

This paper covers the material from my talk at CTW 2009, Paris. * I am indebted to the Scientific Committee and to Leo Liberti and the other members of the Organizing Committee for the opportunity to present this paper.

Nonlinear discrete optimization, in its broadest sense, is simply the study of optimization models involving nonlinear functions in discrete variables. This is so hopelessly broad as to be a subject ripe for charlatans and cranks. Sober individuals cannot hope to devise efficient methods — practical or theoretical — for the entire class of such problems. So we set out some reasonable goals in the hope of delineating some of the boundary between tractable and intractable.

In §1, we look at polynomial optimization in integer variables from a complexity point of view. We summarize some key hardness results and also describe positive algorithmic results. More details regarding the material on polynomial optimization is collected in [14].

In §2, we take a different slice across nonlinear discrete optimization. In the context of a structured parametric nonlinear discrete optimization model, we describe some hardness results and also several broad cases for which we can give efficient exact or approximation algorithms. Much of that material is from [4; 19; 5]. I am enormously indebted to Shmuel Onn and Robert Weismantel for allowing me to survey some of our joint work which is the essence of §2. A full treatment of that material, which is only summarized here, will appear in our forthcoming monograph [20]. Further thanks are due to Yael Berstein who was also a key player in the development of some of that material.

* Cologne Twente Workshop 2009, 8th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, Ecole Polytechnique and CNAM, Paris, France June 2-5, 2009.

Finally, in §3, we describe a recent effort to implement one of the more novel algorithms from §2, using ultra-high precision arithmetic on a high-performance computational platform. I owe considerable gratitude to John Gunnels and Susan Margulies who were partners of mine in the work [10] summarized in §3.

1. Polynomial optimization

Polynomial optimization in continuous or integer variables refers to the model

$$\min / \max \{f_0(x) : f_i(x) \leq 0, i = 1, \dots, m; x \in D^n\},$$

where the $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ are polynomials, and D is either \mathbb{R} or \mathbb{Z} . Often one looks at the special case in which the constraint functions f_1, \dots, f_m are affine functions, and so the feasible region is either a polyhedron or the integer points in a polyhedron. Polynomial optimization in integer variables constitutes a very broad and natural class of nonlinear discrete optimization problems. As we shall soon see, some very simple subclasses are intractable, while for another broad subclass we get tractability, and for another broad subclass we get strong approximability.

First, we point out how hardness of nonlinear discrete optimization also implies hardness for nonlinear continuous optimization. Specifically, the max-cut problem can be modeled as minimizing a quadratic form over the cube $[-1, 1]^n$, and Håstad [12] demonstrated inapproximability for max-cut. Thus we have the following result:

Theorem 1.1. Polynomial optimization in continuous variables over polytopes in varying dimension is NP-hard. Moreover, there does not exist a fully polynomial-time approximation scheme, unless $P = NP$.

However, polynomial optimization in continuous variables over polytopes can be solved in polynomial time when the dimension is fixed. This follows from Renegar's general result on the complexity of approximating solutions to general algebraic formulae over the reals (see [23]).

For integer variables, hardness sets in for very low dimension. Based on reduction from the NP-complete problem of determining if there exists a positive integer $x < c$ with $x^2 \equiv a \pmod{b}$, we have the following (see [9; 6]):

Theorem 1.2. The problem of minimizing a degree-4 polynomial over the integer points of a convex polygon is NP-hard.

Moreover, hardness sets in with a vengeance. The negative solution of Hilbert's tenth problem by Matiyasevich [21; 22], building on earlier work by Davis, Putnam and Robinson [7], implies that nonlinear integer programming over unbounded fea-

sible regions is incomputable. Due to Jones' strengthening [16] of Matiyasevich's negative result, there also cannot exist any such algorithm for the cases of feasible regions for even a small fixed number of integer variables (see [6]):

Theorem 1.3. The problem of minimizing a linear form over polynomial constraints in at most 10 integer variables is not computable by a recursive function.

Another consequence, as shown by Jeroslow [15], is that even integer quadratic programming is incomputable.

Theorem 1.4. The problem of minimizing a linear form over quadratic constraints in integer variables is not computable by a recursive function.

So far, we have painted a rather bleak picture for polynomial optimization. But the inherent difficulty is related to non-convexity, and it becomes worse in varying dimension. On the positive side, Khachiyan and Porkolab have demonstrated that in fixed dimension, the problem of minimizing a convex polynomial objective function over the integers, subject to polynomial constraints describing a convex body, can be solved in time polynomial in the encoding length of the input [17]. This result was strengthened by Heinz [13] to achieve the following result based on generalizing Lenstra's algorithm for linear integer optimization in fixed dimension [18].

Theorem 1.5. Let the dimension n be fixed. The problem of minimizing $f_0(x)$ on the set of integer points satisfying $f_i(x) \leq 0$, $i = 1, 2, \dots, m$, where the $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ are quasi-convex polynomials with integer coefficients, for $i = 0, 1, \dots, m$, can be solved in time polynomial in the degrees and the binary encoding of the coefficients.

Owing to the difficulty, already, of optimizing (non-convex) degree-4 polynomials over the integer points in a convex polygon (Theorem 1.2), the best that we can hope for, in fixed dimension without a convexity assumption on the objective, is an approximation result. In fact, a very strong result — namely a fully polynomial-time approximation scheme — has been established (see [6]):

Theorem 1.6. Let the dimension n be fixed. Let $P \subset \mathbb{R}^n$ be a rational convex polytope. Let f be a polynomial with rational coefficients that is non-negative on $P \cap \mathbb{Z}^n$, given as a list of monomials with rational coefficients c_β encoded in binary and exponent vectors $\beta \in \mathbb{Z}_+^n$ encoded in unary. Then we can find a feasible solution $x \in P \cap \mathbb{Z}^n$ with $f_{\max} - f(x) \leq \epsilon f_{\max}$, in time polynomial in the input and $1/\epsilon$.

2. Parametric Nonlinear Discrete Optimization

In this section, we take another view across the landscape of nonlinear discrete optimization. While in the last section our viewpoint was to look at specializations of mathematical programming models, in the present section our viewpoint more closely aligned with that taken in combinatorial optimization. From this different viewpoint, we will see other aspects of the boundary between tractable and intractable nonlinear discrete optimization models.

We consider the *parametric nonlinear discrete optimization* model

$$\min / \max \{f(Wx) : x \in \mathcal{F}\},$$

where $W \in \mathbb{Z}^{d \times n}$, $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is specified by a comparison oracle, and $\mathcal{F} \subset \mathbb{Z}^n$ is *well described* (i.e., we have access to an oracle for optimizing an arbitrary linear objective on \mathcal{F}). One motivation for the study of such a model is multi-objective optimization, where we view each row of W as specifying a linear objective, and then the nonlinear f balances the d competing linear objectives. Besides this appealing motivation, the structure of this model also provides a nice structure for exploring the boundary between intractable and tractable. So, in the remainder of this section, we will expose some of this boundary, as we vary hypotheses on f , W and \mathcal{F} .

We make some brief comments about our general hypothesis on \mathcal{F} , that it is well described. Usually such a term would be formally defined as meaning that we have a separation oracle for $\text{conv}(\mathcal{F})$. But of course the polynomial equivalence of separation and optimization is well known (see [11]). Finally, the hypothesis that we can optimize arbitrary linear functions on the discrete set \mathcal{F} is very natural, from both the theoretical and practical viewpoints, as we try to lift up to *nonlinear* discrete optimization.

One of our primary complexity levers is the encoding of W . We will see that typically for binary-encoded W , we will have intractability, and so to obtain positive results we will need to hypothesize that the number of rows d is fixed, and that the entries of W are somehow small. The exact hypotheses vary over the results that we present, so we lay out here the possibilities: (i) unary encoding of the w_{ij} , (ii) $w_{ij} \in \{a_1, \dots, a_p\}$, where p is fixed and the $a_k \in \mathbb{Z}$ are binary encoded input, (iii) $w_{ij} \in \{a_1, \dots, a_p\}$, where $a_1, \dots, a_p \in \mathbb{Z}_+$ are fixed, (iv) $w_{ij} = \sum_k \lambda_k^{ij} a_k$, where p is fixed, the $a_k \in \mathbb{Z}$ are binary-encoded input, and the λ_k^{ij} are unary-encoded input. In this last case, we say that W has a *unary encoding over* $\{a_1, \dots, a_p\}$.

The following three results demonstrate the strong intractability of parametric nonlinear discrete optimization. The results emphasize matroids, because for *linear* objectives such problems are *very easy* — the greedy algorithm works.

Theorem 2.1. Computing the optimal objective value of

$$\min / \max \{f(wx) : x \in \mathcal{F}\},$$

when $d = 1$, $w \in \mathbb{Z}_+^n$, f is a univariate function presented by a comparison oracle, and \mathcal{F} is the set of bases of a uniform or graphic matroid on an n -element ground sets, cannot be done in time polynomial in n and the binary encoding of w .

Theorem 2.2. Computing the optimal objective value of

$$\min / \max \{f(Wx) : x \in \mathcal{F}\},$$

when $d = n$, $W = I_n$, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ presented by a comparison oracle, and \mathcal{F} is the set of bases of a uniform or graphic matroid on an n -element ground sets, cannot be done in time polynomial in n .

Theorem 2.3. Determining whether the optimal objective value is zero for

$$\min \{f(wx) : x \in \mathcal{F}\},$$

when $d = 1$, binary-encoded $w \in \mathbb{Z}_+^n$, f is the explicit convex univariate function $f(y) := (y - u_1)^2$, and \mathcal{F} is the set of bases of a uniform or graphic matroid on an n -element ground sets, is NP-complete.

Despite the strong intractability of the general model, we are able to get positive complexity results for broad classes of interest. We are able to do this, for the most part, by fixing the number of rows of W and restricting the encoding of its entries. Depending on the precise restrictions on W , we are able to address different types of functions f .

Theorem 2.4. If \mathcal{F} is well described, f is quasi-convex, and W has a fixed number of rows and has a unary encoding over binary encoded $\{a_1, \dots, a_p\}$, then there is an efficient deterministic algorithm for $\max \{f(Wx) : x \in \mathcal{F}\}$.

Theorem 2.5. If \mathcal{F} is well described, f is a norm, and W has a fixed number of rows and is binary-encoded and non-negative, then there is an efficient deterministic constant-approximation algorithm for $\max \{f(Wx) : x \in \mathcal{F}\}$.

A function $f : \mathbb{R}_+^d \rightarrow \mathbb{R}$ is *ray concave* if

$$\lambda f(u) \leq f(\lambda u) \text{ for } u \in \mathbb{R}_+^d, 0 \leq \lambda \leq 1.$$

For example, if f is a norm on \mathbb{R}^d , then it ray-concave and non-decreasing on \mathbb{R}_+^d . As a further example, $f(u) := \|u\|_1 - \|u\|_s$, for any integer $s \geq 1$ or infinity, is ray-concave and non-decreasing on \mathbb{R}_+^d . Notice that already for $d = 2$ and $s = \infty$, $f(u)$ is not a norm — indeed, for this case $f(u) = \min(u_1, u_2)$.

Theorem 2.6. If \mathcal{F} is well described, f is ray concave and non-decreasing, and W has a fixed number of rows and has a unary encoding over binary encoded $\{a_1, \dots, a_p\}$, then there is an efficient deterministic constant-approximation algorithm for $\min \{f(Wx) : x \in \mathcal{F}\}$.

Turning to general functions f , we must be much more modest in our expectations. The next results establishes very strong intractability. An *independence system* $\mathcal{F} \subset \{0, 1\}^n$ has the property that for $x \in \mathcal{F}$ and $y \in \{0, 1\}^n$ with $y \leq x$, we have $y \in \mathcal{F}$.

Theorem 2.7. There is no efficient algorithm for computing an optimal solution of the *one-dimensional* nonlinear optimization problem $\min / \max \{f(wx) : x \in \mathcal{F}\}$ over a well-described independence system, with f presented by a comparison oracle, and single weight vector $w \in \{2, 3\}^n$.

Still, we can establish a positive result, using a new notion of approximation that is appropriate for general functions f . We say that $x^* \in \mathcal{F}$ is *r-best* for

$$\min / \max \{f(Wx) : x \in \mathcal{F}\},$$

if at most r better values than $f(Wx^*)$ are achievable as $f(Wx)$, over points $x \in \mathcal{F}$. A p -tuple a is *primitive* if its entries are distinct positive integers having gcd 1.

Theorem 2.8. For every primitive p -tuple a , there is a constant $r(a)$ and an efficient algorithm that, given any well-described independence system $\mathcal{F} \subseteq \{0, 1\}^n$, a single weight vector $w \in \{a_1, \dots, a_p\}^n$, and function $f : \mathbb{Z} \rightarrow \mathbb{R}$ presented by a comparison oracle, finds an $r(a)$ -best solution to $\min / \max \{f(wx) : x \in \mathcal{F}\}$.

Moreover, (i) if a_i divides a_{i+1} for $i = 1, \dots, p-1$, then the algorithm provides an optimal solution; (ii) for $p = 2$, that is, for $a = (a_1, a_2)$, the algorithm provides an $(a_1 a_2 - a_1 - a_2)$ -best solution.

Even though the situation for arbitrary well described independence systems is tough, for a matroid even presented by an independence oracle, we have an efficient algorithm for optimizing general functions f .

Theorem 2.9. If \mathcal{F} is the set of characteristic vectors of bases or independent sets of a single matroid presented by an independence oracle, $c^T \in \mathbb{Z}^n$ is binary encoded, f is arbitrary and given by a comparison oracle, and $d \times n$ matrix W has a fixed number of rows and has entries in binary encoded $\{a_1, \dots, a_p\}$ with p fixed, then there is an efficient deterministic algorithm for $\min / \max \{cx + f(Wx) : x \in \mathcal{F}\}$.

Turning to vectorial matroids (over the rationals so as to make our complexity statements simple), and modifying the assumptions on the encoding of W , we are able to again get an efficient algorithm.

Theorem 2.10. If \mathcal{F} is the set of characteristic vectors of bases or independent sets of a single rational vectorial matroid represented by a binary-encoded integer matrix A , f is arbitrary and given by a comparison oracle, and W has a fixed number of rows and is unary encoded, then there is an efficient deterministic algorithm for $\min / \max \{f(Wx) : x \in \mathcal{F}\}$.

Finally, for matroid intersection, again for vectorial matroids, we are able to get an efficient *randomized* algorithm for general f .

Theorem 2.11. If \mathcal{F} is the set of characteristic vectors of common bases or independent sets of a pair of rational vectorial matroids, represented by binary-encoded integer matrices A_1 and A_2 , on a common ground set, f is arbitrary and given by a comparison oracle, and W has a fixed number of rows and is unary encoded, then there is an efficient randomized algorithm for $\min / \max \{f(Wx) : x \in \mathcal{F}\}$.

3. Supercomputing

In §2, we have omitted the algorithms and analyses that form the proofs of the theorems. Many of the algorithms are not particularly esoteric, so the range of parameters for which they are practical is mostly apparent.

But this generalization has some exceptions. The algorithms that form the bases of the proofs of Theorems 2.10 and 2.11 might seem to be of only theoretical interest. In this section we describe the algorithm from the proof of Theorem 2.10, and a bit about how we have implemented it, in ultra-high precision arithmetic on a Blue Gene/L supercomputer [1].

Without loss of generality, we can assume that W is non-negative and that we are optimizing over the bases of M (the case of arbitrary W and independent sets is treated, easily, in [20]). Let $A \in \mathbb{Z}^{r \times n}$ be the matrix representation of the (rational) vectorial matroid M , and let \mathcal{F} be the set of characteristic vectors of bases of M .

It turns out that it is sufficient to be able to efficiently calculate an optimal Wx — there is a simple methodology for recovering an associated x . The motivating idea of the algorithm is to determine, in one go, the entire set of points

$$U := \{Wx : x \text{ is the characteristic vector of a base of } M\}.$$

We observe that U is a subset of $Z := \{0, 1, \dots, r\omega\}^d$. By our assumptions, $|Z|$ is bounded by a polynomial in the size of the data encoding. So, once we have U , we can easily determine an optimal Wx using the comparison oracle of f .

Define the following polynomial in d variables y_1, \dots, y_d :

$$g = g(y) := \sum_{u \in Z} g_u \prod_{k=1}^d y_k^{u_k},$$

where the coefficient g_u corresponding to $u \in Z$ is the non-negative integer

$$g_u := \sum \left\{ \det^2(A_x) : x \in \mathcal{F}, Wx = u \right\},$$

where A_x is the $r \times r$ submatrix of A indicated by the 0/1 vector x . Now, $\det^2(A_x)$ is positive for every $x \in \mathcal{F}$. Thus, the coefficient g_u corresponding to $u \in Z$ is non-zero if and only if there exists an $x \in \mathcal{F}$ with $Wx = u$. So the desired set U is precisely the set of exponent vectors u of monomials $\prod_{k=1}^d y_k^{u_k}$ having non-zero coefficient g_u in g .

Next, a simple lemma provides a key ingredient for our algorithm.

Lemma 3.1.

$$g(y) = \det(AYA^T).$$

Finally, the key idea is that we can determine the coefficients g_u of the monomials in g indirectly, by using the lemma to evaluate g at enough points. Thus we get Algorithm 1.

Algorithm 1: Efficient enumeration of the image of \mathcal{F} under W

input: full row-rank $A \in \mathbb{Z}^{r \times n}$ (binary encoded), $W \in \mathbb{Z}_+^{d \times n}$ (unary encoded);

let $\omega := \max w_{i,j}$, $s := r\omega + 1$ and $Z := \{0, 1, \dots, r\omega\}^d$;

let $Y := \text{diag}_j \left(\prod_{i=1}^d y_i^{w_{i,j}} \right)$;

for $t = 1, 2, \dots, s^d$ **do**

let $Y(t)$ be the numerical matrix obtained by substituting $t^{s^{i-1}}$ for y_i

$(i = 1, 2, \dots, d)$ in Y ;

compute $\det(AY(t)A^T)$;

end

compute the unique solution $g_u, u \in Z$, of the square linear system:

$$\sum_{u \in Z} t^{\sum_{i=1}^d u_i s^{i-1}} g_u = \det(AY(t)A^T), \quad t = 1, 2, \dots, s^d;$$

return $U := \{u \in Z : g_u > 0\}$.

We would like to view the system of equations from the algorithm a bit more concretely in the form

$$V^T g = b, \tag{3.1}$$

where V is an order s^d square matrix, g is an s^d vector of real variables, and the right-hand side b is an s^d -vector of constants. Clearly we will let $b_t := \det(AY(t)A^T)$, for $t = 1, 2, \dots, s^d$. As for the variables, we need a numbering

of the elements of Z . A natural numbering is via the $\phi : Z \rightarrow \{1, 2, \dots, s^d\}$ defined by $\phi(u) := 1 + \sum_{i=1}^d u_i s^{i-1}$. In fact this map is just a lexical ordering of the elements of Z ; for example, $\phi((0, 0, \dots, 0)^T) = 1$ and $\phi((r\omega, r\omega, \dots, r\omega)^T) = s^d$.

With this notation, we can now view the linear system as

$$\sum_{j=1}^{s^d} t^{j-1} g_j = b_t, \quad t = 1, 2, \dots, s^d. \quad (3.2)$$

Letting $k := s^d$, we let the $k \times k$ matrix V^T be defined by

$$V_{t,j}^T := t^{j-1}, \quad \text{for } 1 \leq t, j \leq k.$$

With this definition of V^T , (3.2) has the form (3.1).

In this form, we see that V is a (special) Vandermonde matrix (so it is invertible), and the system (3.1) that we wish to solve is a so-called ‘‘dual problem.’’ We propose to solve it simply by evaluating V^{-1} , and letting $g := V^{-T}b$.

Our Vandermonde matrix is a very special one. It even has a closed form for the inverse V^{-1} :

$$V_{i,j}^{-1} := \begin{cases} (-1)^{i+k} \frac{1}{(i-1)!(k-i)!}, & j = k; \\ i V_{i,j+1}^{-1} + \begin{bmatrix} k+1 \\ j+1 \end{bmatrix} V_{i,k}^{-1}, & 1 \leq j < k, \end{cases}$$

where $\begin{bmatrix} k+1 \\ j+1 \end{bmatrix}$ denotes a Stirling number of the first kind (see [8], though they define things slightly differently there). The form for $V_{i,j}^{-1}$ indicates how each row of V^{-1} can be calculated independently, with individual entries calculated from right to left, albeit with the use of Stirling numbers of the first kind. We note that the Stirling number used for $V_{i,j}^{-1}$ does not depend on the row i , so the needed number can be computed once for each column j . The (signed) Stirling numbers of the first kind can be calculated in a ‘‘triangular manner’’ as follows (see [24]). For $-1 \leq j \leq k$, we have

$$\begin{bmatrix} k+1 \\ j+1 \end{bmatrix} := \begin{cases} 0, & k \geq 0, j = -1; \\ 1, & k \geq -1, j = k; \\ \begin{bmatrix} k \\ j \end{bmatrix} - k \begin{bmatrix} k \\ j+1 \end{bmatrix}, & k > j \geq -1. \end{cases}$$

A remark is in order concerning the practicality of working with large Vandermonde systems and Stirling numbers. The numerics would quickly get out of hand, using ordinary limited-precision arithmetic, when $k = s^d$ is even modest in magnitude. So, the practical implementation of [10] uses the ultra-high precision arithmetic library ARPREC (see [2; 3]).

Finally, it is easy to see that there is enormous potential for parallelism in the calculation of the needed Stirling numbers and in the formation and use of the Vandermonde inverse (see [10] for details).

4. Remarks

It is not the case that algorithms and implementations like those described in §3 are currently very practical. After all, not everyone has a supercomputer, and even for the lucky few, there remains a large gap between instances that we would like to solve and those that we can currently handle. However, I hope that we have demonstrated that as computational platforms evolve, our view of what is possible and eventually practical for discrete optimization should evolve accordingly. We have only worked out the details and implemented one algorithm — the one for Theorem 2.10. An algorithm for Theorem 2.11, though more complicated, is based on similar ideas, and there is clearly the potential to make an effective implementation for it. Certainly, a broad paradigm for solving discrete optimization based on matrix algebra in ultra-high precision on supercomputers would be very attractive. We hope that this work is a first step in such a direction.

References

- [1] G. Almási, C. Archer, J. Castaños, J. Gunnels, C. Erway, P. Heidelberger, J. Moreira, K. Pinnow, J. Ratterman, B. Steinmacher-Burow, W. Gropp & B. Toonen (2005). Design and implementation of message-passing services for the Blue Gene/L supercomputer, *IBM Journal of Research and Development*, 49(2–3):393–406.
- [2] D. H. Bailey, ARPREC, crd.lbl.gov/~dhbailey/mpdist/mpdist.html.
- [3] D. H. Bailey (2005). High-precision arithmetic in scientific computation, *Computing in Science and Engineering*, 7(3):54–61.
- [4] Y. Bernstein, J. Lee, H. Maruri-Aguilar, S. Onn, E. Riccomagno, R. Weismantel & H. Wynn (2008). Nonlinear matroid optimization and experimental design, *SIAM Journal on Discrete Mathematics*, 22(3):901–919.
- [5] Y. Bernstein, J. Lee, S. Onn & R. Weismantel (2008). Nonlinear optimization for matroid intersection and extensions, Report RC24610, IBM Research.
- [6] J. A. De Loera, R. Hemmecke, M. Köppe & R. Weismantel (2006). Integer polynomial optimization in fixed dimension, *Mathematics of Operations Research*, 31(1):147–153.
- [7] M. Davis, H. Putnam & J. Robinson (1961). The decision problem for exponential diophantine equations, *Ann. of Math. (2)*, 74:425–436.

- [8] A. Eisenberg, G. Franzé & P. Pugliese (1998). Vandermonde matrices on integer nodes, *Numerische Mathematik*, 80(1):75–85.
- [9] M. R. Garey & D. S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman and Company, New York, NY.
- [10] J. Gunnels, J. Lee & S. Margulies (2008). Efficient high-precision dense matrix algebra on parallel architectures for nonlinear discrete optimization, Report RC24682, IBM Research.
- [11] M. Grötschel, L. Lovász & A. Schrijver (1981). The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica*, 1:169–197.
- [12] J. Håstad (1997). Some optimal inapproximability results, in: *Proceedings of the 29th Symposium on the Theory of Computing (STOC)*, (pp. 1–10), ACM.
- [13] S. Heinz (2005). Complexity of integer quasiconvex polynomial optimization, *J. Complexity*, 21(4):543–556.
- [14] R. Hemmecke, M. Köppe, J. Lee & R. Weismantel (2008). Nonlinear integer programming, *Mathematical Programming* (to appear).
- [15] R. G. Jeroslow (1973). There cannot be any algorithm for integer programming with quadratic constraints, *Operations Research*, 21(1):221–224.
- [16] J. P. Jones (1982). Universal diophantine equation, *Journal of Symbolic Logic*, 47(3):403–410.
- [17] L. Khachiyan & L. Porkolab (2000). Integer optimization on convex semialgebraic sets., *Discrete and Computational Geometry*, 23(2):207–224.
- [18] H. W. Lenstra, Jr. (1983). Integer programming with a fixed number of variables, *Math. Oper. Res.*, 8(4):538–548.
- [19] J. Lee, S. Onn & R. Weismantel (2008). Nonlinear optimization over a weighted independence system, Report RC24513, IBM Research.
- [20] J. Lee, S. Onn & R. Weismantel (2009). *Nonlinear Discrete Optimization*, draft monograph.
- [21] Yu. V. Matiyasevich (1970). Enumerable sets are diophantine, *Doklady Akademii Nauk SSSR*, 191:279–282, (Russian); English translation, *Soviet Mathematics Doklady*, vol. 11 (1970), pp. 354–357.
- [22] Yu. V. Matiyasevich (1993). *Hilbert's tenth problem*, The MIT Press, Cambridge, MA, USA.
- [23] J. Renegar (1992). On the computational complexity of approximating solutions for real algebraic formulae, *SIAM Journal on Computing*, 21(6):1008–1025.
- [24] I. Tweddle (2003). *James Stirling's Methodus Differentialis: An Annotated Translation of Stirling's Text*, Sources and Studies in the History of Mathematics and Physical Sciences, Springer.

List of Authors

- Abouelaoualim A., 115
Alves C., 215
Amaldi E., 355
Andonov R., 341
Arbib C., 75
Ayala R., 277
- Bampas E., 68
Belgacem L., 225
Ben-Ameur W., 105
Bentz C., 313
Bermudo S., 297
Bettinelli A., 269
Bhandari R., 97
Bianco L., 210
Bodirsky M., 261
Bonomo F., 195, 281
Borozan V., 115
Brinkmann G., 333
Brinkmeier M., 301
Bruglieri M., 11
- Cacchiani V., 171
Cameron K., 120
Caprara A., 171
Caramia M., 210
Casazza M., 7
Cavellucci C., 317
Ceselli A., 7, 163
Charon I., 225
Collet G., 341
Colorni A., 11
Coniglio S., 355
Cordone R., 163, 180
Cornuéjols G., 144
Correa J.R., 347
Cremonini M., 163
- Darmann A., 293
de Abreu N.M.M., 157
de Carvalho J.M.V., 215
de Figueiredo C., 55
de Freitas M.A.A., 157
- de Souza C.C., 48, 187
Del-Vecchio R. R., 157
Dhyani K., 355
Dobson M. P., 153
Dong C., 3
Durán G., 195, 281
- Engels B., 239
Ernst C., 3
Ewe H., 109
- Faigle U., 361
Faria L., 229
Fawcett J., 120
Fernández-Ternero D., 277
Fernau H., 79, 205
Ficarelli F., 180
Filho C. L., 317
Fischetti M., 171
Frota Y., 48, 187
- Galluccio A., 251
Gaspers S., 205
Gentile C., 251
Giakoumakis V., 305
Gibrat J.-F., 341
González J. F. V., 317
Grande E., 321
Grippio L. N., 281
Gropp H., 28
Guedes A. L. P., 229
- Habib M., 257
Horváth I., 149
Hoshino E. A., 187
Hudry O., 225
- Jäger G., 3
- Küfer K., 109
Kang R. J., 60
Katona G. Y., 149
Kosuch S., 140
Kratsch D., 205

Krumke S., 239
 La Rota C., 325
 Lavor C., 337
 Lee J., 373
 Leoni V. A., 153
 Liberti L., 144, 175, 269
 Liedloff M., 205
 Liers F., 351
 Lisser A., 140
 Lodi A., 125
 Lozin V. V., 40
 Lozovanu D., 221
 Lué A., 11
 Lyons A., 199
 Müller D., 265
 Müller T., 60
 Maßberg J., 84
 Macedo R., 215
 Machado R., 55
 Macina M., 251
 Maculan N., 337
 Maischberger M., 93
 Manoussakis Y., 115
 Marengo J., 195
 Margulies S., 367
 Marinelli F., 75
 Markenzon L., 229
 Martinhon C., 115
 Maßberg J., 35
 Megow N., 347
 Mirchandani Pitu B., 321
 Molitor P., 3
 Morelato França A. L., 89
 Morelato Frana P., 89
 Mucherino A., 337
 Muthu R., 115
 Nannicini G., 144
 Nasini G. L., 153
 Neto J., 105
 Nicosia G., 44
 Nieberg T., 35
 Nikolopoulos S.D., 23
 Nordh G., 261
 Nunkesser M., 7
 Ould El Mounir C.B., 305
 Pacifici A., 44, 321
 Pagourtzis A., 68
 Papadopoulos C., 23
 Pardella G., 351
 Petrosyan P. A., 64
 Pferschy U., 44, 293
 Pickl S., 221
 Pierrakos G., 68
 Raible D., 79, 205
 Raimondi F., 175, 269
 Ralphs T.K., 125
 Raman R., 347
 Richter D., 3
 Righini G., 180
 Roda F., 175
 Rodríguez-Velázquez J. A., 297
 Rossi A., 191
 Saad R., 115
 Safe M. D., 281
 Sargsyan H. E., 64
 Sau I., 19
 Savourey D., 269
 Schüle I., 109
 Schauer J., 293
 Schneider J., 84
 Schrader R., 239
 Schultz R., 137
 Scoppola C. M., 75
 Scozzari A., 235
 Sevaux M., 191
 Shigezumi T., 285
 Sigarreta J. M., 297
 Simonetti L., 48
 Soto M., 191
 Stephan R., 247
 Suchan K., 347
 Syrgkanis V., 68
 Tardella F., 235
 Tarissan F., 325

Thilikos D. M., 19
Uno Y., 285
Usberti F. L., 89, 317
Valencia-Pabon M., 195
Van Cleemput N., 333
Ventura P., 251
Vilches J. A., 277
Vinh L.A., 367
von Oertzen T., 261
Voss J., 361
Watanabe O., 285
Woeginger G. J., 293
Yanev N., 341
Yero I. G., 297
Zeck C., 239

List of Sessions

Applications, 163, 171, 175, 180

Bioinformatics, 333, 337, 341

Clustering, 347, 351, 355

Coloring I, 55, 60, 64, 68

Coloring II, 187, 191, 195, 199

Combinatorial Optimization, 35, 40,
44, 48

Complexity, 235, 239

Cutting and Packing, 75, 79, 84

Exact Algorithms, 205, 210, 215

Games, 361

Graph Theory I, 19, 23, 28

Graph Theory II, 149, 153, 157

Graph Theory III, 277, 281, 285

Graph Theory IV, 293, 297, 301, 305

Integer Programming, 137, 140, 144

Matrices, 367

Networks I, 221, 225, 229

Networks II, 313, 317, 321, 325

Paths, 89, 93, 97

Plenary I, 373

Plenary II, 125

Plenary III, 257

Polyhedra, 247, 251

Polynomial-time Algorithms, 261,
265, 269

Quadratic Programming, 105, 109

Traveling Salesman Problem, 3, 7, 11

Trees, 115, 120

List of Timeslots

Thu 4 [08:45–10:15], 261, 265, 269,
277, 285

Thu 4 [10:30–12:30], 281, 293, 297,
301, 305, 313, 317, 321, 325

Thu 4 [14:00–15:30], 333, 337, 341,
347, 351, 355

Thu 4 [15:45–16:15], 361, 367

Thu 4 [16:45–17:30], 373

Tue 2 [08:45–10:15], 3, 7, 11, 19, 23,
28, 64

Tue 2 [10:30–12:30], 35, 40, 44, 48,
55, 60, 68

Tue 2 [14:00–15:30], 75, 79, 84, 89,
93, 97

Tue 2 [15:45–16:45], 105, 109, 115,
120

Tue 2 [16:45–17:30], 125

Wed 3 [08:45–10:15], 137, 140, 144,
149, 153, 157

Wed 3 [10:30–12:30], 163, 171, 175,
180, 187, 191, 195, 199

Wed 3 [14:00–15:30], 205, 210, 215,
221, 225, 229

Wed 3 [15:45–16:45], 235, 239, 247,
251

Wed 3 [16:45–17:30], 257

Keywords

Q -integral graph, 157
 δ -hyperbolic metric spaces, 257

acyclic, 60
acyclic coloring, 199
acyclic graph, 115
allocation mechanism, 361
anonymity, 269
anonymous routing, 269
anonymous subgraph problem, 269
approximation algorithms, 265
APX-hardness, 313
arc routing problem, 11, 89
arrangements, 105

Berge graph, 28
biclique, 205
bilevel programming, 125
bioinformatics, 337, 341
block graphs, 195
boolean probability bounding problem, 235
boundary of tractability, 373
branch and bound, 144, 163, 210
branch and prune, 337
branch-and-price, 187, 215
branching, 144
branchwidth, 19
Breadth-First Search, 257

c -edge-colored graph, 115
cardinality constraint, 347
cartesian product of graphs, 297
caterpillar trees, 48
center computations, 257
chemistry, 333
cherry trees, 235
chordal graphs, 235
chromatic number, 191, 297
circle graphs, 281
classification, 333
clique, 285, 347

clustering, 163
cographs, 199
coherent probability, 235
collaborative filtering, 175
colored graphs, 35
colored spanning trees, 115
coloring, 35
column generation, 321, 355
combinatorial algorithm, 351
combinatorial optimization, 48, 225, 293, 341
complexity, 105
computational complexity, 261
conflict graph, 225, 293
conformation, 337
constrained routing, 97
cooperation, 361
coordinate algorithm, 93
core, 361
critical simplex, 277
cutting plane, 215
cutting planes, 125
Cutting Stock Problem, 75
cycle with a unique chord, 55

decomposition methods, 137
degree, 285
descent method, 225
diameter, 257
directed hypergraph, 229
discrete control problems, 221
distance geometry, 337
dominance constraints, 137
double graph, 157
duality, 19
dynamic programming, 163, 247

edge coloring, 64
edge connectivity, 301
edge connectivity augmentation, 301
edge cutset, 97
edge-covering number, 153

edge-perfect graph, 153
 efficient algorithms, 7, 261
 emergency services, 180
 enumeration, 333
 enumeration algorithm, 305
 Euclidean traveling salesman problem, 3
 exact algorithm, 205, 321
 exact algorithms, 215
 exact method, 48
 excellent discrete Morse function, 277
 exponential-time algorithm, 205
 extremal properties, 149
 extremal stable graphs, 149
 extreme set, 301

feeding precedences, 210
 finite fields, 367
 flows over time, 321
 formulation by representatives, 187
 fractional packing, 265
 frugal, 60

games, 44, 68
 gene regulatory networks reconstruction, 325
 general disjunctions, 144
 generalized flow, 239
 goal directed search, 93
 goal oriented shortest path, 93
 gradient path, 277
 gradient vector field, 277
 graph, 19, 55, 60, 79, 97, 149, 153, 157, 199, 205, 277, 293, 297
 graph algorithms, 313
 graph coloring, 60, 191, 195
 graph partitioning, 347
 graph theory, 28
 graph's topology, 269
 graphs on surfaces, 19

Helly circle graphs, 281
 heuristic, 3, 225, 239
 hierarchical product, 157
 hop constraints, 247
 hypergraph, 229

hyperplane clustering, 355

improper, 60
 independent set, 35, 40, 225
 infinite locally finite graph, 277
 integer flow, 239
 integer linear programming, 180
 integer nonlinear programming, 317
 integer programming, 48, 137, 144, 180, 261, 341
 intermediate trees, 120
 interval coloring, 64
 inverse problems, 325
 isolated clique size, 285
 iterative approach, 3

Kasteleyn city, 351
 knapsack and subset sum problem, 44
 knapsack problem, 293

Lagrangian relaxation, 171, 210
 LIFO constraints, 7
 linear-time recognition algorithm, 23
 location, 180
 lower bounds, 75

makespan, 210
 Markov processes, 221
 mathematical programming, 325
 matrices, 367
 matrices over finite fields, 367
 max-cut, 351
 maximally violated valid inequalities, 125
 maximum adjacency order, 301
 maximum capacity path, 175
 maximum independent set problem, 40
 metaheuristic, 355
 min-max resource sharing, 265
 minimal prime extension, 305
 minimal weight increment, 97
 minimum caterpillar spanning problem, 48
 minimum cut, 301
 minimum spanning tree, 293
 minimum sum coloring, 195

mixed integer nonlinear programming, 373
 modular decomposition, 305
 modules in graphs, 305
 molecule, 337
 Morse functions, 277
 multi-agent optimization, 44
 multicommodity min-cost flow over time, 321
 multicuts, 313
 multiobjective optimization, 317
 multipliers, 239

 nanocone, 333
 network, 97, 257
 network optimization, 317
 nonapproximability bounds, 115
 nonlinear discrete optimization, 373
 NP-completeness, 55, 84, 115, 235
 NP-hard problem, 205
 NP-hardness, 89, 313
 Nullstellensatz, 367

 odd chordless cycle, 153
 offensive alliances, 297
 optical networks, 225
 OSPF, 97

 packings, 79
 parameterized algorithms, 79
 parameterized complexity, 40
 pareto frontier, 317
 parking, 11
 parking warden tour, 11
 partition coloring, 187
 path polytope, 247
 permanent, 367
 pivot algorithm, 120
 planar graph, 351
 polyhedral combinatorics, 251
 polyhedral embedding, 19
 polynomial algorithm, 301
 polynomial algorithms, 115
 polynomial instances, 153
 power-law, 285
 probabilistic constraint, 140

 problem contraction, 3
 problem reduction, 89
 protein, 337, 341
 protein molecules, 337
 protein structure alignment, 341
 protein threading problem, 341
 pseudo backbone, 3

 quadratic semi-assignment problem, 109
 quasi-threshold graphs, 23

 Ramsey Theory, 40
 randomization, 361
 recognition algorithm, 229
 rectangle packing, 84
 reducible flow graph, 229
 reformulation linearization technique, 109
 regular graphs, 55
 replacement model, 285
 rerouting, 97
 resistance of a graph, 64
 resource allocation, 68
 restricted entries, 367
 RLT, 109
 robustness, 171
 rounding heuristics, 239
 RWA, 225

 scheduling, 210
 secret santa, 269
 separation, 125
 set-coloring, 195
 shortest path, 93, 239
 signless Laplacian, 157
 SLD, 225
 sliding shortest path, 97
 spanning tree, 120
 split disjunctions, 144
 stability, 149
 stability number, 153
 stable graphs, 149
 stable set polytope, 251
 star coloring, 199
 stochastic dominance, 137

stochastic integer programming, 137
stochastic network, 221
stochastic programming, 140
structural characterizations, 281
submodular functions, 347
successive shortest paths, 239
symmetry, 361
system, 361

tabu search, 163
time-expanded network, 221
total chromatic number, 55
train timetabling, 171
traveling salesman problem, 3, 7
two-stage knapsack problem, 140

unconstrained quadratic programming,
105
undirected graphs, 301
unit disk graph, 35
upper bounding scheme, 191
upper bounds, 191

value, 361
vehicle routing problem, 215
vertex degree, 120
VLSI design, 35, 265

WDM, 225
Weber set, 361
window based approach, 3