16th Cologne-Twente Workshop on Graphs and Combinatorial Optimization

CNAM
Paris, France
June 18-20, 2018

Proceedings of the Workshop

General Chair: Leo Liberti
Editors: Emiliano Traversi, Fabio Furini, Leo Liberti

le cnam — École Polytechnique Université Paris-Saclay — Université Paris 13

CTW 2018 *(CNAM, Paris, France, 18-20 June)*

This conference is supported by a hell of an organizing committee. Special thanks go to Amélie Lambert (local arrangements), Lucas Létocart (website), Fabio Furini (email), Emiliano Traversi (proceedings). All complaints should go to Leo Liberti (sigh).

Schedule

	Mon 18 June						**Tue 19 June**						**Wed 20 June**					
	Room PP		*Room Z*		*Room Y*		*Room PP*		*Room Z*		*Room Y*		*Room PP*		*Room Z*		*Room Y*	
09:00-09:30	Registration (hall) and opening (PP)																	
09:30-10:00	*Complexity*	Anapolska	*Math. Progr. I*	Iommazzo	*Networks I*	Oustry	*Games II*	Yang	*Transportation I*	Bauguion	*Energy I*	Schwenk	*Graphs IV (Cordone)*	Kumbargoudra	*Math. Progr. III*	Aoudia	*Energy II*	Vanier
10:00-10:30		Pradhan		Lee		Gunnec		Furini		Righini		Thomopulos		Tian		Traversi		Mencarelli
10:30-11:00	coffee (hall)						coffee (hall)						coffee (hall)					
11:00-11:30	**Seiller (plenary, PP)**						**Wiegele (plenary, PP)**						*Graphs V*	Zheng	*Transportation II*	Pisacane	*Games III*	Boehnlein
1130-12:00														Behmaram		Bruglieri		Pacifici
12:00-14:00	lunch (on your own)						lunch (on your own)						Closing (PP)					
14:00-14:30	*Graphs I*	Nguyen	*Algorithms I*	Hommelsheim	*Graph Embeddings*	Silva	*Clustering*	Edelmann	*Math. Progr. II*	Francois	*Scheduling*	Pan						
14:30-15:00		Gomes da Silva		Vernet		Serocold		Gentile		Casazza		Schaudt						
15:00-15:30		Hossain		Vandomme		Lavor		Cordone		Marinelli		Nicosia						
15:30-16:00	coffee (hall)						coffee (hall)											
16:00-16:30	*Graphs II*	Obreja	*Comb. Opt. (Schrader)*	Vretta	*Games I*	Lozovanu	*Graphs III*	Gishboliner	*Algorithms II*	Klootwijk	*Networks II*	Ghanem						
16:30-17:00		Wolfler		Apke		Kern		Hu		Verma		Baste						
17:00-17:30								Del-Vecchio		Weller		Danisch						
19:00-21:00							Cocktail (salle des textiles)											

Did you know that CNAM hosts a Sciences Museum? This is one of the most crucial places in the novel "Foucault's Pendulum" by Umberto Eco (possibly my favorite writer). Many years ago I had applied to an assistant professorship at CNAM. I did not get the position, but during the interview I could not refrain from declaring that one of my strong motivations to apply was working in a place celebrated in a novel I loved. The hiring committee burst out laughing, and maybe that's why I wasn't offered the position. In any case you should go and visit the museum (same building, different entrance). Do not miss the part of the museum which hosts Foucault's pendulum, which hangs from the dome of the church of St. Martin-des-Champs (literally: St. Martin-in-the-Fields, which describes a sister church in London, equally central, but of a different confession I think).

The seminar rooms are the *Paul Painlevé* (PP), the *Robert Faure* (Z) and the *Jean-Baptiste Say* (Y) amphitheatres, located in Access 1, lower ground floor. Opening, plenary and closing sessions will take place in the PP amphitheatre. The cocktail event on Tuesday evening will take place in the *salle des textiles* room, located in Access 3, 1st floor. Coffee pauses will take place in the hall before the three amphitheatres. http://cedric.cnam.fr/~courtiep/planCnam/plan_Cnam_3e_arrondissement.html

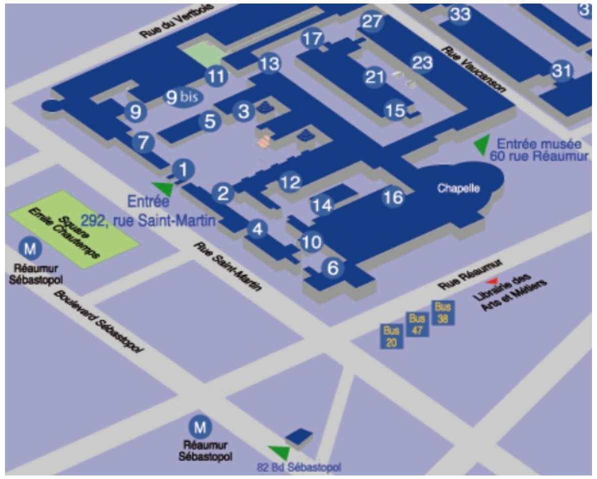
Session chairs. The last speaker of the session will chair the session, with two exceptions for PhD-only sessions: *Combinatorial Optimization* (Mon 18, Room PP, 16-17) chaired by R. Schrader, and *Graphs III* (Wed 20, Room PP, 9:30-10:30) chaired by R. Cordone. Session chairs must remind speakers to load up slides on laptops, and keep the sessions on time. Session chairs are encouraged to be cruel and despotic as regards times allotted, since there are parallel sessions. If a speaker will not get your hints, standing is often not enough: just cut him/her short and invite the next speaker (as the last speaker in the session, you have every incentive to do so, but please don't be the chair who overruns his own time slot). Conversely, if a speaker ends before the time is up, you should encourage some questions/discussion/debate: e.g. invite questions from the audience and leave a pause long enough to be slightly awkward, then possibly someone will ask a question just to fill in the horrible silence, and then other questions may follow. If no-one asks, you can start off the debate by asking a session yourself. In any case, keep all slots to exactly 30 minutes (parallel sessions regime).

CTW 2018 *(CNAM, Paris, France, 18-20 June)*

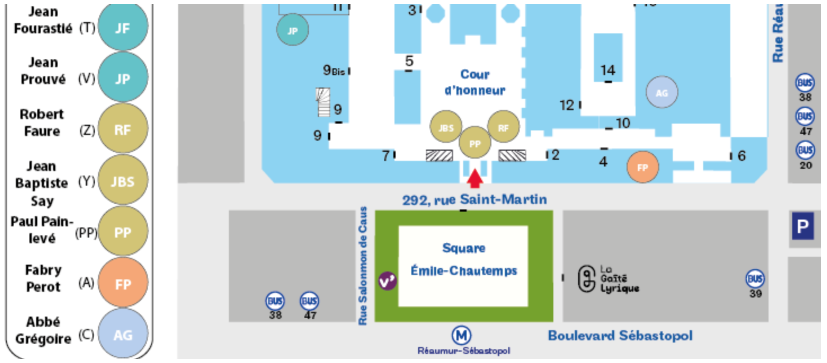
Invited speakers

Thomas Seiller, Univ. Paris-Nord, *Mon 18, PP, 11-12*
From Proofs to Programs, Graphs and Dynamics. Geometric perspectives on computational complexity.

Angelika Wiegele, Alpen-Adria Univ. Klagenfurt, *Tue 19, PP, 11-12*
Modeling and Solving Combinatorial Optimization Problems using Semidefinite Programming



Enter CNAM by the entrance labelled "1". The amphitheatres are underground, underneath the entrance court (see picture below). The "Salle des textiles" (where the cocktail event takes place) is labeled by "3", on the first floor.



The proceedings of this workshop are distributed in a PDF file which is available for download at www.lix.polytechnique.fr/~liberti/ctw18-proceedings.pdf.

A special issue of Discrete Applied Mathematics will be dedicated to the topics of the CTW18. Watch out for calls for papers to this issue during summer/autumn/winter 2018.

Speaker	**Title**	***Session***
Anapolska	Minimum Color-Degree Perfect b-Matchings	*Complexity*
Aoudia	Star forest polytope on complete graph	*Math. Progr. III*
Apke	A Characterization of Interval Orders with Semiorder Dimension Two	*Comb. Opt.*
Baste	Temporal matching in link stream: kernel and approximation	*Networks II*
Bauguion	Multimodal transportation plan adjustment with passengers behaviour constraints	*Transportation I*
Behmaram	On matching and distance property of m-barrele Fullerene	*Graphs V*
Boehnlein	Make or Buy: Revenue Maximization in Stackelberg Scheduling	*Games III*
Bruglieri	The Electric Vehicle Relocation Problem in Carsharing Systems with Collaborative Operators	*Transportation II*
Casazza	Dual bounds for a Maximum Lifespan Tree Problem	*Math. Progr. II*
Cordone	Some polynomial special cases for the Minimum Gap Graph Partitioning Problem	*Clustering*
Danisch	A Modular Overlapping Community Detection Algorithm: Investigating the "From Local to Global" Approach	*Networks II*
Del-Vecchio	A new centrality measure: spectral closeness.	*Graphs III*
Edelmann	Graph partitioning using matrix differential equations	*Clustering*
François	Mixed Integer Linear Programming Approach for a Distance-Constrained Elementary Path Problem	*Math. Progr. II*
Furini	Attacking the Clique Number of a Graph	*Games II*
Gentile	An algorithm for computing lower bounds for the Microaggregation problem	*Clustering*
Ghanem	How to exploit structural properties of dynamic networks to detect nodes with high temporal closeness	*Networks II*
Gishboliner	A Generalized Turan Problem and its Applications	*Graphs III*
Gomes Da Silva	Equitable total chromatic number of two classes of complete r-partite p-balanced graphs	*Graphs I*
Gunnec	Influence Maximization in Social Networks under Deterministic Linear Threshold Model	*Networks I*
Hommelsheim	Robust Matching Augmentation	*Algorithms I*
Hossain	Multicoloring of Pattern Graphs for Sparse Matrix Determination	*Graphs I*
Hu	On the spectra of general random mixed graphs	*Graphs III*
Iommazzo	A methodology for addressing the Algorithm Configuration problem on mathematical programming solvers	*Math. Progr. I*
Kern	The asymptotic price of anarchy for k-uniform congestion games	*Games I*
Klootwijk	Probabilistic Analysis of Optimization Problems on Generalized Random Shortest Path Metrics	*Algorithms II*
Kumbargoudra	Total k-rainbow Domatic Number	*Graphs IV*
Lavor	New advances on the branch-and-prune algorithm for the discretizable molecular distance geometry problem	*Graph Embeddings*
Lee	Gomory by column generation	*Math. Progr. I*
Lozovanu	Nash Equilibria in Mixed Stationary Strategies for m-Player Cyclic Games on Networks	*Games I*
Marinelli	A star-based reformulation for the maximum quasi-clique problem	*Math. Progr. II*
Mencarelli	A Multiplicative Weights Update Algorithm for a Class of Pooling Problems	*Energy II*
Nguyen	On some tractable constraints on paths in graphs and in proofs	*Graphs I*
Nicosia	Single machine scheduling with bounded job rearrangements	*Scheduling*
Obreja	Extremal Graphs with respect to the Modified First Zagreb Connection Index	*Graphs II*
Oustry	Optimal Deployment of Wireless Networks	*Networks I*
Pacifici	Two Stackelberg Knapsack games	*Games IV*
Pan	A hybrid heuristic for multi-activity tour scheduling	*Scheduling*
Pisacane	Solving the Green Vehicle Routing Problem with Capacitated Alternative Fuel Stations	*Transportation II*
Pradhan	Algorithmic aspects of neighborhood total domination in graphs	*Complexity*
Righini	Dynamic programming for the Electric Vehicle Orienteering problem with multiple technologies	*Transportation I*
S. Schaudt	Parallel machine scheduling with unit time distinct due windows	*Scheduling*
Schwenk	A Green Energy Grid Coupling Problem (GEGCP)	*Energy I*
Serocold	Rigidity of 1-coordinated frameworks in 2 dimensions	*Graph Embeddings*
Silva	Graphs with at most one crossing	*Graph Embeddings*
Tian	Sufficient degree conditions for traceability of claw-free graphs	*Graphs IV*
Thomopulos	A Constrained Shortest Path formulation for the Two-Reservoir Hydro Unit Commitment Problem	*Energy I*
Traversi	Decomposition Methods for Quadratic Programming	*Math. Progr. III*
Vandomme	Fully leafed induced subtrees (extended abstract)	*Algorithms I*
Vanier	Column Generation for the Energy-Efficient in Multi-Hop Wireless Networks Problem	*Energy II*
Verma	Edge Domination in subclasses of bipartite graphs	*Algorithms II*
Vernet	Successive Shortest Path Algorithm for Flows in Dynamic Graphs	*Algorithms I*
Vretta	A characterization for binary signed-graphic matroids	*Comb. Opt.*
Weller	Listing Conflicting Triples in Optimal Time	*Algorithms II*
Wolfler	A branch-and-price framework for decomposing graphs into relaxed cliques	*Graphs II*
Yang	On the One-Cop-Moves Game on Graphs	*Games II*
Zheng	Implicit heavy subgraph conditions for hamiltonicity of almost distance-hereditary graphs	*Graphs V*

Preface

This volume collects the abstracts of invited plenary and accepted contributed talks presented at the 16th Cologne-Twente Workshop (CTW) on graphs and combinatorial optimization, which took place at the Conservatoire National d'Arts et Métiers (CNAM) in Paris, 18-20 June 2018. Only those accepted abstracts for which the authors gave an explicit consensus of appearance are collected in this volume. The copyright of each single abstract rests with its authors. This volume is posted online at `http://www.lix.polytechnique.fr/~liberti/ctw18-proceedings.pdf`. Following tradition, a special issue of Discrete Applied Mathematics (DAM) dedicated to this workshop and its main topics of interest will be edited.

The CTW workshop series has been initiated by Ulrich Faigle, around the time he moved from Twente University to the University of Köln. After many CTW editions in Twente and Köln, it was decided that CTWs were mature enough to move about: in 2004 the CTW was organized in Villa Vigoni (Como, Italy) by F. Maffioli (Politecnico di Milano) and myself. Since then, the CTW visited Italy again many times and in many places (and more visits are planned), France and Turkey. The first edition of CTW in France occurred when I chaired the 8th edition of CTW in 2009 in Paris (at CNAM). In this second French edition of CTW, which I am again chairing, I aimed at more or less the same organization style as in 2009: the wonderful CNAM venue, which affords beautiful buildings, a wonderful science museum, a central Paris location close to lots of small, quaint and (relatively) cheap restaurants where you can while lunch breaks away; a cocktail on the second day; but other than that, an orga nization which is as simple as possible. For the first time, we shall not distribute paper copies of these proceedings. Instead, we shall distribute a single sheet of paper with the timetable and the list of talk titles with presenting authors (`http://www.lix.polytechnique.fr/~liberti/ctw18-program.pdf`).

The scientific program of this CTW edition (codenamed CTW18) includes two plenary talks (by Dr. Thomas Seiller and Prof. Angelika Wiegele), and 57 contributed (accepted) talks. The 57 accepted talks were selected from an initial set of 69: counter to computer science habits, this is not a "selective workshop". Having been initially set up by discrete applied mathematicians, it still follows the mathematical tradition whereby the main purpose of workshops is to present and discuss (possibly preliminary) results, rather than publish proceedings articles which are fully accomplished and have an archival nature. CTWs are not selective, and hence, in today's academic publish-or-perish worldview, not as attractive as they used to be. Are they still necessary? Among the initial motivations for CTWs we find a special attention to young (nonpermanent) researchers: MSc and PhD students as well as postdoctoral fellows. Another initial motivation was to provide a venue where preliminar y work could be presented and discussed. In this sense, this edition is perfectly in line with these two motivations (which I personally find very valid). At CTW18, 31 out of 57 contributed talks will be given by MSc, PhD or Postdocs. Half of the registered participants are MSc, PhD or Postdocs. While some talks relate to accomplished works, many have a preliminary/ongoing nature.

The governance of the CTW workshop series is assured by a "steering committee" which also acts as "programme committee", in the sense that it screens contributed abstracts and rejects those which are scientifically objectionable, written extremely poorly, or off topic. New members of the steering committee are sometimes chosen from CTW organizers. Currently, this committee counts 19 researchers from Germany, Italy, Turkey and France. Organizing committees are newly formed for each CTW edition. This year we have Fabio Furini (Paris-Dauphine), Amélie Lambert (CNAM), Lucas Létocart (Paris-Nord), Ivana Ljubic (ESSEC, Paris), Emiliano Traversi (Paris-Nord), Roberto Wolfler Calvo (Paris-Nord) and myself (CNRS & Ecole Polytechnique).

Not every CTW edition features invited plenaries, but this one does. Two young and brilliant researchers were invited: Thomas Seiller and Angelika Wiegele. Thomas is a CNRS researcher affiliated to the Computer Science Dept. (LIPN) at Paris-Nord. His research focuses on a certain unusual semantics for linear logic which holds some promise as a tool for separating complexity classes. Although this topic is far from the usual CTW crowd, I believe it is important enough that this community should know about it. Thomas was asked to give a "tutorial" on this line of research. Angelika, an associate professor at the Mathematics Dept. of Alpen-Adria University in Klagenfurt, Austria, is a well-known member of the mathematical programming community. She specializes in semidefinite programming applied to combinatorial optimization problems. She is one of those rare researchers who pursue the whole "pipeline" of a scientific result in mathematical programming, from theorems through algorithms to software (see e.g. `doi.org/10.1007/s10107-008-0235-8` to `biqmac.uni-klu.ac.at` and `biqbin.fis.unm.si`).

I very much hope you will all enjoy this 2018 edition of CTW.

Leo Liberti
CTW18 General Chair
CNRS LIX, Ecole Polytechnique

Organization

The CTW18 venue is the Conservatoire National d'Arts et Métiers (CNAM) in Paris (lecture halls PP, Y and Z) located in the third arrondissement of Paris (France).

The CNAM has several sites, and the rooms of CTW18 are located in the main site, 292 rue Saint-Martin, 75003 Paris.

The seminar rooms are the *Robert Faure* (Z), *Paul Painlevé* (PP) and *Jean-Baptiste Say* (Y) amphitheatres, located in Access 1, lower ground floor. The cocktail event on Tuesday evening will take place in the *salle des textiles* room, located in Access 3, 1st floor.

Scientific Committee:

- Ali Fuat Alkaya (U Marmara)
- Alberto Ceselli (U Milano)
- Roberto Cordone (U Milano)
- Ekrem Duman (U Ozyegin)
- Ulrich Faigle (U Koeln)
- Johann L. Hurink (U Twente)
- Leo Liberti (CNRS & École Polytechnique)
- Bodo Manthey (U Twente)
- Gaia Nicosia (U Roma Tre)
- Andrea Pacifici (U Roma Tor Vergata)
- Britta Peis (RWTH Aachen)
- Stefan Pickl (UBw München)
- Bert Randerath (Technische Hochshule Koeln)
- Giovanni Righini (U Milano)
- Heiko Roeglin (U Bonn)
- Oliver Schaudt (U Koeln)
- Rainer Schrader (U Koeln)
- Rüdiger Schultz (U Duisburg-Essen)
- Frank Vallentin (U Koeln)

Local Organization:

- Fabio Furini (U Paris Dauphine)
- Amélie Lambert (CNAM)
- Lucas Létocart (U Paris XIII)
- Leo Liberti (CNRS & École Polytechnique)
- Ivana Ljubic (ESSEC)
- Evelyne Rayssac (École Polytechnique, Paris)
- Emiliano Traversi (U Paris XIII)
- Roberto Wolfler Calvo (U Paris XIII)

Table of Contents

Minimum Color-Degree Perfect b-Matchings

Mariia Anapolska[1], Christina Büsing[1], Martin Comis[1]
Lehrstuhl II für Mathematik, RWTH Aachen University, Aachen, Germany
{buesing,comis}@math2.rwth-aachen.de

Abstract

We study the complexity of the Minimum Color-Degree Perfect-b-Matching Problem as an extension of the perfect b-matching problem on edge colored graphs. The problem is strongly NP-hard on bipartite graphs but can be solved in polynomial time on series-parallel graphs and trees for a fixed number of colors.

Keywords : *b-matchings, complexity, dynamic programming, series-parallel graphs, trees*

1 Introduction

Assignment problems are among the most famous combinatorial optimization problems. In their simplest form, every assignment problem consists of a set of agents A, a set of jobs J and a set of agent-job pairs $E \subseteq A \times J$ that define which agent can perform which job. The objective is to find an assignment of jobs to agents, such that every job is assigned to exactly one agent and every agent performs exactly one job. Graph-theoretically, this problem corresponds to the perfect matching problem in a bipartite graph which is known to be polynomial-time solvable by the Hungarian method. Unfortunately, for many applications this simple version of the assignment problem does not capture all relevant requirements. Therefore, various more complex forms of the assignment problem exist, e.g., [1, 3, 5].

In this paper we study a different extension, the so-called *minimum color-degree perfect b-matching problem* (Col-BM). In the Col-BM agents are allowed to perform multiple jobs and jobs are categorized into multiple classes, e.g., according to their location. Assuming that the execution of multiple jobs of the same category is desirable, we seek an assignment of jobs to agents such that the maximum number of differently categorized jobs assigned to one agent is minimized. In terms of graph theory Col-BM is a perfect b-matching problem on an edge colored graph with the objective of minimizing the maximum number of differently colored edges adjacent to any node. Before we formalize Col-BM, we define a node's color degree. Let $G = (V, E)$ be a graph with edge coloring $F_1 \dot\cup \ldots \dot\cup F_q = E$. For $M \subseteq E$ and $v \in V$, let $col_M(v)$ denote the set of colors in $\delta_M(v) := \{\{v, w\} \in M\}$, i.e., $col_M(v) := \{j \mid 1 \leq j \leq q \wedge \delta_M(v) \cap F_j \neq \emptyset\}$. The number of colors $|col_M(v)|$ is called the *(M)-color-degree* of v.

Definition 1 (Minimum Color-Degree Perfect b-Matching Problem (Col-BM)) Given an undirected graph $G = (V, E)$, an edge coloring $F_1 \dot\cup \ldots \dot\cup F_q = E$, and a mapping $b\colon V \to \mathbb{N}_0$, the Col-BM asks for a perfect b-matching $M \subseteq E$ with $|\delta_M(v)| = b(v)$ for all $v \in V$ of minimal maximum color degree $\max_{v \in V} |col_M(v)|$.

Related Work Weighted b-matching problems (WBM) are polynomial-time solvable by an extension of Edmonds' blossom algorithm. Chen et al. [3] studied the conflict aware WBM on bipartite graphs (CA-WBM) in which nodes may be in conflict and should therefore have disjoint neighborhoods. They show that CA-WBM is NP-hard and propose a greedy approximation algorithm. The diverse WBM (D-WBM) studied by Ahmed et al. [1] can be considered as the counterpart to the Col-BM. Given a weighted edge colored bipartite graph, a b-matching

M satisfying upper and lower bounds on $\delta_M(v)$ is sought, such that the weight of edges adjacent to a node is equally distributed amongst all colors in order to ensure diversification. Instead of employing a max-min approach as it is done in this paper, the authors encourage diversification by minimizing a quadratic function that penalizes uneven weight-color distributions. The authors present a greedy algorithm for D-WBM and claim that D-WBM is NP-hard.

Contribution In this paper we study the complexity of the Col-BM. By a reduction from (2B,3)-SAT, we show the problem's strong NP-hardness on bipartite graphs with 2 colors and conclude that there can be no $(2-\varepsilon)$-approximation algorithm unless P=NP (Section 2). Finally, we show that Col-BM with a fixed number of colors is polynomial-time solvable by dynamic programming on series-parallel graphs and trees (Section 3).

2 Complexity

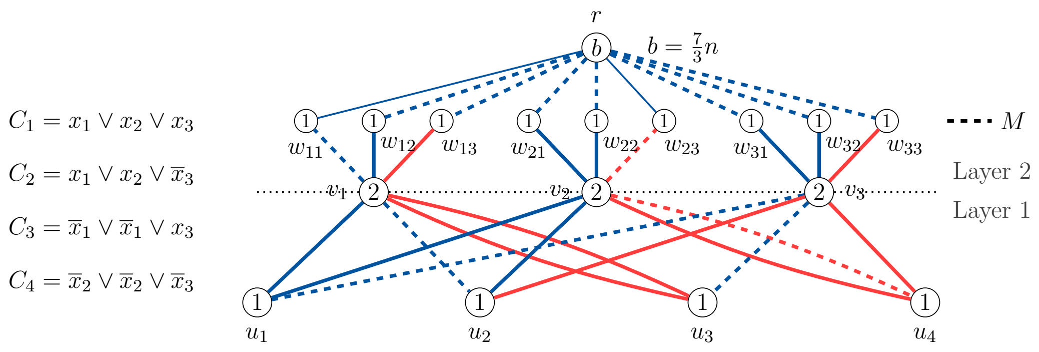
The Col-BM problem reduces to a simple, polynomial-time solvable perfect matching problem if $b(v) = 1$ for all $v \in V$. We show at the end of this section that already for $b\colon V \to \{1, 2\}$ the problem becomes strongly NP-hard. We begin with an intermediate statement.

Theorem 1 *The Col-BM on bipartite graphs with at most 2 colors is strongly NP-hard.*

Proof : We reduce (2B,3)-SAT, a specialization of the 3SAT problem in which each clause consists of exactly 3 literals and every literal occurs exactly twice, to Col-BM. The (2B,3)-SAT problem is strongly NP-complete [2]. Let $\mathcal{I}$ be a (2B,3)-SAT instance with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. We construct a Col-BM instance $\widetilde{\mathcal{I}}$ as follows: the graph $G = (V, E)$ is composed of two layers (Fig. 1). Layer 1 contains two sets of nodes $V := \{v_1, \ldots, v_n\}$ and $U := \{u_1, \ldots, u_m\}$ representing the variables and clauses of $\mathcal{I}$, respectively. We connect the node sets V and U via the following edges: blue edges $\{v_i, u_j\}$ for all positive literals $x_i \in C_j$ and red edges $\{v_i, u_j\}$ for all negated literals $\overline{x}_i \in C_j$. Finally, we set $b(v_i) = 2$ for $1 \le i \le n$ and $b(u_j) = 1$ for $1 \le j \le m$.

Layer 1 is bipartite and since $3m = 4n$ we have $\sum_{i=1}^{n} b(v_i) > \sum_{j=1}^{m} b(u_j)$. To ensure the existence of a perfect b-matching, we construct a second layer of G: For each vertex $v_i \in V$ we introduce three vertices $w_{i,1}, w_{i,2}, w_{i,3}$, edges $\{v_i, w_{i,1}\}$, $\{v_i, w_{i,2}\}$ colored in blue, and an edge $\{v_i, w_{i,3}\}$ colored red. Let $W := \{w_{i,k} \,|\, 1 \le i \le n \wedge 1 \le k \le 3\}$ and set $b(w_{i,k}) = 1$ for all $w_{i,k} \in W$. Finally, a vertex r with $b(r) = \frac{7n}{3}$ is introduced and connected to all vertices $w_{i,k} \in W$ with blue edges . Note that $\frac{7n}{3}$ is integer as $3 \mid n$.

We show: $\mathcal{I}$ is a Yes-instance if and only if $\widetilde{\mathcal{I}}$ has a perfect b-matching M with maximum color degree $\max_{v \in V(G)} |col_M(v)| = 1$. Let M be such a perfect b-matching. For each $v_i \in V$, $|col_M(v_i)| = 1$ and we set $x_i = $ true, if $\delta_M(v_i)$ is blue, and $x_i = $ false if $\delta_M(v_i)$ is red. For every vertex $u_j \in U$ exactly one edge $e_{ij} = \{v_i, u_j\}$ is in M as $b(u_j) = 1$. If e_{ij} is blue, $x_i \in C_j$ and hence our choice $x_i = $ true verifies clause C_j. Analogously, if e_{ij} is red, $\overline{x}_i \in C_j$ and by setting $x_i = $ false the clause C_j is verified which implies that x is a satisfying assignment of $\mathcal{I}$.



$C_1 = x_1 \vee x_2 \vee x_3$
$C_2 = x_1 \vee x_2 \vee \overline{x}_3$
$C_3 = \overline{x}_1 \vee \overline{x}_1 \vee x_3$
$C_4 = \overline{x}_2 \vee \overline{x}_2 \vee \overline{x}_3$
r
b
$b = \frac{7}{3}n$
w_{11}
w_{12}
w_{13}
w_{21}
w_{22}
w_{23}
w_{31}
w_{32}
w_{33}
v_1
v_2
v_3
u_1
u_2
u_3
u_4
M
Layer 2
Layer 1

FIG. 1: Construction of the perfect b-matching

For the other direction let x be a satisfying truth assignment of $\mathcal{I}$. We construct the perfect b-matching M of G as follows: for each clause C_j choose a verifying literal x_i ($\overline{x}_i$) and add the corresponding blue (red) edge $e_{ij} = \{v_i, u_j\}$ to M. This way, we select m edges in Layer 1 and for all $u_j \in U$ it holds $|\delta_M(u_j)| = b(u_j) = 1$. Since x_i and $\overline{x}_i$ cannot be simultaneously satisfied in x, $\delta_M(v_i)$ contains only edges of the same color for all $v_i \in V$. It remains to extend M to Layer 2: if $v_i \in V$ is incident to exactly one blue edge $e_{ij} \in M$, add $\{\{v_i, w_{i,1}\}\}$ to M; if $v_i \in V$ is incident to exactly one red edge $e_{ij} \in M$, add $\{\{v_i, w_{i,3}\}\}$ to M; if $v_i \in V$ is incident to no edge in M, add $\{\{v_i, w_{i,1}\}, \{v_i, w_{i,2}\}\}$ to M. This ensures $|\delta_M(v_i)| = 2$ and $|col_M(v_i)| = 1$ for all $v_i \in V$. As M contains exactly m edges between V and U (Fig. 1), $2n - m = \frac{2n}{3}$ vertices in W are matched to V. All remaining $3n - \frac{2n}{3} = \frac{7}{3}n$ vertices in W are matched to r. This ensures $|\delta_M(r)| = b(r)$ with $|col_M(r)| = 1$ and consequently M is a perfect b-matching of $\widetilde{\mathcal{I}}$ with $\max_{v \in V(G)} |col_M(v)| = 1$. $\square$

Corollary 1 *There can be no $(2-\varepsilon)$-approximation algorithm for Col-BM unless P=NP.*

By eliminating vertex r in Layer 2 of G, extending W by $\{w_{i,4} \mid 1 \le i \le n\}$ with $b(w_{i,4}) = 1$, and connecting all pairs of nodes in W by blue edges our reduction only requires two b-values.

Corollary 2 *The Col-BM with $b\colon V \to \{1,2\}$ and at most 2 colors is strongly NP-hard.*

3 Series-parallel Graphs and Trees

In this section we consider the Col-BM on series-parallel graphs and trees and show that for a fixed number of colors Col-BM can be solved in polynomial-time by dynamic programming.

Definition 2 A *(2-terminal) series-parallel (SP) graph* with two distinguished nodes s and t called *source* and *sink*, respectively, is defined recursively. An edge $\{s,t\}$ is SP; if G_1, G_2 are two SP-graphs with sources s_1, s_2 and sinks t_1, t_2, then the graph resulting from identifying t_1 with s_2 is SP (called the *series composition of* G_1 *and* G_2), and the graph resulting from identifying s_1 with s_2 and t_1 with t_2 is SP (called the *parallel composition of* G_1 *and* G_2).

Every SP graph G can be associated with a *decomposition tree* $T = T(G)$, a rooted binary tree with nodes corresponding to the subgraphs of G appearing in the recursive construction. Leaves correspond to edges in G, and inner nodes are of two types: S-nodes correspond to the series-composition of the graphs associated with their child nodes, and P-nodes analogously correspond to their parallel composition. By construction, the root r of T corresponds to G itself. A decomposition tree of a series-parallel graph can be computed in linear time [4]. Note that the graph constructed in the proof of Theorem 1 is, in general, not series-parallel.

We propose a dynamic program to solve the Col-BM on SP-graphs. Let $G = (V, E)$ be SP with edge coloring $F_1 \dot\cup \ldots \dot\cup F_q = E$, decomposition tree T, and $b\colon V \to \mathbb{N}_0$. For $v \in V(T)$, let G_v denote the subgraph of G with source s_v and sink t_v corresponding to v.

We introduce labels $lab_v = (\alpha, \sigma, \beta, \tau)$ for all $v \in V(T)$. The parameters $0 \le \alpha \le b(s_v)$ and $0 \le \beta \le b(t_v)$ define new smaller b-values at s_v and t_v, respectively while the *color vectors* $\sigma, \tau \in \{0,1\}^q$ define the required set of colors incident to s_v and t_v. Thus we call all $M \subseteq E(G_v)$ with $|\delta_M(u)| = b(u)$ for all $u \in V(G_v) \setminus \{s_v, t_v\}$, $|\delta_M(s_v)| = \alpha$, $|\delta_M(t_v)| = \beta$, $col_M(s_v) = \sigma^{-1}(\{1\})$ and $col_M(t_v) = \tau^{-1}(\{1\})\}$ lab_v-restricted matchings, and define the lab_v-restricted Col-BM as

$$\min_{M \subseteq E(G_v)} \Big\{ \max_{u \in V(G_v)} |col_M(u)| \;\Big|\; M \text{ is } lab_v\text{-restricted matching in } G_v \Big\}.$$

The cost $c_v(lab_v)$ is the optimal solution value of the lab_v-restricted Col-BM. We compute label costs recursively. If $v \in V(T)$ is a leaf in T, the graph G_v consists of one edge e and the label cost computation is straightforward. If $v \in V(T)$ is an S-node with children ℓ and w, then $s_v = s_\ell$, $t_v = t_w$, and $t_\ell = s_w =: x$. Every $(\alpha_v, \sigma_v, \beta_v, \tau_v)$-restricted matching M of G_v is composed of an $(\alpha_v, \sigma_v, \beta_\ell, \tau_\ell)$-restricted matching $M_\ell \subseteq G_\ell$ and an $(\alpha_w, \sigma_w, \beta_v, \tau_v)$-restricted matching $M_w \subseteq G_w$. The value $b(x)$ is split in β_ℓ and $\alpha_w = b(x) - \beta_\ell$, and it holds $col_M(x) = col_{M_\ell}(x) \cup col_{M_w}(x)$.

The maximum color-degree of M is $\max\{c_\ell(\alpha_v, \sigma_v, \beta_\ell, \tau_\ell),\ c_w(b(x)-\beta_\ell, \sigma_w, \beta_v, \tau_v),\ |col_M(x)|\}$ with $|col_M(x)| = \|\tau_\ell \vee \sigma_w\|_1$ where $\vee$ denotes the logical bitwise-or operator. Hence, we can compute the cost $c_v(\alpha_v, \sigma_v, \beta_v, \tau_v)$ by considering all possible choices for β_ℓ, τ_ℓ and σ_w, that is

$$c_v(lab_v) = \min_{0\le k\le b(x),\ \tau_\ell,\sigma_w\in\{0,1\}^q} \max\{c_\ell(\alpha_v, \sigma_v, k, \tau_\ell),\ c_w(b(x)-k, \sigma_w, \beta_v, \tau_v),\ \|\tau_\ell \vee \sigma_w\|_1\}.$$

If $v \in V(T)$ is a P-node with children ℓ and w, then $s_\ell = s_w = s_v$, $t_\ell = t_w = t_v$, and every lab_v-restricted matching M is composed of an $(\alpha_\ell, \sigma_\ell, \beta_\ell, \tau_\ell)$-restricted matching M_ℓ and $(\alpha_w, \sigma_w, \beta_w, \tau_w)$-restricted matching M_w. The edges in $\delta_M(s_v)$ and $\delta_M(t_v)$ are split between M_ℓ and M_w such that $\alpha_v = \alpha_\ell + \alpha_w$ and $col_M(s_v) = col_{M_\ell}(s_\ell) \cup col_{M_w}(s_w)$ (analogously for t_v). Consequently, minimizing over all possible ways of splitting $\delta_M(s_v)$ and $\delta_M(t_v)$ yields

$$c_v(lab_v) = \min_{\substack{0\le k\le \alpha_v,\ \sigma_\ell\vee\sigma_w=\sigma_v\\ 0\le m\le \beta_v,\ \tau_\ell\vee\tau_w=\tau_v}} \max\{c_\ell(k, \sigma_\ell, m, \tau_\ell),\ c_w(\alpha_v-k, \sigma_w, \beta_v-m, \tau_w),\ \|\sigma_v\|_1,\ \|\tau_v\|_1\}.$$

The algorithm computes all label costs for every node $v \in V(T)$ proceeding from leaves to the root r. The optimal solution value can be calculated as $\min_{\sigma,\tau} c_r(b(s_r), \sigma, b(t_r), \tau)$.

Theorem 2 *Col-BM on SP graphs with a fixed number of colors can be solved in* $\mathcal{O}(|E|\cdot\|b\|_\infty^4)$.

Proof : The correctness of the algorithm can be shown by induction. As for its runtime, observe that for each node $v \in V(T)$, $\mathcal{O}(\|b\|_\infty^2 \cdot 4^q)$ labels must be computed. The computational complexity of computing labels is dominated by P-nodes. For P-nodes we compute the minimum of at most $\mathcal{O}(9^q \cdot \|b\|_\infty^2)$ maxima, and every maximum can be calculated in $\mathcal{O}(1)$ time. Since $|V(T)| = 2|E| - 1$, the total runtime of the algorithm is in $\mathcal{O}(|E| \cdot 36^q \cdot \|b\|_\infty^4)$. $\square$

We note that for all graphs, $\|b\|_\infty \le |E|$ and therefore our algorithm runs in polynomial time. Moreover, we can extend our algorithm to the Col-BM on trees as follows: given a Col-BM instance $\mathcal{I}$ with tree $T = (V, E)$, we construct an auxiliary SP graph G by adding a new vertex t, connecting it to all leaves of T and setting $b(t) = 0$. Then every perfect b-matching in G contains no edges from $\delta_G(t) = E(G)\setminus E$ and thus is a perfect b-matching in T.

4 Conclusion

In this paper we introduce the Col-BM and prove its strong NP-hardness on bipartite graphs with a fixed number of colors and show that Col-BM is $(2-\varepsilon)$-inapproximable. For SP-graphs and trees we propose a dynamic program solving Col-BM in polynomial time for a fixed number of colors. Future work will include research on efficient exact algorithms for other graph classes, particularly graphs with bounded treewidth. Furthermore, we take a closer look at the approximability of Col-BM.

References

[1] F. Ahmed, J. P. Dickerson, and M. Fuge. Diverse weighted bipartite b-matching. In *Proceedings of IJCAI-17*, pages 35–41, 2017.

[2] P. Berman, M. Karpinski, and A. D. Scott. Approximation hardness of short symmetric instances of max-3sat. 01 2003.

[3] C. Chen, L. Zheng, V. Srinivasan, A. Thomo, K. Wu, and A. Sukow. Conflict-aware weighted bipartite b-matching and its application to e-commerce. *IEEE Trans. on Knowl. and Data Eng.*, 28(6):1475–1488, June 2016.

[4] B. de Fluiter and H. Bodlaender. *Parallel algorithms for series parallel graphs.* PhD thesis, University of Utrecht, 1997.

[5] S. L. Tanimoto, A. Itai, and M. Rodeh. Some matching problems for bipartite graphs. *J. ACM*, 25(4):517–525, Oct. 1978.

Algorithmic aspects of neighborhood total domination in graphs

S. Banerjee, Anupriya Jha, D. Pradhan[*]

Department of Applied Mathematics
Indian Institute of Technology (ISM), Dhanbad
sumanta.banerjee5@gmail.com; jha.anupriya@gmail.com; dina@iitism.ac.in

Abstract

A set $D \subseteq V$ of a graph $G = (V, E)$ is called a *neighborhood total dominating* set of G if D is a dominating set of G and the subgraph of G induced by the open neighborhood of D has no isolated vertex. Given a graph G, Min-NTDS is the problem of finding a neighborhood total dominating set of G of minimum cardinality. The decision version of Min-NTDS is known to be NP-complete for bipartite graphs and chordal graphs via split graphs. In this paper, we first extend this NP-completeness result to undirected path graphs, chordal bipartite graphs, and planar graphs and then present a linear time algorithm for computing a minimum neighborhood total dominating set in proper interval graphs. We show that Min-NTDS cannot be approximated within a factor of $(1 - \varepsilon) \log |V|$, unless NP$\subseteq$DTIME$(|V|^{O(\log \log |V|)})$ and can be approximated within a factor of $O(\ln \Delta)$. Finally, we show that Min-NTDS is APX-complete for graphs of maximum degree 3.

Keywords : *Domination, total domination, neighborhood total domination, polynomial time algorithm, NP-complete, APX-complete.*

1 Introduction

A set D of vertices of a graph $G = (V, E)$ is a *dominating* set of G if every vertex in $V \setminus D$ is adjacent to some vertex in D. The *domination number* of a graph G, denoted by $\gamma(G)$, is the minimum cardinality of a dominating set of G. The concept of domination and its variations have many applications and have been widely studied in literature (see [4, 5]). A set D of vertices of a graph $G = (V, E)$ is a *total dominating* set of G if every vertex in V is adjacent to at least one vertex of D. The *total domination number* of a graph G, denoted by $\gamma_{\mathrm{t}}(G)$, is the minimum cardinality of a total dominating set of G. For extensive literature and survey of total domination in graphs, we refer to [6, 9].

In a graph $G = (V, E)$, the sets $N_G(v) = \{u \in V : uv \in E\}$ and $N_G[v] = N_G(v) \cup \{v\}$ denote the *open neighborhood* and the *closed neighborhood* of a vertex v, respectively. For a set $S \subseteq V$, $N_G(S) = \cup_{u \in S} N_G(u)$ and $N_G[S] = N_G(S) \cup S$. A total dominating set D can be seen as a dominating set D such that induced subgraph $G[D]$ has no isolated vertex. Looking the similar property of the open neighborhood of a dominating set D, Arumugam and Sivagnanam [1] introduced the concept of neighborhood total domination in graphs. Formally, a dominating set D of a graph G is called a *neighborhood total dominating* set, abbreviated as NTD-set if $G[N_G(D)]$, i.e., the subgraph of G induced by $N_G(D)$ has no isolated vertex. The *neighborhood total domination number*, denoted by $\gamma_{\mathrm{nt}}(G)$, is the minimum cardinality of a NTD-set of G.

Notice that in any graph without isolated vertices, every NTD-set is a dominating set and every total dominating set is a NTD-set. So the following observation follows.

[*]Corresponding Author

Observation 1 ([1]) *For any graph G without any isolated vertex, $\gamma(G) \leq \gamma_{\rm nt}(G) \leq \gamma_{\rm t}(G)$.*

Observation 1 motivates researchers to study the neighborhood total domination in graphs since the neighborhood total domination number lies between the domination number and the total domination number, the two most important domination parameters in graphs. Henning and Rad [7] continued the further study on neighborhood total domination in graphs and gave several bounds on the neighborhood total domination number. Henning and Wash [8] characterized the trees with large neighborhood total domination number. Mojdeh et al. [11] studied the neighborhood total domination related to a graph and its complement. Recently, the algorithmic complexity of MIN-NTDS has been studied by Lu et al. [10]. In particular, Lu et al. [10] proved that the decision version of MIN-NTDS is NP-complete for bipartite graphs and chordal graphs via split graphs and presented a linear time algorithm for computing a minimum NTD-set in trees. In this paper, we first extend the known NP-completeness result of the decision version of MIN-NTDS to undirected path graphs, chordal bipartite graphs, and planar graphs. We present a linear time algorithm for MIN-NTDS in proper interval graphs. We then present results on the hardness of approximation, approximation algorithm, and APX-completeness for MIN-NTDS.

2 NP-completeness

In this section, we provide a polynomial time reduction from the decision version of MIN-DOM-SET to the decision version of MIN-NTDS and using this reduction we prove that the decision version of MIN-NTDS is NP-complete for undirected path graphs, chordal bipartite graphs, and planar graphs. MIN-DOM-SET is defined as the problem of finding a minimum dominating set for a given graph.

Let $G = (V, E)$ be a graph. We construct a new graph $G' = (V', E')$, where $V' = V \cup \{a_v, b_v, c_v, x_v, y_v : v \in V\}$ and $E' = E \cup \{va_v, a_vb_v, b_vc_v, b_vx_v, c_vy_v\}$. Notice that for every $v \in V$, x_v and y_v are pendant vertices of G'. Now it can be proved that G has a dominating set of cardinality at most k if and only if G' has a NTD-set of cardinality at most $k + 2|V|$.

Notice that if G is a chordal bipartite (resp. a planar or an undirected path) graph, then G' is also a chordal bipartite (resp. a planar or an undirected path) graph. Since the decision version of MIN-DOM-SET is NP-complete for undirected path graphs [2], for planar graphs [3], and for chordal bipartite graphs [12], we have the following theorem.

Theorem 1 *The decision version of* MIN-NTDS *is NP-complete for undirected path graphs, chordal bipartite graphs, and planar graphs.*

3 Algorithm for MIN-NTDS in proper interval graphs

A graph G is called a *proper interval* graph if G is the intersection graph of a nonempty family of intervals on the real line such that no interval properly contains another interval. A vertex v of a graph G is called a *simplicial* vertex of G if $N_G[v]$ is a clique of G. An ordering $\sigma = (v_1, v_2, \ldots, v_n)$ is a *perfect elimination ordering* (PEO) of G if v_i is a simplicial vertex of $G_i = G[\{v_i, v_{i+1}, \ldots, v_n\}]$ for all $i, 1 \leq i \leq n$. A PEO $\sigma = (v_1, v_2, \ldots, v_n)$ of a chordal graph is a *bi-compatible elimination ordering*(BCO) if $\sigma^{-1} = (v_n, v_{n-1}, \ldots, v_1)$, i.e. the reverse of σ, is also a PEO of G. It is well known that a graph G is a proper interval graphs if and only if G has a BCO.

Let G be a connected proper interval graph with a BCO $\sigma = (v_1, v_2, \ldots, v_n)$. For each $v_i, 1 \leq i \leq n$, let $\ell(v_i) = \max\{\{i\} \cup \{k : v_iv_k \in E(G) \text{ and } k > i\}\}$.

We now present our algorithm, namely MNTDS-PIG(G) to compute a minimum NTD-set of a given connected proper interval graph G with at least 2 vertices. If G is a proper interval graph with at most two vertices, then it is easy to construct a minimum NTD-set of G.

Algorithm 1: MNTDS-PIG(G)

```
Input: A connected proper interval graph $G = (V, E)$ with at least 2 vertices;
Output: A minimum NTD-set $D$ of $G$;
Compute a BCO $\sigma = (v_1, v_2, \ldots, v_n)$ of $G$;
Initialize $D = \emptyset$;
$i = 1$;
while ($i \leq n$) do
    Let $\ell(v_i) = j$;
    if ($d_{G_i}(v_i) = 1$ and $i = 1$) then
        if ($v_j = v_n$) then
            $D = D \cup \{v_i, v_j\}$;
        else
            if ($\ell(\ell(\ell(v_j))) = \ell(\ell(v_{j+1}))$) then
                $D = D \cup \{v_j, v_{\ell(v_j)}\}$;
                $i = \ell(v_{\ell(v_j)}) + 1$;
            else
                $D \cup \{v_i, v_{\ell(v_{j+1})}\}$;
                $i = \begin{cases} \ell(\ell(v_{j+1})) + 1, & \text{if } |N_G(v_{\ell(v_{j+1})}) \cap \{v_{\ell(v_{j+1})+1}, \ldots, v_n\}| \geq 2 \text{ or } v_{\ell(v_{j+1})+1}v_{\ell(v_{j+1})-1} \in E; \\ \ell(v_{j+1}) + 1, & \text{otherwise.} \end{cases}$
    else
        if ($v_i$ is not dominated) then
            $D = D \cup \{v_j\}$;
            $i = \begin{cases} \ell(v_j) + 1, & \text{if } |N_G(v_j) \cap \{v_{j+1}, \ldots, v_n\}| \geq 2 \text{ or } v_{j+1}v_{j-1} \in E; \\ \ell(v_i) + 1, & \text{otherwise.} \end{cases}$
        else
            if (($d_{G_i}(v_i) = 0$) or ($d_{G_i}(v_i) = 1$ and $v_j = v_n$)) then
                $D = D \cup \{v_i\}$;
            else
                Let $\ell(v_{i+1}) = p$ and $\ell(v_p) = p'$;
                $D = D \cup \{v_p\}$;
                $i = \begin{cases} \ell(v_p) + 1, & \text{if } |N_G(v_p) \cap \{v_{p+1}, v_{p+2}, \ldots, v_n\}| \geq 2 \text{ or } v_{p+1}v_{p-1} \in E; \\ \ell(v_{i+1}) + 1, & \text{otherwise.} \end{cases}$
return $D$;
```

Theorem 2 MNTDS-PIG(G) *correctly computes a minimum* NTD*-set of a given proper interval graph G with at least 2 vertices in linear time.*

Proof : (Sketch:) Suppose that MNTDS-PIG(G) executes for k number of iterations. Then $k \leq n$. Let D_r, $1 \leq r \leq k$ be the set constructed by MNTDS-PIG(G) after the execution of the r-th iteration. We first prove that D_k is a NTD-set of G. Then by using the method of induction, we prove that for each r, $1 \leq r \leq k$, D_r is contained in some minimum NTD-set of the proper interval graph G. We can argue that all the steps of the algorithm can be executed in at most $O(n + m)$ time.

4 Hardness results and approximation algorithm

We establish the following two theorems corresponding to the lower bound and upper bound on the approximation ratio for MIN-NTDS.

Theorem 3 *Let $G = (V, E)$ be a graph with n vertices. Unless NP $\subseteq$ DTIME($n^{O(\log \log n)}$),* MIN-NTDS *cannot be approximated within a factor of $(1 - \varepsilon) \ln n$ for any $\varepsilon > 0$. The same holds for split graphs and bipartite graphs.*

Proof : (Idea) This can be proved by establishing an approximation preserving reduction from MIN-DOM-SET to MIN-NTDS.

Theorem 4 MIN-NTDS *in a graph* $G = (V, E)$ *can be approximated within an approximation ratio of* $2(\ln(\Delta(G) + 1) + 1)$.

By using the construction used in Theorem 1, we can establish an L-reduction from MIN-DOM-SET for graphs of degree at most 3 to show that MIN-NTDS is APX-complete for graphs of degree at most 4. Then we establish an L-reduction from MIN-NTDS for graphs of degree at most 4 to MIN-NTDS for graphs of degree at most 3. So we have the following theorem.

Theorem 5 MIN-NTDS *is APX-complete for graphs of degree at most* 3.

References

[1] S. Arumugam and C. Sivagnanam. Neighborhood total domination in graphs. *Opuscula Math.*, 31:519–531, 2011.

[2] K. S. Booth and J. H. Johnson. Dominating sets in chordal graphs. *SIAM J. Comput.*, 11(1):191–199, 1982.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1990.

[4] T.W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs.* Chapman & Hall/CRC Pure and Applied Mathematics. Taylor & Francis, 1998.

[5] T.W. Haynes, S. Hedetniemi, and P. Slater. *Domination in Graphs: Volume 2: Advanced Topics.* Chapman & Hall/CRC Pure and Applied Mathematics. Taylor & Francis, 1998.

[6] M. A. Henning. Recent results on total domination in graphs: A survey. *Discrete Math.*, 309:32–63, 2009.

[7] M. A. Henning and N. J. Rad. Bounds on neighborhood total domination in graphs. *Discrete Appl. Math.*, 161:2460–2466, 2013.

[8] M. A. Henning and K. Wash. Trees with large neighborhood total domination number. *Discrete Appl. Math.*, 187:96–102, 2015.

[9] M. A. Henning and A. Yeo. *Total domination in graphs.* Springer Monographs in Mathematics, Springer-Verlag New York, 2013.

[10] C. Lu, B. Wang, and K. Wang. Algorithm complexity of neighborhood total domination and $(\rho, \gamma_{\mathrm{nt}})$-graphs. *J. Comb. Optim.*, 35(2):424–435, 2017.

[11] D. A. Mojdeh, M. R. Sayed Salehi, and M. Chellali. Neighborhood total domination of a graph and its complement. *Australasian J. Combinatorics*, 65:37–44, 2016.

[12] H. Müller and A. Brandstädt. The NP-completeness of STEINER TREE and DOMINATING SET for chordal bipartite graphs. *Theor. Comput. Sci.*, 53(2-3):257–265, August 1987.

Dynamic programming for the Electric Vehicle Orienteering Problem with multiple technologies

Dario Bezzi[1], Alberto Ceselli[1], Giovanni Righini[1]

Dept. of Computer Science, University of Milan, Italy

dario.bezzi,alberto.ceselli,giovanni.righini@unimi.it

Abstract

We describe a bi-directional dynamic programming algorithm to solve the Electric Vechile Orienteering Problem, arising as a pricing sub-problem in column generation algorithms for the Electric VRP with multiple recharge technologies.

Keywords : *Combinatorial optimization, dynamic programming, shortest path.*

1 Problem description

The Electric Vehicle Routing Problem (EVRP) has been introduced by Erdogan and Miller-Hooks under the name of Green Vehicle Routing Problem in [1]. Several variations have been studied, including problem with time windows, partial recharges, multiple technologies and both exact and heuristic algorithms have been developed. Examples of heuristic algorithms for the EVRP are given in Felipe et al. [2], Schneider et al. [3] and Koc and Karaoglan [4]. More references on VRP variants involving the use of electric vehicles can be found in a recent and extensive survey by Pelletier et al. [5].

The computation of exact solutions is more challenging than for the classical VRP, because of the additional subproblem of deciding the optimal recharges at some points along the routes. An additional source of complexity is the presence of different recharge technologies, each one characterized by a unit cost and a recharge speed. Schiffer and Walther [6] recently considered a similar problem in the context of location-routing. Sweda et al. [7] studied the optimal recharge policy when the route is given. As with many other variations of the VRP, the most common choice to design effective exact optimization algorithms is to rely upon branch-and-cut-and-price, starting from a reformulation of the routing problem as a set covering or set partitioning problem, where each column represents the duty of a vehicle. For instance, Desaulniers et al. [8] developed a branch-and-price-and-cut algorithm for the exact solution of the EVRP with time windows. In this study we investigate the Electric Vehicle Orienteering Problem, arising as a pricing sub-problem when the EVRP is solved by branch-and-price and in particular we consider a dynamic programming algorithm for the case with multiple technologies.

2 Formulation

Let $G = (N \cup R, E)$ be a given weighted undirected graph whose vertex set is the union of a set N of customers and a set R of recharge stations. A distinguished station in R is the depot, numbered 0. A fleet of V identical vehicles, located at the depot, must visit the customers. All customers in N must be visited by a single vehicle; split delivery is not allowed. Each customer $i \in N$ is characterized by a demand and each vehicle has a capacity as in the classical Capacitated VRP. Vehicles are equipped with batteries of given capacity B. Recharge stations can be visited at any time; multiple visits to them is allowed and partial recharge is also allowed. We consider a set of different technologies for battery recharge. For each technology we assume a given recharge speed. When visiting a station, only one of the available technologies can be used.

All vertices $i \in R \cup N$ are also characterized by a service time, representing the time taken by pick-up/delivery operations for $i \in N$ or a fixed time to be spent to set-up the recharge for $i \in R$. The distance d_e and the travel time are known for each edge $e \in E$. The energy consumption is assumed to be proportional to the distance through a given coefficient π. The duration of each route (including service time, travel time and recharge time) is required not to exceed a given limit.

A feasible route must comply with capacity and duration constraints. Furthermore the level of the battery charge must be kept between 0 and B at any time. A set of feasible routes is a feasible solution if all customers are visited once and no more than V vehicles are used. The objective to be optimized is given by the overall recharge cost, consisting of a fixed cost and a variable cost. Since batteries allow for a limited number of recharge cycles during their operational life, we associate a fixed cost f with each recharge operation. The variable cost associated with a recharge operation at any station $i \in R$ is proportional to the amount of energy recharged, but it also depends on the recharge technology.

We indicate with Ω the set of all feasible routes. We associate a binary variable x_r with each feasible route $r \in \Omega$: Binary coefficients y_{ir} take value 1 if and only if customer $i \in N$ is visited along route $r \in \Omega$. We indicate by c_r the cost of each route $r \in \Omega$. With these definitions and notation we obtain the following ILP model (master problem):

$$\text{minimize} \sum_{r\in\Omega} c_r x_r \tag{1}$$

$$\text{s.t.} \sum_{r\in\Omega} y_{ir} x_r \geq 1 \qquad \forall i \in N \tag{2}$$

$$\sum_{r\in\Omega} x_r \leq V \tag{3}$$

$$x_r \in \{0,1\} \qquad \forall r \in \Omega. \tag{4}$$

At each node of a branch-and-bound tree the linear relaxation of the master problem is solved by column generation. We indicate by λ_i the non-negative dual variables vector corresponding to the covering constraints (2) and by μ the scalar non-negative dual variable corresponding to constraints (3) restated in $\geq$ form. With this notation, the expression of the reduced cost of a generic column r is

$$\hat{c}_r = c_r - \sum_{i\in N} \lambda_i y_{ir} + \mu.$$

3 The pricing sub-problem

The pricing problem, whose ILP formulation is not reported here for brevity, is a variation of the Orienteering Problem and it requires to find a minimum cost closed walk from the depot to the depot, not visiting any customer vertex more than once and not consuming more than a given amount of available resources (capacity, time and energy). Edges between stations can be traversed more than once. This problem is also a variation of the Resource Constrained Elementary Shortest Path Problem, in which the elementary path constraints are imposed only on a subset of vertices, the resources are partly discrete and partly continuous and one of the resources (energy) is renewable.

3.1 The algorithm

We have devised an exact pricing algorithm based on dynamic programming, where labels are associated with paths emanating from the depot and have the following form:

$$L = (u, S, \phi, \underline{t}, \underline{\hat{c}}, \underline{\Delta}, \overline{\Delta}, \underline{\delta}, \overline{\delta}),$$

where u is the endpoint of the path different from the depot, S is the set of customer vertices visited along the path, $\underline{t}$ is the minimum time required to traverse the path, $\underline{\hat{c}}$ is the minimum reduced cost of the path, $\underline{\Delta}$ and $\overline{\Delta}$ (scalar values) are the minimum and the maximum amount of residual energy that can exist in the battery when the vehicle reaches u from the depot, $\underline{\delta}$ and $\overline{\delta}$ (vectors with one component for each technology) are the lower and upper bounds on the total amount recharged with each technology along the path. For brevity, we indicate by P the polytope defined by the lower and upper bounds. The information conveyed by $\underline{t}$ and $\underline{\hat{c}}$ is indicated for convenience but it can be obtained from the knowledge of P.

Relying upon these definitions we developed and tested a dynamic programming algorithm to price out columns. Besides fathoming dominated states, the algorithm also relies on acceleration techniques such as bounding and state space relaxation.

In our talk we will present computational results obtained on benchmark instances from the literature on the pricing problem for the EVRP.

References

[1] S. Erdogan and E. Miller-Hooks, *A Green Vehicle Routing Problem*, Transportation Research Part E 48, 100-114, 2012.

[2] Á. Felipe, M.T. Ortuño, G. Righini and G. Tirado, *A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges*, Transportation Research Part E 71, 111-128, 2014.

[3] M. Schneider, A. Stenger and D. Goeke, *The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations* Transportation Science 48(4), 500-520, 2014.

[4] C. Koc and I. Karaoglan, *The green vehicle routing problem: A heuristic based exact solution approach*, Applied Soft Computing 39, 154-164, 2016.

[5] S. Pelletier, O. Jabali and G. Laporte, *Goods distribution with electric vehicles: review and research perspectives*, Transportation Science 50(1), 3-22, 2016.

[6] M. Schiffer and G. Walther, *The electric location routing problem with time windows and partial recharging*, European Journal of Operational Research 260(3), 995-1013, 2017.

[7] T.M. Sweda, I.S. Dolinskaya and D. Klabjan, *Optimal Recharging Policies for Electric Vehicles*, Transportation Science 51(2), 457-479, 2017.

[8] G. Desaulniers, F. Errico, S. Irnich and M. Schneider, *Exact Algorithms for Electric Vehicle-Routing Problems with Time Windows*, Operations Research 64(6), 1388-1405, 2016.

Gomory by column generation[*]

Jon Lee[1]

University of Michigan, Ann Arbor, Michigan

jonxlee@umich.edu

Abstract

Gomory cutting planes, for both the pure-integer and mixed-integer extensions of linear optimization, were introduced in the early 1960's. We present Gomory's ideas in a different algebraic manner than he did, resulting in a cleaner view of his finite-convergence proofs and simpler linear algebra to carry out.

Keywords : *Gomory, cutting plane, pure-integer, mixed-integer, column generation*

1 Introduction

Gomory cutting planes, both pure-integer and mixed-integer, are classically presented for the standard-form mixed-integer linear problem

$$\min\{c'x \;:\; Ax = b,\; x \geq 0,\; x_j \in \mathbb{Z} \text{ for } j \in \mathcal{J}\}, \qquad (P_{\mathcal{I}})$$

where A is $m \times n$, everything else is sized accordingly, and $\mathcal{J} \subset \{1, 2, \ldots, n\}$. This starting point leads to cutting-plane methods for $(P_{\mathcal{I}})$. The linear-algebra that we carry out in applying such a method is cumbersome, because for each cut we need to add a constraint (and slack variable), and the simplex-method bases for the continuous relaxation of $(P_{\mathcal{I}})$ grow in size. Still, Gomory could use this framework to make finitely-converging cutting-plane algorithms, employing the lexicographic dual-simplex algorithm (see [3] and [2]). Moreover, Gomory cuts eventually became practically relevant (see [1]).

2 Gomory in a column-generation framework

Our starting point is rather

$$\max\{y'b \;:\; y'A \leq c',\; y_i \in \mathbb{Z} \text{ for } i \in \mathcal{I}\}, \qquad (D_{\mathcal{I}})$$

where now $\mathcal{I} \subset \{1, 2, \ldots, m\}$. A nice feature of $(D_{\mathcal{I}})$ is that the dual of its continuous relaxation is the standard form linear-optimization problem $P_{\emptyset}$. By developing Gomory's cuts differently, using his same geometric reasoning but now with different linear algebra, we get cutting-plane methods for $(D_{\mathcal{I}})$ that are simple column-generation methods for $P_{\emptyset}$. We find a few interrelated benefits: (i) the linear-algebra is simpler to carry out, with simplex-method bases not growing in size, (ii) there is no need to appeal to the *dual* simplex method at all, and (iii) versions of our approach gain their finite convergence using the lexicographic *primal* simplex algorithm. To drive home the appeal of our pedagogy, our CTW presentation features a demonstration of a free `Matlab` tool for carrying out our Gomory (see [5])

[*]based on joint works with Qi He and Angelika Wiegele, presented in the papers [4] and [6]. Supported by ONR grants N00014-14-1-0315 and N00014-17-1-2296.

References

[1] Gérard Cornuéjols. Revival of the Gomory cuts in the 1990's. *Annals of Operations Research*, 149(1):63–66, 2007.

[2] Ralph E. Gomory. An algorithm for the mixed integer problem. Technical Report RM-2597, The RAND Cooperation, 1960.

[3] Ralph E. Gomory. An algorithm for integer solutions to linear programs. In *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York, 1963.

[4] Qi He and Jon Lee. Another pedagogy for pure-integer Gomory. *RAIRO - Operations Research*, 51(1):189–197, 2017.

[5] Jon Lee. *A First Course in Linear Optimization (Third Edition, version 3.00).* Reex Press, 2013–17. `https://github.com/jon77lee/JLee_LinearOptimizationBook`.

[6] Jon Lee and Angelika Wiegele. Another pedagogy for mixed-integer Gomory. *EURO Journal on Computational Optimization*, 5(4):455–466, 2017.

Optimal Deployment of Wireless Networks

Antoine Oustry[1], Marion Le Tilly[2]

[1] Ecole polytechnique, Palaiseau, France
antoine.oustry@polytechnique.edu

[2] Ecole polytechnique, Palaiseau, France
marion.le-tilly@polytechnique.edu

Abstract

This paper aims at providing a quantitative method to optimize the deployment of a wireless network. Firstly it presents a frequency-domain finite difference method to simulate wave propagation in a building. Secondly it proposes a Mixed-Integer Linear Programming formulation to minimize the cost of the network deployment taking into account the computed signal propagation, the WiFi demand at each point and the number of available channels.

1 Introduction

Nowadays, wireless network planning relies on intuition and experience with very limited use of simulation software. Therefore, it usually yields in approximated infrastructure placements providing suboptimal services. In this context, we aim to achieve a network planning methodology based on simulating the electromagnetic field strength within the deployment location, and using the simulated data as an input to various mathematical programming formulations. Therefore, we build our own simulator taking into account physical effects such as interference or diffraction, to generate data for our combinatorial program. Eventually, this program aims to provide wireless access to a whole building at the lowest cost, considering potential statistical changes in the wireless demands across the building.

2 Data Model

First, radio wave propagation is simulated in the target building, since this data is the basis of the combinatorial problem. While standard empirical methods were available, we chose to develop a more accurate method based on the simulation of a partial differential equation.

2.1 Classical methods

In practice, methods belong to one of the following category:

- **Empirical methods:** These methods, such as *the multi-wall model*, predict the average behavior of waves based on the distance as well as the number and the nature of walls between transmitter and receiver. These approaches are widely used for network design, thanks to their low computational load requirement. However these approaches are less accurate than the one presented below given that some physical effects are not taken into account, e.g *diffraction, self-interference or corridor effect.*
- **Ray-tracing methods:** These methods are based on geometrical optics: using the Fermat's principle of least time, it determines a ray's trajectory between source point and field locations enabling the computation of the propagation loss at those locations. Yet, it also does not take into account diffraction and self-interference, and its computational complexity is proportional to the number of rays launched by the source and grows exponentially with the number of reflections each ray undergoes.

2.2 Our frequency-domain finite difference method

Wave propagation in frequency domain: the Helmholtz equation. Inspired by both Chopard's ParFlow method [1] and the MR-FDPF method developed at *INSA Lyon* [3], the Helmholtz equation predicts the radio wave propagation in a deterministic manner, considering physical effects such as diffraction, self-interference or corridor effect. To begin with, we worked with a 2D environment. To do so, consider - as done in [3] - the classical propagation equation of a scalar wave: $\Delta u(x, y, t) - \mu\epsilon\partial_t^2 u(x, y, t) = -s(x, y, t)$, where $s(x, y, t)$ is a source term, $\epsilon(x, y)$ is the local electric permittivity and $\mu(x, y)$ the local permeability. The physical values ϵ and μ depend on the material : they represent the architecture of the floor.
Applying the Fourier transform to the wave equation eliminates the time differential, and adding a diffusive term permits not to overestimate the reflections on the walls and on the boundary ($\sigma \neq 0$, where σ is the electric conductivity). Eventually, it yields in the following complex Helmholtz equation : $\Delta\Psi + (\omega^2\mu\epsilon - i\omega\mu\sigma)\Psi = -S(x, y, \omega)$

Finite difference scheme: The Helmholtz equation is simulated through a finite difference scheme which consists in discretizing the rectangle $[0, L] \times [0, l]$ in a grid $[|0, N_x|] \times [|0, N_y|]$, using the same step Δ_x for both dimensions. Ψ_{jN_x+i} is an approximation of $\Psi(i\Delta_x, j\Delta_x)$. The classic discretization of the Laplacian leads to the following sparse linear system :

$$\forall k \in [1, N_xN_y], \Psi_{k+1} + \Psi_{k-1} + \Psi_{k+N_x} + \Psi_{k-N_x} + (\beta^2 n_k^2 - 4 - i(\Delta_x)^2\omega\alpha_k)\Psi_k = F_k$$

with the following conventions: $F_{jN_x+i} = -(\Delta_x)^2 S(i\Delta_x, j\Delta_x, \omega), \beta = \frac{\omega\Delta_x}{c_0}, \alpha_k = \mu_k\sigma_k$, $\forall i \notin [1, N_xN_y]\Psi_i = 0$. This sparse linear system can be efficiently solved by using an LU factorization of the system matrix. In practice, the Python library SuperLU [2] was used.

Time complexity: Thanks to the *SuperLU* library our simple discretization method achieves the same complexity as the MR-FDPF method [4]. For a given 2D map, a pre-computation is required in time $O(N_x^3)$ to factorize the system. The computation of the field created by a source in any given point is in $O(N_x^2 \log(N_x))$.

From 2D to 3D: To make the model fit reality, it is crucial to model indoor radio wave propagation in 3D environment. Trivially increasing the number of voxels in the methodology sketched above would yield an excessive complexity increase. Thus, the 2.5D empirical approach presented in [4] is more relevant to deal with 3D, it relies on the projections of the field in the floor k to compute the field in the floor $k + 1$, using on of these alternatives :

- *Field Projecting* models the 3D propagation by projecting the field map through the roof with an attenuation coefficient depending on the nature of the ceiling.
- *Source Projecting* consists in projecting the source (of the floor k) in the floor $k + 1$ with an attenuation factor and then in computing the 2D propagation in the floor $k + 1$ from this virtual source.
- A combination of the two latter alternatives.

3 Mathematical Programming formulation

The data from the simulation enabled to build a mixed-integer linear program - with stochastic constraints - which ensures wireless connection all over the building at minimum cost and takes into account wireless demand at each point of the building. To build this network, our model considers two types of equipment: wired access points (AP) and wireless repeaters. Both types of devices have different costs and capacities. Considering a 2D or 3D grid, $V_{clients}$ defines a set representing the points of the grid to cover and V_{cand} a set representing eligible positions for APs or repeaters. A point which has to to be covered and which is also a potential AP position is duplicated, therefore the union of both sets is empty. $V = V_{clients} \cup V_{cand}$. $V^* = V \cup \{r\}$ represents all the vertices of the grid including r the root of the graph, to which all APs have to be connected.

Parameters:

- The power gain matrix $P = (p_{i,j})_{(i,j)\in V^2}$ computed by the data generation model;
- $(C_i^A)_{i\in V_{cand}}$ such that $C_i^A > 0$ is the installation cost of an AP at i;
- $(C_i^R)_{i\in V_{cand}}$ such that $C_i^R > 0$ is the installation cost of a repeater at i. $C_i^R < C_i^A$;
- $\gamma^A > 0$ and $\gamma^R > 0$ are the maximum communication rate that an AP and a repeater can handle;
- The vector $(D_i)_{i\in V_{clients}}$ with $(D_i \geq 0)$ represents the bandwith demand at point i. These random variables are not independent. Moreover we set : $\forall i \in V_{cand}, D_i = 0$;
- $\forall j \in V, p^j_{max} = max_{i\in V_{cand}} p_{i,j}$
- $n_{max} > 0$ a maximal noise level which is the only control parameter;
- We define a capacity matrix $W = (w_{i,j})_{(i,j)\in(V^*)^2}$:
  - $\forall(i,j) \in V \times V_{cand}, w_{i,j} = Blog(1 + \frac{p_{i,j}}{n_{max}}) > 0$: maximal data rate from i to j.
  - $\forall(i,j) \in (V \times V_{clients}) \cup (\{r\} \times V^*) \cup (V_{clients} \times \{r\}), w_{i,j} = 0$
  - $\forall(i,j) \in V_{cand} \times \{r\}, w_{i,j} = M$ where M a real number such that $\sum_{i\in V_{clients}} D_i \leq M$ almost surely;

Thus we consider the capacited oriented graph $G = (V^*, (V^*)^2, W)$, and the purpose is to build a flow on G from the client points to the root by selecting relay nodes.

Decision variables:

- $(A_i)_{i\in V_{cand}} \in \{0,1\}^{V_{cand}}$: indicates the presence of an AP at i;
- $(A_i^c)_{i\in V_{cand}, c\in\mathcal{C}} \in \{0,1\}^{V_{cand}\times\mathcal{C}}$: indicates the presence of an AP at i emitting on the channel c;
- $(R_i)_{i\in V_{cand}} \in \{0,1\}^{V_{cand}}$: indicates the presence of a repeater at i;
- $(R_i^c)_{i\in V_{cand}, c\in\mathcal{C}} \in \{0,1\}^{V_{cand}\times\mathcal{C}}$: indicates the presence of a repeater at i emitting on the channel c;
- $(f_{i,j})_{(i\neq j)\in(V^*)^2} \in R_+^{(V^*)^2}$: the packet flow from i to j.
- $(f^c_{i,j})_{(i\neq j,c)\in V^2\times\mathcal{C}} \in R_+^{V^2\times\mathcal{C}}$: the packet flow from i to j on channel c.

Objective: minimize the installation cost: $\sum_{i\in V_{cand}} A_i C_i^A + R_i C_i^R$

Constraints:

- Link capacity: $\forall(i\neq j) \in (V^*)^2, f_{i,j} \leq w_{i,j}$;
- Kirchhoff law : $\forall i \in V, \sum_{j\in V^*\setminus\{i\}} f_{i,j} - f_{j,i} = D_i$
- AP or repeater: $\forall i \in V_{cand}, A_i + R_i \leq 1$;
- Only APs can be directly wired to the root: $\forall i \in V_{cand}, f_{i,r} \leq MA_i$;
- Machine capacity: $\forall j \in V_{cand}, \sum_{i\in V} f_{i,j} \leq A_j\gamma^A + R_j\gamma^R$;
- Unique emission channel: $\forall i \in V_{cand}, A_i = \sum_{c\in\mathcal{C}} A_i^c, R_i = \sum_{c\in\mathcal{C}} R_i^c$;
- Flow decomposition : $\forall(i\neq j) \in V^2, f_{i,j} = \sum_{c\in\mathcal{C}} f^c_{i,j}$
- Channel selection between client and candidate: $\forall(i,j,c) \in V_{clients} \times V_{cand} \times \mathcal{C}, f^c_{i,j} \leq w_{i,j}(A_j^c + R_j^c)$;
- Channel selection between candidates: $\forall(i\neq j,c) \in V_{cand} \times V_{cand} \times \mathcal{C}, f^c_{i,j} \leq w_{i,j}(A_i^c + R_i^c)$
- Noise constraint between a client and a candidate: $\forall(i,j,c) \in V_{clients} \times V_{cand} \times \mathcal{C}$:

$$\sum_{k\in V_{cand}\setminus\{j\}} p_{k,i}(A_k^c + R_k^c) \leq (p^i_{max} + n_{max})(1 - \frac{f^c_{i,j}}{w_{i,j}}) + n_{max}\frac{f^c_{i,j}}{w_{i,j}}$$

- Noise constraints between candidates: $\forall(i,j,c) \in V_{cand} \times V_{cand} \times \mathcal{C}$:

$$\sum_{k\in V_{cand}\setminus\{i,j\}} p_{k,i}(A_k^c + R_k^c) \leq (p^i_{max} + n_{max})(1 - \frac{f^c_{j,i}}{w_{j,i}}) + n_{max}\frac{f^c_{j,i}}{w_{j,i}}$$

4 Solution and perspectives

Simulation's performance TAB.1 below contains computation times obtained with a processor Intel(R) Xeon(R) CPU E3-1271 v3 @ 3.60GHz for several simulations. The length and the width of the building correspond to a discretization step of 3cm, equivalent to the quarter of the wavelength for the WiFi 2.4GHz standard.

Length	Width	N_x	N_y	Factorisation time	Resolution time (one source)
18m	12m	600	400	5s	0.07s
30m	10,5m	1000	350	8s	0.15s
30m	24m	1000	800	46s	0.4s
60m	30m	2000	1000	141s	1.2s

TAB. 1: Building dimensions, grid dimensions and computation times

First attempt to solve the problem For this first attempt we limit the analysis to a deterministic demand : the vector D is constant and thus we get a classic MILP formulation. We encoded it with AMPL and solved it for different instances using CPLEX solver. Below are the computation times obtained with a processor Intel(R) Xeon(R) CPU E3-1271 v3 @ 3.60GHz for several instance sizes :

$\|V_{clients}\|$	$\|V_{cand}\|$	Computation time
2	6	1s
5	10	6s
10	20	40s
20	60	4500s

TAB. 2: Number of clients and candidates, computation time

Perspectives Our current implementation considers a deterministic WiFi demand on the building, yet a statistical approach would give a more relevant deployment for real instances. In that case, we would need to choose between robust or stochastic optimization.

5 Figures

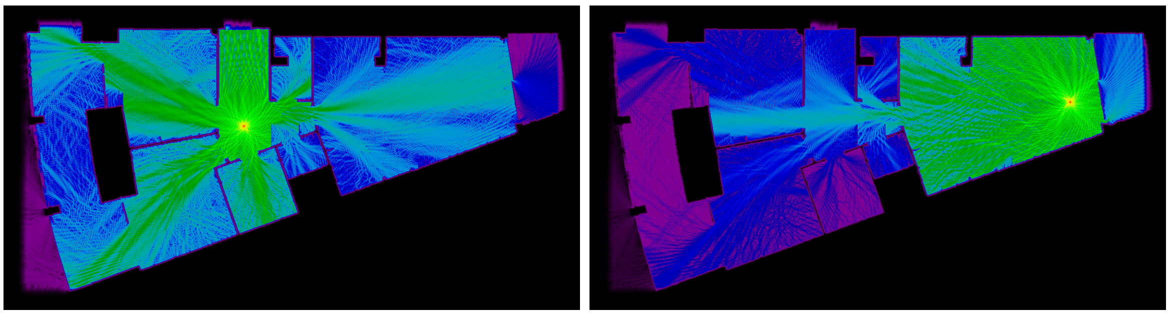


FIG. 1: 2.4GHz WiFi field in a building with source at different positions (red dot).

References

[1] B. Chopard, P. O. Luthi, and J. F. Wagen. Lattice boltzmann method for wave propagation in urban microcells. *IEE Proceedings - Microwaves, Antennas and Propagation*, 144(4):251–255, Aug 1997.

[2] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[3] Jean-Marie Gorce, Katia Jaffrès-Runser, and Guillaume De La Roche. The Adaptive Multi-Resolution Frequency-Domain ParFlow (MR-FDPF) Method for Indoor Radio Wave Propagation Simulation. Part I : Theory and Algorithms. Technical Report RR-5740, INRIA, November 2005.

[4] Guillaume De La Roche. *Simulation de la propagation des ondes radio en environnement multi-trajets pour l'etude des reseaux sans fil.* PhD thesis, INSA Lyon, 2007.

Influence Maximization in Social Networks under Deterministic Linear Threshold Model

Furkan Gursoy[1], Dilek Gunnec[2]

[1] Dept. of Management Information Systems, Bogazici University, 34342, Istanbul, Turkey
furkan.gursoy@boun.edu.tr

[2] Dept. of Industrial Engineering, Ozyegin University, 34794, Istanbul, Turkey
dilek.gunnec@ozyegin.edu.tr

Abstract

We define the new Targeted and Budgeted Influence Maximization under Deterministic Linear Threshold Model problem by extending the original influence maximization problem to a targeted version where nodes might carry heterogeneous profit values, and to a budgeted version where nodes might carry heterogeneous costs for becoming seed nodes. As a solution to this problem, we develop a novel and scalable general algorithm which utilizes a set of alternative methods for different operations: TArgeted and BUdgeted Potential Greedy (TABU-PG) algorithm.

TABU-PG works in an iterative and greedy fashion where nodes are compared at each iteration and the best one(s) are chosen as seed. The main idea behind TABU-PG is to invest in potential future gains which are hoped to be materialized at later iterations. Alternative methods are provided for calculating potential gain, and for comparing nodes. In comparing nodes, we propose a hybrid model which considers both gain and efficiency. In calculating potential gains, we propose methods which dynamically assign suitable weights to potential gains based on remaining budget. We also propose a new method which ignores the potential gains which are results of partial influences under a parameterized ratio. Moreover, we equip TABU-PG with novel scalability methods which reduces runtime by limiting the seed node candidate pool, or by selecting more nodes at each iteration; trading-off between runtime and spread performance. In addition, we suggest new data generation methods for influence weights on links; and threshold, profit, and cost values for nodes which better mimics the real world dynamics.

Extensive computational experiments with 8 different dataset on 4 real-life networks (Epinions, Acedemia, Pokec, and Inploid) show that TABU-PG heuristics perform significantly better than benchmark heuristics. Moreover, runtime can be reduced with very limited reduction in final influence spread.

Keywords : *Influence Maximization, Social Networks, Diffusion Models, Targeted Marketing, Greedy Algorithm.*

From Proofs to Programs, Graphs and Dynamics. Geometric perspectives on computational complexity

Thomas Seiller

Laboratoire d'Informatique de Paris Nord,
Université de Paris 13 and Sorbonne Paris Cité
CNRS (UMR 7538), 93430 Villetaneuse, France
`seiller@lipn.fr`

Abstract

The current state of the art in the field of complexity theory is a demonstrated lack of proof methods against problems still open. The combination of three separate results, called barriers (Relativisation, Natural Proofs and Algebrization), implies that none of the currently known proof methods for separation will successfully settle the remaining open problems. A single research program – Geometric Complexity Theory (GCT) – is considered viable by the community. However, according to its initiator and major contributor K. Mulmuley, GCT will not provide new results within our lifetimes; recent results have moreover closed the easiest path to GCT. As a consequence, complexity theory is in dire need of new tools and methods as such advances should require "fundamentally new methods" to paraphrase S. Aaronson and A. Widgerson. This talk will be about how such methods may be founded upon some recent developments in logic, and more precisely some specific models of proofs introduced under the name "Interaction Graphs".

The interplay between logic and computational complexity has been the subject of research for more than 50 years, but it has arguably failed to provide insights on the classification problem. Nevertheless, it has shown how logic is tightly bound to computation, clearly circumscribing the limits of the different approaches. The framework of Interaction Graphs, although taking its roots in logic, offers a mathematical model of programs that bypasses these limits and accounts for subtle aspects of computation. Moreover, it unveils deep connections with methods from geometry and dynamical systems that one may hope to exploit to enable potent proof methods from mathematics to be used by researchers against open problems in complexity theory.

On some tractable constraints on paths in graphs and in proofs

Lê Thành Dũng Nguyễn[1][2]

[1] École normale supérieure, Paris Sciences et Lettres, Paris, France
le.thanh.dung.nguyen@ens.fr

[2] LIPN, UMR 7030 CNRS, Université Paris 13, Sorbonne Paris Cité, Villetaneuse, France

Abstract

We show that trails avoiding forbidden transitions and rainbow paths for complete multipartite color classes can be found in linear time, whereas finding rainbow paths is NP-complete for any other restriction on color classes. For the tractable cases, we also state new structural properties equivalent to Kotzig's theorem on bridges in unique perfect matchings. Finally, we mention some connections with proof nets in linear logic and combinatorial proofs ("proofs without syntax") for classical propositional logic.

Keywords : *Perfect matchings, forbidden transitions, properly colored paths, rainbow paths.*

1 Introduction

Many problems which consist of finding a path or trail[1] under some constraints between two given vertices are equivalent to the *augmenting path* problem for matchings, and thus tractable. Some of these problems have associated "structure from acyclicity" theorems which were shown [13] to be equivalent to Kotzig's theorem on the existence of bridges in unique[2] perfect matchings (cf. [13, Theorem 1]): the absence of constrained cycles or closed trails entails the positive existence of some structure in the graph.

Our results here consist of finding new members of this family of constraints on paths which are equivalent in a certain sense, and excluding other constraints through NP-hardness results. We also bring to attention the fact that this family has a representative in proof theory.

Edge-colored graphs From an assignment of colors to the edges of a graph, one can define either *local* or *global* constraints:

- In a *properly colored* (PC) path (see [2, Chapter 16]) or trail (see [1]), *consecutive* edges must have different colors. Both can be found in linear time by reduction to augmenting paths, and conversely augmenting paths are a special case of both these problems. The structural result for PC cycles is Yeo's theorem on cut vertices separating colors [2, §16.3].
- In a *rainbow* (also called *heterochromatic* or *multicolored*) path, *all* edges have different colors. The subject of *rainbow connectivity* has been an active area of research recently, but the problem is NP-complete [4] in the general case.

For rainbow paths, we investigate whether restrictions on the shape of the *color classes* – that is, the subgraphs induced by all edges of a given color – make the problem tractable, and we establish that there is a single case which is not NP-hard:

[1]Following a common usage (see e.g. [2, Section 1.4]), a *path* is a walk without repeating *vertices* and a *trail* is a walk without repeating *edges*; a *cycle* (resp. *closed trail*) is a closed walk without repeating vertices (resp. edges). Paths (resp. cycles) are trails (resp. closed trails), but the converse does not always hold.

[2]This is indeed an acyclicity condition: recall that a perfect matching is unique if and only if it admits no alternating cycle.

Theorem 1. *Let $\mathcal{A}$ be a class of graphs without isolated vertices*[3]*. The rainbow path problem for graphs whose color classes are all in $\mathcal{A}$ can be solved in* linear time *if all graphs in $\mathcal{A}$ are* complete multipartite*, and is* NP-complete *otherwise.*

The first case is part of our family of equivalent constraints, and the associated structural theorem is as follows:

Theorem 2. *Let G be an edge-colored graph whose color classes are complete multipartite. If G has no rainbow cycle, then there exists a color c such that for all c-colored edges (u, v), u and v are in different connected components after removing the color class of c.*

Forbidden transitions A very general notion of *local* constraints is to simply forbid some pairs of edges from occuring consecutively in a path. We take the following definition from [12].

Definition 1. Let $G = (V, E)$ be a multigraph. A *transition graph* for a vertex $v \in V$ is a graph whose vertices are the edges incident to v. A *transition system* on G is a family $T = (T(v))_{v \in V}$ of transition graphs.

A path (resp. trail) $v_1, e_1, v_2 \ldots, e_{k-1}, v_k$ is said to be *compatible* (or *avoiding forbidden transitions*) if for $i = 1, \ldots, k-1$[4], e_i and e_{i+1} are adjacent in $T(v_{i+1})$.

That is, the edges of the transition graphs specify the *allowed* transitions. Finding a compatible *path* has been proven to be NP-complete [12]. However, the question for compatible *trails* does not seem to have been asked before in its full generality. We show that:

Theorem 3. *Finding a compatible trail can be done with a time complexity* linear *in the number of* allowed *transitions (thus, in at most quadratic time in the size of the graph).*

Theorem 4 ("Structure from acyclicity")**.** *Let G be a multigraph with transition system T, with at least one edge. If, for all vertices v in G, the transition graph $T(v)$ is* connected*, and G has no closed trail compatible with T, then G has a bridge.*

Corollary 1 (New[5] proof of [1, Theorem 2.4])**.** *Let G be an edge-colored graph such that every vertex of G is incident with at least two differently colored edges. Then, if G does not have a PC closed trail, then G has a bridge.*

2 The edge-colored line graph

A key ingredient in the aforementioned results is a kind of *line graph* construction mapping graphs with forbidden transitions to edge-colored graphs.

Definition 2. Let $G = (V, E)$ be a multigraph and T be a transition system on G. The *EC-line graph* $L_{EC}(G, T)$ is formed by taking the line graph of G, coloring its edges so that the clique corresponding to v is given the color v (using the vertices of G as the set of colors), and deleting the edges corresponding to forbidden transitions.

Formally, $L_{EC}(G, T)$ is defined as the graph with vertex set E and edge set $F = \bigsqcup_{v \in V} T(v)$, equipped with an edge coloring $c : F \to V$ with values in V: for $f \in F$, $c(f)$ is the unique vertex such that $f \in T(c(f))$.

Proposition 1. *Let G be a multigraph with transition system T, and $s \neq t$ be vertices of G.*

The compatible paths *between s and t correspond bijectively to* rainbow paths *in $L_{EC}(G, T)$ between some vertex of $\partial(s)$ and some vertex of $\partial(t)$ which do not cross edges with color s or t.*

Similarly, the compatible trails *between s and t where neither s nor t appear as intermediate vertices correspond bijectively to* PC paths *in $L_{EC}(G, T)$ between some vertex of $\partial(s)$ and some vertex of $\partial(t)$ which do not cross any edge with color s or t.*

[3]Indeed, a color class, which is an edge-induced graph, cannot have isolated vertices.

[4]For a cycle (resp. closed trail), we must also require e_{k-1} and e_1 to be adjacent in $T(v_1) = T(v_k)$.

[5]The original proof applies Yeo's theorem to a construction which does not generalize to forbidden transitions, but provides a trail-finding algorithm in linear time *in the size of the graph.*

Theorems 3 and 4 immediately follow from the second half of this proposition together with the known results on PC paths. However, to get the hardness result for rainbow paths, in addition to the EC-line graph, we need to reuse the proof techniques from [12] and [4], in particular a characterization of complete multipartite graphs by excluded vertex-induced subgraphs [12, Lemma 7]. As for the first half of Theorem 1, it uses the fact that one can retrieve the vertex partition of a complete multipartite graph in linear time, for instance by computing its cotree [5].

3 Constrained cycles in logic

In a recent work [9], we showed that the *correctness* of a *proof net* – a graph-like representation of a proof in *linear logic* [6] – is equivalent to the uniqueness of a given perfect matching, and is therefore part of our family of equivalent problems. Thus, it can be decided in linear time, and the associated structural property is the key lemma in the proof of the "sequentialization theorem", an inductive characterization of the set of correct proof nets which mirrors exactly the inference rules of linear logic.

One direction of the equivalence, from proof nets to perfect matchings, had been established previously by Retoré [11, §1][6]. His reduction can be understood *a posteriori* as a composition of constructions on edge-colored graphs: it amounts to equipping a proof net with a transition system, taking the EC-line graph introduced above, and applying a known reduction from edge-colored graphs with chromatic degree ≤ 2 to perfect matchings [8][7].

Let us give a rough presentation of proof nets in graph-theoretic terms. A proof net may be seen as the syntax tree of a propositional formula, with $\wedge$ and $\vee$ nodes and literals at the leaves, together with additional edges between the leaves pairing together opposite literals. The syntax tree may be interpreted as the *cotree* of a *cograph* whose vertices are the literals, as usual, see e.g. [3]. This leads to a restatement of correctness, also due to Retoré [11, §2].

Definition 3. A *cographic proof* is an pair of graphs (G, M), G being a cograph and M a 1-regular graph, with the same set of vertices.

A *vicious circle* in (G, M) is a *chordless* cycle in[8] $G \cup M$ which alternates between edges in G and edges in M. A cographic proof is *correct* if it contains no vicious circle.

A proof net is correct if and only if the corresponding cographic proof (with the 1-regular graph representing the pairing of the leaves) is correct in the sense above. Note that vicious circles are not merely properly colored cycles for the natural 2-edge-coloring of the cographic proof, because of the additional chordlessness condition.

Finally, let us mention that cographic proofs also have applications outside of linear logic. Indeed, they have been used to define "proofs without syntax" for classical propositional logic: Hughes's *combinatorial proofs* [7] are graph homomorphisms (with additional properties) from some correct cographic proof to the cograph of the classical formula being proven, and this gives a sound and complete proof system. The tractability of our family of constraints on cycles ensures that proofs are checkable in polynomial time.

[6]This was the first indication of a connection between linear logic and unique perfect matchings. Let us mention as well that in an earlier attempt to connect linear logic with graph theory [10, Chapter 2], Retoré proved a weaker version of the structural theorem for rainbow acyclic graphs (it requires the color classes to be complete *bipartite* instead of complete multipartite).

[7]This paper only defines the reduction for 2-edge-colored graphs, but the required generalization is straightforward. Note also that the two last steps give a direct reduction from compatible trails to perfect matchings. Although we have not managed to find it in the literature, there is at least one other place where it occurs implicitly, which also inspired us: a solution to an algorithmic puzzle by Christoph Dürr, see `http://tryalgo.org/en/matching/2016/07/16/mirror-maze/`.

[8]By $G \cup M$ we mean the graph whose edges are the union of those in G and M, on the common vertex set. This union may result in a multigraph with parallel edges.

4 Conclusion and perspectives

We summarize the complexity of the problems studied here in the following table. Our contributions, marked in bold, fill some gaps in the table, thus answering several natural questions. Furthermore, we exhibited a construction which provides a bridge between different kinds of constraints on paths and trails, and described how different reductions relate to each other.

	Time complexity / additional results
Path avoiding forbidden transitions	NP-complete with dichotomy result [12]
Trail avoiding forbidden transitions	**Linear with structural theorem**
Properly colored path	Linear with structural theorem (cf. [2])
Properly colored trail	Linear with structural theorem [1]
Rainbow path/trail[9] (general)	NP-complete [4], **with dichotomy result**
Rainbow path/trail (restricted[10])	**Linear with structural theorem**

To clarify, the connection with proof nets works specifically for a system called Multiplicative Linear Logic with the Mix rule. Without this Mix rule, correctness becomes a “tree-like” condition instead of an acyclicity (“forest-like”) condition.

What analogous conditions could one ask of a constrained graph? In the case of rainbow paths and cycles, we may consider edge-colored graphs whose maximum rainbow subgraphs are all trees. Remarkably, it seems that we have a polynomial-time recognition algorithm and a structural property for these graphs without any restriction on the shape of color classes[11].

References

[1] A. Abouelaoualim, K. Ch. Das, L. Faria, Y. Manoussakis, C. Martinhon, and R. Saad. Paths and trails in edge-colored graphs. *Theoretical Computer Science*, 409(3):497–510, December 2008.

[2] Jørgen Bang-Jensen and Gregory Gutin. *Digraphs. Theory, algorithms and applications.* 2nd ed.

[3] Seth Chaiken, Neil V. Murray, and Erik Rosenthal. An application of P_4-free graphs in theorem-proving. *Annals of the New York Academy of Sciences*, 555(1):106–121, May 1989.

[4] Sourav Chakraborty, Eldar Fischer, Arie Matsliah, and Raphael Yuster. Hardness and algorithms for rainbow connection. *Journal of Combinatorial Optimization*, 21(3):330–347, April 2011.

[5] Derek G. Corneil, Yehoshua Perl, Lorna K. Stewart. A linear recognition algorithm for cographs. *SIAM Journal on Computing*, 14(4):926–934, 1985.

[6] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–101, January 1987.

[7] Dominic J.D. Hughes. Proofs without syntax. *Annals of Mathematics*, 143(3):1065–1076, 2006.

[8] Yannis Manoussakis. Alternating paths in edge-colored complete graphs. *Discrete Applied Mathematics*, 56(2):297–309, January 1995.

[9] Lê Thành Dũng Nguyễn. Unique perfect matchings and proof nets. Submitted. URL: `https://hal.archives-ouvertes.fr/hal-01692179`.

[10] Christian Retoré. *Réseaux et séquents ordonnés.* PhD thesis, Université Paris VII, February 1993.

[11] Christian Retoré. Handsome proof-nets: perfect matchings and cographs. *Theoretical Computer Science*, 294(3):473–488, February 2003.

[12] Stefan Szeider. Finding paths in graphs avoiding forbidden transitions. *Discrete Applied Mathematics*, 126(2-3):261–273, 2003.

[13] Stefan Szeider. On theorems equivalent with Kotzig’s result on graphs with unique 1-factors. *Ars Combinatoria*, 73, 2004.

[9] The existence of a rainbow path is equivalent to the existence of a rainbow trail between two vertices.

[10] Restricted to edge-colored graphs with complete multipartite color classes.

[11] The trick is that any such graph is a spanning subgraph of another with complete bipartite color classes.

Equitable total chromatic number of two classes of complete r-partite p-balanced graphs

A. G. da Silva[1], D. Sasaki[2], S. Dantas[3]

[1] Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil
andersongs@hotmail.com.br

[2] IME, State University of Rio de Janeiro, Rio de Janeiro, Brazil
diana.sasaki@ime.uerj.br

[3] IME, Fluminense Federal University, Niteroi, Brazil
sdantas@im.uff.br

Abstract

An equitable total coloring of a graph is the assignment of colors to the vertices and edges of a graph subject to the following conditions: adjacent vertices or edges must receive different colors; an edge and a vertex that is incident to it have to be assigned to different colors; and the difference between the cardinalities of any two color classes is at most one. The least integer for which a graph has an equitable total coloring is called the equitable total chromatic number of the graph and denoted by χ''_e. It has been conjectured by Wang (2002) that $\Delta + 1 \leq \chi''_e \leq \Delta + 2$. Such conjecture is known as the Equitable Total Coloring Conjecture (ETCC). Fu (1994) determined χ''_e for the bipartite p-balanced graphs. Silva, Dantas and Sasaki (2016) verified the ETCC for four classes of complete r-partite p-balanced graphs, which are: r and p odd ($\chi''_e = \Delta + 1$); $r \geq 4$ even and p odd ($\chi''_e = \Delta + 2$); $r \geq 4$ even and p even ($\chi''_e \leq \Delta + 2$); and r odd and p even ($\chi''_e \leq \Delta + 2$). In this paper, we present new techniques and prove that complete r-partite p-balanced graphs (for $r \geq 4$ even and p even, and for r odd and p even) have $\chi''_e = \Delta + 1$, which concludes the study for all cases of equitable total coloring of complete r-partite p-balanced graphs, showing sharp values for the equitable total chromatic number.

Keywords : *Equitable total coloring, complete r-partite p-balanced graphs, graph coloring.*

1 Introduction

Througuout this paper all graphs analyzed are finite, undirected and simple. The *equitable total chromatic number* of a graph G, denoted by $\chi''_e(G)$ is the least integer for which G has an equitable total coloring. An *equitable total coloring*, in turn, is the assignment of colors to the vertices and edges of a graph such that incident and adjacent elements do not receive the same color and the difference between the cardinalities of any two color classes is at most 1.

A *complete r-partite p-balanced graph*, denoted by $K_{r\times p}$ is a graph where the vertex set can be partitioned into r parts ($V(K_{r\times p}) = \{X_1, \cdots, X_r\}$) so that two vertices within the same part are not adjacent and there is an edge between any two vertices of different parts. The total chromatic number of all complete r-partite p-balanced graphs was determined by Bermond [2]. Wang [6] conjectured that the equitable total chromatic number of a graph is either $\Delta + 1$ or $\Delta + 2$. Fu [3] determined that the equitable total chromatic number of complete bipartite graphs is $\Delta + 2$, whereas Silva, Dantas and Sasaki [5] determined the equitable total chromatic number for two classes of complete r-partite p-balanced graphs, which are the cases $r \geq 4$ even and p odd ($\chi''_e = \Delta + 2$); and r and p odd ($\chi''_e = \Delta + 1$). In this paper, we prove that $K_{r\times p}$ has $\chi''_e = \Delta + 1$ if $r \geq 4$ and p are even and if r is odd and p is even, concluding all cases of complete r-partite p-balanced graphs.

2 $K_{r\times p}$, p even

We adopt the following convention regarding the complete r-partite p balanced graphs, denoted by $K_{r\times p}$: when we refer to a graph of this kind, keep in mind that the display of the vertices is like a matrix with r columns and p rows, where each column represents a part X_i of the partition of the vertex set. The vertex x_{ij} is the j-th vertex of the part X_i and will be in the j-th row and i-th column. Based on this display of vertices, we define a *horizontal edge* as an edge of the kind $x_{ab}x_{cb}$. Also, a *horizontal matching of distance* i between rows a and b ($1 \leq a < b \leq p$) is defined as the matching $\{x_{ja}x_{j+i,b} | 1 \leq j \leq r\}$, where the index $j+i$ is taken modulo r.

Throughout the paper, it will be necessary to obtain matchings of the complete graph K_i. We refer to the matchings of K_i as P_i when we need to take the matchings of the complete graph with p vertices and R_i, when the complete graph has r vertices (r and p being related to the fact that we are coloring r-partite p-balanced graphs).

$K_{r\times p}$, with $r \geq 4$ even and p even

A *Latin square* is an $n \times n$ matrix whose entries are the elements of the set $\{1, 2, \cdots, n\}$ such that each symbol occurs precisely once per row and per column. Given a Latin square of order n, a *transversals* is a set of n different entries of different rows and columns. In [4] it is proved the following theorem: defining $T(n)$ as the maximum number of transversals over all Latin squares of order n, we have that $b^k \leq T(k)$ for $k \geq 5$, where $b \approx 1,719$. We present now a lemma and omit its proof, because it is trivial.

Lemma 1. *There exists a Latin square of even order n ≥ 4 whose elements in the main diagonal are pairwise different.*

Sketch of the algorithm for the case $p = 2$: we define a *coloring matrix* as a matrix whose entries determine the colors assigned to the elements of a graph. Let $A_{R_1R_2}$ be a matrix of order r in which the entry a_{ij} represents the color that the edge $x_{i1}x_{j2}$ receives if $i \neq j$ and the color that the vertices of the part X_i receive, otherwise. The matrix $A_{L_1L_2}$ must be a Latin square whose elements in the main diagonal are all distinct. Lemma 1 ensures the existence of such matrix. Each one of these r colors is used $r+1$ times.

We have that $r-1$ colors still need to be used. They will be applied in horizontal edges as follows: obtain the $r-1$ matchings of K_r. Suppose that $R_i = \{v_{a_1}v_{a_2}, \cdots, v_{a_{r-1}}v_{a_r}\}$. Then the edges $x_{a_11}x_{a_21}, \cdots, x_{a_{r-1}1}x_{a_r1}, x_{a_12}x_{a_22}, \cdots, x_{a_{r-1}2}x_{a_r2}$ must receive the same color, for each $i = 1, 2, \cdots, r-1$. By construction we get that the algorithm describes an equitable total coloring of $K_{r\times 2}$ ($r \geq 4$ even).
Sketch of the algorithm for the case $p = 4$: to color the vertices we use a different color for each one of the following pairs: x_{11} and x_{12}; x_{13} and x_{14}; x_{21} and x_{22}; x_{23} and x_{24}; $\cdots$; $x_{(r-1)1}$ and $x_{(r-1)2}$; $x_{(r-1)3}$ and $x_{(r-1)4}$; x_{r1} and x_{r4}; x_{r2} and x_{r2}. The part X_r is the only one that has a different pattern for the coloring of the vertices.

The colors used in the vertices of the rows 1 and 2 of the parts $X_1, X_2, \cdots, X_{r-1}$ will be applied on rows 3 and 4 in horizontal edges according to the matchings of K_r, whereas the colors used in the vertices of rows 3 and 4 of the same parts will be assigned to horizontal edges of rows 1 and 2 according to the matchings of K_r. If $R_1 = \{v_{b_1}v_{b_2}, \cdots, v_{b_{r-1}}v_{b_r}\}$, then assign the color of the vertices x_{11} and x_{12} to the edges $x_{b_1i}x_{b_2i}, \cdots, x_{b_{r-1}i}x_{b_ri}$ ($i = 3, 4$) and assign the color of the vertices x_{13} and x_{14} to the edges $x_{b_1i}x_{b_2i}, \cdots, x_{b_{r-1}i}x_{b_ri}$ ($i = 1, 2$). Proceed analogously regarding the colors of the vertices of the parts $X_2, X_3, \cdots, X_{r-1}$.

We define matrices $A_{R_1R_2}$, $A_{R_3R_4}$, $A_{R_2R_3}$, $A_{R_1R_4}$, $A_{R_1R_3}$ and $A_{R_2R_4}$, of order r, where the entry a_{ij} of the matrix $A_{R_kR_l}$ represents the color that the edge $x_{ik}x_{jl}$ receives if $i \neq j$. We leave the entry empty if $i = j$.

We have to use $r-1$ colors in matrices $A_{R_1R_3}$ and $A_{R_2R_4}$, according to the following pattern: on the first row, we apply the colors in ascending order, that is $(1 \quad 2 \quad 3 \quad \cdots \quad r)$; on the second row, we shift the first row one unity to the right, that is, $(r \quad 1 \quad \cdots \quad (r-1))$. After that, we omit the entries of the main diagonal.

To fill the matrices $A_{R_1R_2}$ and $A_{R_3R_4}$ we need Latin squares with the elements in the main diagonal being pairwise different (even though such entries will be ommited). The entries of the matrix $A_{R_1R_2}$ will be the colors used in the vertices of rows 1 and 2 of parts $X_1, X_2, \cdots, X_{r-1}$, whereas the entries of the matrix $A_{R_3R_4}$ are the colors used in the vertices of rows 3 and 4 of the same parts. Some entries stay empty at this point. Obtain a Latin square whose colors are described above and whose entries of the main diagonal are all distinct. The empty entries are the ones that would be the same color of the entry a_{rr}.

Suppose that colors colors α and β have been applied, respectively, in the vertices x_{r1} and x_{r4}; and in the vertices x_{r2} and x_{r3}. Then, in the matrix $A_{R_1R_4}$, color α must occupy entries $a_{1,r-1}, a_{21}, a_{32}, a_{43}, \cdots, a_{r-1,r-2}$, whereas color β must be occupy entries $a_{12}, a_{23}, a_{34}, \cdots, a_{r-1,r}$, a_{r1}. In matrix $A_{R_2R_3}$, the corresponding entries occupied with α in the other matrix must be filled with β and vice versa. It is easy to see that we can apply $r-4$ colors in horizontal matchings of distance linking vertices of rows 1 and 4; and rows 2 and 3.

There are entries in the matrices $A_{R_1R_2}, A_{R_3R_4}, A_{R_2R_3}$ and $A_{R_1R_4}$ not filled with any color. Such entries represent edges that form a 2-regular subgraph H of $K_{r\times p}$. Since H is 2-regular, its connected components are cycles. It can be easily seen that none of the connected components is a cycle odd size. Since the components of H are cycles of even size, these edges can be colored with 2 colors and this finishes the algorithm.

Sketch of the algorithm for the case $p \geq 6$: to color the vertices, we need to obtain the matching P_1 of the graph K_p. If $P_1 = \{v_{b_1}v_{b_2}, v_{b_3}v_{b_4}, \cdots, v_{b_{p-1}}v_{b_p}\}$, then assign a different color to each one of the following pairs of vertices: x_{ib_1} and x_{ib_2}; x_{ib_3} and x_{ib_4}; $x_{ib_{p-1}}$ and x_{ib_p} for all $i = 1, 2, \cdots, r$. Consider the matrices $A_{R_{b_1}R_{b_2}}, A_{R_{b_3}R_{b_4}}, \cdots, A_{R_{b_{p-1}}R_{b_p}}$ as described in the beginning of this section. We use Lemma 1 to get Latin squares whose entries in the main diagonal are pairwise distinct. Furthermore, the entry a_{kk} of a matrix $A_{R_iR_j}$ must be the colors of the vertices x_{ki} and x_{kj}. The entries of the matrices $A_{R_{b_i}R_{b_j}}$ are the colors used in the vertices of rows b_i and b_j.

Colors $1, 2, \cdots, \dfrac{rp}{2}$ were represented in all vertices of two rows. However, they still need to be represented in the vertices of the other rows. We need the following result to do so: [1] for positive even integers m and n with $4 \leq m \leq n$, the graph $K_n - I$ can be decomposed into cycles of size m if and only if the number of edges in $K_n - I$ is a multiple of m. We remark that $K_n - I$ denotes a complete graph with n vertices minus a 1-factor, that is, minus a perfect matching.

For the next step of the algorithm we need to get $\dfrac{p}{2}$ cycles of size $p-2$ of the graph K_p minus a 1-factor. Putting $m = p-2$ and $n = p$ in the theorem proved in [1], we conclude that $K_p - I$ can be decomposed into $\dfrac{p}{2}$ cycles of size $p-2$, as desired. Suppose that $K_p - I = K_p \backslash P_1$, with P_1 being a perfect matching of K_p. It is known that every cycle of even size has an edge coloring with 2 colors. Hence we divide each cycle of even order in two matchings and associate with the edges of P_1, so that each edge of P_1 is associated to the matchings of the cycle of $K_p - I$ that does not contain the vertices v_i and v_j.

With the decomposition of $K_p \backslash P_1$, we get $\dfrac{p}{2}$ cycles. Let M_k and M'_k be the matchings obtained from the k-th cycle of the decomposition of $K_s \backslash P_1$, that does not contain the edge v_iv_j. Then, the colors used in the i-th and j-th vertices of the parts $X_1, X_2, \cdots, X_{\frac{r}{2}}$ must be used in horizontal matchings of distance linking vertices of rows determined by M_k, whereas the colors used in the i-th and j-th vertices of parts $X_{\frac{r}{2}+1}, \cdots, X_r$ are used in horizontal matchings of distance linking vertices of rows determined by M'_k. We have that $\left(\dfrac{r}{2} - 1\right)(p-2)$ colors can still be applied in these available matchings of distance. To end the coloring, we use $r-1$ new colors in horizontal edges determined by the matchings of K_r.

$K_{r\times p}$, with r odd and p even

Sketch of the algorithm for the case $p = 2$: the vertices of part X_i must receive color i, $i = 1, 2, \cdots, r$. These r colors are also used in horizontal edges, as follows. Suppose that the matching $R_j = \{v_a v_b, \cdots, v_c v_d\}$ has v_i as its remaining vertex. Then color cor i has to be used in edges $x_{a1}x_{b1}, \cdots, x_{c1}x_{d1}, x_{a2}x_{b2}, \cdots, x_{c2}x_{d2}$. One can easily check that each one of the r colors were used $r + 1$ times. Now $r - 1$ colors need to be used in non horizontal edges. We use each one of the $r - 1$ colors in horizontal matchings of distance, which have r elements each. By construction the algorithm provides a $(\Delta + 1)$-equitable total coloring of $K_{r\times 2}$.
Sketch of the algorithm for the case $p \geq 4$: we construct a table with $(p - 1)(r - 1) = rp - p - r + 1$ rows where we repeat each matching P_i of K_s $r - 1$ times. Each row represents a different color and each time a matching P_i is repeated it represents a different horizontal matching of distance. Suppose that in the i-th row of the table we have distance j and matching $P_k = \{v_a v_b, \cdots, v_c v_d\}$. This means that color i has to be used in a horizontal matching of distance j between pairs of rows determined by the matching P_k, that is, between rows a and b; $\cdots$; c and d. The second step consists in changing part of what was done in the previous one. All colors that are in the same row as the matchings P_i $(i = 1, 3, 5, \cdots, p - 3)$ in the above described table and also each color that is in the same row as the first occurrence of P_j $(j = 2, 4, \cdots, p - 2)$ transfer the first element of their related matching to r new colors that will be inserted in the new table.

If a given color i had been applied in a horizontal matching of distance j and transfered the element v_{ab} of the matching P_k (k odd), then color i must be applied in the coloring of vertices x_{ja} and x_{jb}. If the index k of the matching P_k is even, then color i must be used to color vertices x_{ra} and x_{rb}. The last r colors (the ones that were added only in the second step) must color vertices $x_{t,p-1}$ and x_{tp} if the corresponding distance in the second table described is t and if it is not the last color. The last color is used in vertices x_{r1} and x_{rp}. Horizontal edges are colored using the matchings of K_r.

3 Conclusion and perspectives

In this paper we prove that $\chi''_e(K_{r\times p}) = \Delta + 1$ for r and p even $(r \geq 4)$; and for r odd and p even. This paper, alongside with [3, 5] conclude the work of determining the equitable total chromatic number for all cases of complete r-partite p-balanced graphs, verifying the ETCC for this class of graphs. Future work include, but are not limited to determining the equitable total chromatic number of complete r-partite non-balanced graphs.

References

[1] Brian Alspach and Heather Gavlas. *Cycle Decompositions of K_n and $K_n - I$.* Journal of Combinatorial Theory, 2001.

[2] J. C. Bermond. *Nombre chromatique total du graphe r-parti complet.* J. London Math Soc., 1974.

[3] Hung Lin Fu. *Some results on equilized total colorings.* Congr. Numer., 1994.

[4] Brendan D. McKay, Jeanette C. McLeod and Ian M. Wanless. *The number of transversals in a Latin square.* Des Codes Crypt, 2006.

[5] A. G. da Silva, S. Dantas and D. Sasaki. *Equitable total coloring of complete r-partite p-balanced graphs*, submitted. Discrete Applied Mathematics, 2016.

[6] W. F. Wang. *Equitable total coloring of graphs with maximum degree 3.* Graphs Combin., 2002.

Multicoloring of Pattern Graphs for Sparse Matrix Determination

Shahadat Hossain[1], Trond Steihaug[2]

[1] University of Lethbridge, Lethbridge, Alberta, Canada
shahadat.hossain@uleth.ca

[2] University of Bergen, Bergen, Norway
Trond.Steihaug@ii.uib.no

Abstract

Evaluation of large and sparse derivative matrices is an essential and significant computation in many numerical algorithms. The combinatorial problem of compressing the sparse Jacobian matrix is an important component in its efficient evaluation and a variety of compression methods (one-sided/two-sided) have been suggested that exploit problem structures such as symmetry and partial separability. A common approach is to define an appropriate graph for the sparse matrix and partition the vertices of the graph into groups or color classes. Recently, the pattern graph has been proposed as a unifying framework to model direct determination of sparse Jacobian and Hessian matrices. In this paper we give a multi-coloring formulation for the two-sided compression of sparse Jacobian and thus combine two closely related compression problems using the same graph. Moreover, we show that the essential computational complexity of the respective compression problems remain the same under column or row permutation of the underlying sparse matrix.

Keywords : *Sparse Hessian Matrix, Sparse Jacobian Matrix, Multicoloring, Direct Determination, Algorithmic Differentiation.*

1 Introduction

Combinatorial problems arising in diverse scientific and engineering areas are conveniently modelled and studied using graphs. Numerical methods for solving system of nonlinear equations, differential equations, or optimization of nonlinear functions often require the evaluation of first or higher-order derivatives, usually in each iteration. A significant fraction of the overall computation of such methods is attributed to the cost of evaluating these large and sparse derivative matrices. A common approach to determining the sparse Jacobian and Hessian matrices is to first represent the matrix pattern using a suitable graph and then employ a grouping or coloring procedure to find a *compressed* representation of the sparsity pattern. As the general grouping or coloring problems are computationally hard (NP-hard) heuristics are commonly employed for compression. For a sparse matrix A, in a one-sided compression, the rows of the sparsity pattern of A (or $A^\top$) are compressed and a two-sided compression corresponds to row compression of both A and $A^\top$. Specifically, let $F : \mathbb{R}^n \mapsto \mathbb{R}^m$ be a once continuously differentiable function in some neighborhood of $x \in \mathbb{R}^n$. Then we can write,

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

where $s \in \mathbb{R}^n$ is a given direction and $\epsilon > 0$ is a small increment. The key observation here is to choose fewest directions s exploiting the sparsity information such that the matrix nonzero unknowns can be determined by the algorithm sketched below.

1. *Seeding or Compression.* Given the seed matrix $S \in \mathbb{R}^{n\times p}$, compute $B(= AS)$ using a forward difference formula or tangent linear (or forward) algorithmic differentiation [4]. (In a two-sided compression also compute $C^\top = W^\top A$ using adjoint (or reverse) algorithmic differentiation for a given $W \in \mathbb{R}^{m\times q}$.)

2. *Harvesting or reconstruction.* For row $i, i = 1, 2, \ldots, m$ of A:

   (a) define the reduced seed matrix $S_i \in \mathbb{R}^{\rho_i\times p}$ for the ith row of A:

   $$\text{define: } S_i \equiv S(v, :)$$

   (b) solve for the ρ_i unknown elements $a_{ij} \neq 0$

   $$\text{solve: } A(i, v)S_i = B(i, :)$$

   where ρ_i denotes the number of nonzero elements in row i of A and vector v of length ρ_i contains the column indices of those nonzero elements. (This step is repeated for two-sided compression with matrices W and C.)

The goal is to choose matrices S (and W in a two-sided compression) minimizing p ($p + q$ in a two-sided compression) such that the nonzero elements of matrix A can be determined uniquely. In this paper the equations solved in step $2(b)$ are permuted diagonal.

The combinatorial problem of finding suitable seed matrices for sparsity pattern of matrix A has been found to be equivalent to grouping or coloring of the vertices of an appropriate graph representing the sparsity pattern. Some of the features that we consider desirable in choosing an appropriate graph model are: flexibility of the model to express alternative methods of determination (one-sided, two-sided compression), exploitation of problem structure (symmetry), and efficient implementation of graph operations. Graph models proposed by researchers include intersection graph [1], column segments graph [6], element isolation graph [10], bipartite graph [2, 3, 5] and more recently pattern graph [7, 9]. With regard to flexibility, unfortunately, intersection model requires the definition of a neutral color in a two-sided compression while the bipartite model leaves out one set of vertices uncolored in a one-sided compression. The pattern graph has been proposed as a unifying framework whereby the sparsity pattern of the underlying matrix is readily apparent while avoiding the aforementioned difficulties.

Computations on graphs, in general, lead to irregular data access and may result in significant performance degradation. Modern high-performance computing systems employ one or more levels of fast cache memory to amortize main memory access latency. The irregular access to data diminishes the benefit of hierarchical memory systems [11]. Some of the data access issues with regard to computer implementation of coloring heuristics have been addressed in [8]. In this paper, we unify the coloring formulation of the Hessian matrix determination that exploits pattern and value symmetry [9] and the Jacobian matrix determination in a two-sided compression [7]. Moreover, we address an outstanding issue with regard to the effect of reordering of rows or columns of the underlying sparse matrix on the associated pattern graph and show that reordering does not affect the essential complexity of the determination problem.

2 Multi Coloring

Given matrix $A, a_{ij'} \neq 0$ is a *lateral neighbor* of $a_{ij} \neq 0$ in A for indices $j, j' \in \{1, \ldots, n\},\ \ i \in \{1, \ldots, m\}$ such that the difference $j' - j$ is the smallest if $j' > j$ or such that the difference $j - j'$ is the smallest if $j > j'$ among all such indices j' in row i. A lateral neighbor of $a_{ij} \neq 0$ in $A^\top$ is its *vertical neighbor* in A. The *pattern graph* associated with matrix $A \in \mathbb{R}^{m\times n}$ is $G_{\mathcal{P}}(A) = (V, E)$, where

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

and

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

We say that there is a *path* of *length* $l \geq 1$ between vertices v_{ij} and v_{kl}, denoted $v_{ij} \equiv w_{i_0,j_0} - w_{i_1,j_1} - w_{i_2,j_2} - \cdots - w_{i_l,j_l} \equiv v_{kl}$ and abbreviated as $v_{ij} \sim v_{kl}$ if there exist distinct vertices $w_{i,j}, i, j = 0, 1, 2, \ldots, l-1$ such that $\{w_{i,j}, w_{i+1,j+1}\} \in E$.

We call element $a_{ij} \neq 0$ *unknown* if its value has not been determined; otherwise the element is *known*. For a symmetric matrix A if a_{ij} is known then so is a_{ji} such that any method for determining a symmetric matrix needs to determine only one of a_{ij} and a_{ji}. We call a determination method *symmetric direct determination (SDD)* if there is a seed matrix $S \in \{0,1\}^{n\times p}$ such that each unknown a_{ij} is determined *directly* i.e., there is an index k such that $a_{ij} = b_{ik}$ or $a_{ji} = b_{jk}$ in the matrix equation $AS = B$. For matrix A not symmetric we call a determination method *direct determination two-sided (DD2)* if there are seed matrices $S \in \{0,1\}^{n\times p}$ and $W \in \{0,1\}^{m\times q}$ such that $a_{ij} = b_{ik}$ in the matrix equation $AS = B$ or that $a_{ij} = c_{kj}$ in the matrix equation $W^\top A = C^\top$. A *SDD* in which number of column p of S is minimum is said to be *optimal SDD*. A *DD2* in which the sum of columns $p+q$ of S and W is minimum is said to be *optimal DD2*.

Let $\mathscr{P}(U)$ denote the set of all nonempty subsets of set U and denote by e_j the jth Cartesian basis vector. We first consider *SDD* of A. Let $\Phi_{SDD} : V \mapsto \mathscr{P}(\{1, 2, \ldots, p\})$ be a mapping such that the seed matrix S

$$S(:,k) = \sum_{\{j \mid k\in\Phi_{SDD}(v_{ij}),\ i=1,\ldots,n\}} e_j,\ k = 1, \ldots, p$$

yields direct determination of matrix A.

Let $\sigma_1, \ldots, \sigma_p, \omega_1, \ldots \omega_q$ be $p+q$ distinct numbers partitioned in two sets $\mathcal{S}$ and $\mathcal{W}$. Analogously, for $DD2$, let the mapping $\Phi_{DD2} : V \mapsto \mathscr{P}(\mathcal{S} \cup \mathcal{W})$, where $\mathcal{S} = \{\sigma_1, \ldots, \sigma_p\}, \mathcal{W} = \{\omega_1, \ldots \omega_q\}$ be such that the seed matrices S and W

$$S(:,k) = \sum_{\{j \mid \sigma_k\in\Phi_{DD2}(v_{ij}), i=1,\ldots,n\}} e_j \text{ and } W(:,l) = \sum_{\{i \mid \omega_l\in\Phi_{DD2}(v_{ij}), j=1,\ldots,m\}} e_i$$

yield direct determination of A.

The key observation here is that the two mappings defined above naturally imply colorings of vertices where each vertex is assigned a subset of colors. Then the direct determination of a nonzero unknown of the matrix can be expressed in terms of colors of the associated vertex and the colors of the other vertices that are reachable from it along some *specific* paths in the associated pattern graph. Consider a vertex v_{ij} in the pattern graph of matrix A. In the *SDD* method we have a $p-$*multi-coloring* [9] of the vertices of the pattern graph satisfying: (a) there is a color $c \in \Phi_{SDD}(v_{ij})$ such that $c \notin \Phi_{SDD}(v_{ij'}), j \neq j'$ or (b) there is a color $c \in \Phi_{SDD}(v_{ji})$ such that $c \notin \Phi_{SDD}(v_{ji'}), i \neq i'$. In the *DD2* method we have a $(p+q)-$*multi-coloring* of the vertices of the pattern graph satisfying: (a) there is a color $c \in \Phi_{DD2}(v_{ij})$ such that $c \notin \Phi_{DD2}(v_{ij'}), j \neq j'$ or (b) there is a color $c \in \Phi_{DD2}(v_{ij})$ such that $c \notin \Phi_{DD2}(v_{i'j}), i \neq i'$. Thus, the mappings Φ_{DD2} and Φ_{SDD} unify the main results (Theorem 1 of [7]) and (Theorem 1 of [9]), respectively, as multi-colorings of the pattern graph.

Denote by π_c and π_r permutations of columns and rows of matrix A, respectively, and let P and Q be the associated permutation matrices. It is clear that the pattern graph of the permuted matrix QAP, in general, will be structurally different from the pattern graph of A. On the other hand, as shown above, in a direct determination of matrix A, the multi-coloring constraints are expressed in terms of the paths in the pattern graph. We claim that for vertices v_{ij} and v_{kl}, $v_{ij} \sim v_{kl}$ in $G_{\mathcal{P}}(A)$ if and only if $v_{\pi_r(i)\pi_c(j)} \sim v_{\pi_r(k)\pi_c(l)}$ in $G_{\mathcal{P}}(PAQ)$. First, consider the case $i = k$ (the case when $j = l$ is analogous). Clearly, $v_{\pi_r(i)\pi_c(j)} \sim v_{\pi_r(i)\pi_c(l)}$ in $G_{\mathcal{P}}(PAQ)$. If $i \neq k$ and $j \neq l$, then we must have that there is an index j' such that

$$v_{ij} \sim v_{kl} = v_{ij} \sim v_{ij'} \sim v_{kj'} \sim v_{kl}$$

or that there is an index i' such that

$$v_{ij} \sim v_{kl} = v_{ij} \sim v_{i'j} \sim v_{i'l} \sim v_{kl}.$$

Considering each subpath separately and with a similar argument as above it is evident that $v_{\pi_r(i)\pi_c(j)} \sim v_{\pi_r(k)\pi_c(l)}, i \neq k$ and $j \neq l$ in $G_{\mathcal{P}}(PAQ)$.

Let $\chi_{SDD}(G_{\mathcal{P}}(A))$ and $\chi_{DD2}(G_{\mathcal{P}}(A))$, respectively, denote the minimum colors needed in a corresponding multi-coloring. The following results now follow from the discussion above.

Theorem 1 $\chi_{SDD}(G_{\mathcal{P}}(A)) = \chi_{SDD}(G_{\mathcal{P}}(P^{\top}AP))$.

Theorem 2 $\chi_{DD2}(G_{\mathcal{P}}(A)) = \chi_{DD2}(G_{\mathcal{P}}(QAP))$.

References

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

[2] T. F. Coleman and A. Verma. The efficient computation of sparse Jacobian matrices using automatic differentiation. *SIAM J. Sci. Comput.*, 19(4):1210–1233, 1998.

[3] A. H. Gebremedhin, F. Manne, and A. Pothen. What color is your Jacobian? Graph Coloring for Computing Derivatives. *SIAM Rev.*, 47(4):629–705, 2005.

[4] A. Griewank and A. Walther. Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. 2nd edn. SIAM, Philadelphia, PA. 2008.

[5] A. S. Hossain and T. Steihaug. Computing a sparse Jacobian matrix by rows and columns. *Optimization Methods and Software*, 10(1):33–48, 1998.

[6] S. Hossain and T. Steihaug. Optimal direct determination of sparse Jacobian matrices. *Optimization Methods and Software*, 28(6):1218–1232, 2013.

[7] S. Hossain and T. Steihaug. Graph models and their efficient implementation for sparse Jacobian matrix determination. *Discrete Applied Mathematics*, 161(12):1747 – 1754, 2013.

[8] M. Hasan, S. Hossain, A. I. Khan, N. H. Mithila, and A.H. Suny. DSJM: a software toolkit for direct determination of sparse Jacobian matrices. In *Proceedings of the 5th International Conference on Mathematical Software, ICMS*, volume 9725 of *LNCS*, pages 275–283, Berlin,Germany, 2016. Springer.

[9] S. Hossain and N. H. Mithila. Pattern Graph for Sparse Hessian Matrix Determination. *Optimization Methods and Software.* DOI: 10.1080/10556788.2018.1458849, 2018.

[10] G. N. Newsam and J. D. Ramsdell. Estimation of sparse Jacobian matrices. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):404–417, 1983.

[11] JS. Park, M. Penner, and V. K. Prasanna. Optimizing graph algorithms for improved cache performance. *IEEE Trans. Parallel Distrib. Syst.*, 15(9):769–782, September 2004.

Robust Matching Augmentation

Viktor Bindewald[1], Felix Hommelsheim[1], Moritz Mühlenthaler[1], Oliver Schaudt[2]

[1] Fakultät für Mathematik, TU Dortmund University, Germany

{viktor.bindewald, felix.hommelsheim, moritz.muehlenthaler}@math.tu-dortmund.de

[2] RWTH Aachen University, Germany

schaudt@mathc.rwth-aachen.de

Abstract

The *matching preclusion number* of a graph is the minimal number of edges whose removal destroys all perfect matchings. We provide algorithms and hardness results for the task of increasing the matching preclusion number from one to two in bipartite graphs at minimal cost. Our motivation is to make matchings of a graph robust against the failure of a single edge. Our methods rely on a close relationship to the classical strong connectivity augmentation problem. For the unit weight problem we provide a deterministic $\log_2 n$-factor approximation algorithm, as well as polynomial-time algorithms for graphs of bounded treewidth and chordal-bipartite graphs.

Keywords : *Matchings, Robustness, Connectivity Augmentation*

1 Introduction

Suppose we are given a bipartite graph and some adversary may remove any single edge from it with the intention of destroying all perfect matchings of the graph. How many edges do we need to buy, such that the adversary does not succeed? The matching preclusion number of a graph is the minimal number of edges that need to be removed, such that the resulting graph has no perfect matching. We call a graph *robust*, if its matching preclusion number is at least two. That is, the adversary cannot be successful on a robust graph, since no removal of a single edge destroys all perfect matchings. Given a bipartite graph G that admits a perfect matching, the problem Robust Matching Augmentation asks for a minimum-cardinality set L from the bipartite complement of G, such that $G+L$ is robust. This problem fits into the context of *augmentation problem* as follows. An augmentation problem asks for a minimum-cost supergraph with a certain property, which is typically related to connectivity (for example, Strong Connectivity Augmentation, see [5, 7]), but for instance, hamiltonicity has also been considered (Hamiltonian Completion [6, GT34]). In our setting, the property of interest is robustness.

It has been shown in [4, 8] that it is NP-complete to decide if the matching preclusion number of a graph is at least k, if k is part of the instance. Note that if k is constant, we can check this property in polynomial time by enumeration. Robust matchings with a given recovery budget have been considered by Dourado et al. in [4]. Our notion of robustness corresponds to *1-robust* ∞*-recoverable* in their terminology. They show that it is NP-hard to determine, if a graph admits a perfect matching M, such that M can be repaired by changing at most r edges after the removal of any single edge. Their hardness result does not apply to Robust Matching Augmentation, since we have no repair budget. Furthermore, Robust Matching Augmentation is a special case of the bulk-robust assignment problem studied by Adjiashvili et al. in [1]. They consider explicitly given subsets of edges of a graph that may fail and ask for a minimum-cost subgraph that admits a perfect matching after any of the failure scenarios have emerged. They provide a randomized $O(\log n)$-factor approximation algorithm for this problem, which can also be applied to Robust Matching Augmentation.

However, the hardness results for bulk-robust problems in [2, 1] do not apply in our setting. We show that the randomized $O(\log n)$-factor approximation algorithm from [1] is essentially optimal for ROBUST MATCHING AUGMENTATION.

Theorem 1 ROBUST MATCHING AUGMENTATION *admits no $o(\log n)$-factor approximation unless* $\mathsf{P} = \mathsf{NP}$.

Furthermore, we provide polynomial-time algorithms for ROBUST MATCHING AUGMENTATION on chordal-bipartite graphs and graphs of bounded treewidth, as well as a *deterministic* $\log_2 n$-factor approximation algorithm. Our main technical result is Theorem 2, which establishes a close relation between ROBUST MATCHING AUGMENTATION and the problems SET COVER and STRONG CONNECTIVITY AUGMENTATION. All our algorithmic results follow from Theorem 2.

2 Results

We first give a statement of our main result and then sketch the main ideas behind the proof. Finally we present some algorithmic consequences of the main result and discuss a generalization of the problem to matchings of a specified size.
For our main result we need some notion of SET COVER. Therefore let $(U, \mathcal{S})$ be an instance of SET COVER. The *incidence graph* of a SET COVER instance $(U, \mathcal{S})$ is an undirected bipartite graph with bipartition $(U, \mathcal{S})$, that has an edge us iff the item $u \in U$ is contained in the set $s \in \mathcal{S}$.

Theorem 2 *There is a polynomial-time algorithm that, given an instance* $\mathrm{I} = (G)$ *of* ROBUST MATCHING AUGMENTATION*, computes two instances $\mathcal{A}_1$ and $\mathcal{A}_2$ of* SET COVER*, such that the following holds:*

1. $\mathrm{OPT}(\mathrm{I}) = \max\{\mathrm{OPT}(\mathcal{A}_1), \mathrm{OPT}(\mathcal{A}_2)\}$.
2. *From a solution C_1 of $\mathcal{A}_1$ and a solution C_2 of $\mathcal{A}_2$ we can construct in polynomial time a solution L of* I*, such that $|L| = \max\{|C_1|, |C_2|\}$.*

There is a close relation between robustness and strong connectivity, which is an important ingredient in the proof of Theorem 2. Let $G = (U + W, E)$ be a graph and let M be a perfect matching in G. Since M is a perfect matching, each edge $e \in M$ is incident to a single vertex u_e of U. We consider two directed auxiliary graphs $D_1(G, M) = (U, A_1)$ and $D_2(G, M) = (W, A_2)$, whose arc-sets are given by

$$A_1 := \{uu' \mid u, u' \in U : \text{there is a vertex } w \in W \text{ such that } uw \in M \text{ and } wu' \in E \setminus M\},$$
$$A_2 := \{ww' \mid w, w' \in W : \text{there is a vertex } u \in U \text{ such that } wu \in M \text{ and } uw' \in E \setminus M\}.$$

It can be argued that all of the statements that follow hold for $D_1(G, M)$ as well as $D_2(G, M)$, so we will refer to $D_1(G, M)$ just as $D(G, M)$ and may omit G and M if there is no risk of confusion.

Proposition 1 *M is robust if and only if each strongly connected component of $D(G, M)$ is non-trivial, that is, it contains at least two vertices.*

A vertex of a digraph is called a *source* (*sink*) if it has no incoming (outgoing) arc. Consider the condensation $C(D)$ of a digraph D, that is, the directed acyclic graph of strongly connected components of D. We call a source or sink of $C(D)$ *strong* if the corresponding strongly connected component of G is non-trivial. From Proposition 1 it follows that strong sources and sinks are robust against the failure of a single matching edge. We may assume without loss of generality that a minimum-cardinality set L of arcs such that $G + L$ is robust connects sinks of $C(D)$ to sources of $C(D)$.

The main idea of the algorithm mentioned in Theorem 2 is the following. We first pick an arbitrary perfect matching M of G. Our goal is to select a suitable set of sources and sinks of $C(D)$ such that we can use the Eswaran-Tarjan algorithm [5] on these sources and sinks in order to establish in polynomial time the condition that every edge of M is contained in an M-alternating cycle. We call an edge $e \in M$ *critical*, if its removal destroys all perfect matchings of G. For this purpose we construct from G two SET COVER instances $\mathcal{B}_1 = (M, \mathcal{S}_1)$ and $\mathcal{B}_2 = (M, \mathcal{S}_2)$. The sets $\mathcal{S}_1$ and $\mathcal{S}_2$ are constructed as follows: For each source s of $C(D)$, we add a set X_s to $\mathcal{S}_1$, where X_s contains all critical edges that are reachable from any node in the component s in D. Similarly, for each sink t of $C(D)$ we add a set X_t to $\mathcal{S}_2$, where X_t contains all critical edges from which we can reach the component.

Taking a closer look at the incidence graphs of the SET COVER instances, we can show that the property of being chordal bipartite is preserved from the original graph G. Since SET COVER can be solved in polynomial time on instances where the incidence graph is chordal bipartite [9], we get the following result.

Corollary 1 ROBUST MATCHING AUGMENTATION *restricted to chordal bipartite graphs admits a polynomial-time algorithm.*

Unfortunately, if the input graph has bounded treewidth, the incidence graph of the SET COVER instance does not necessarily have this property. However, by using the structure of $C(D)$ instead of the SET COVER instance, one can obtain a polynomial-time algorithm for graphs of bounded treewidth.

Corollary 2 ROBUST MATCHING AUGMENTATION *restricted to graphs of bounded treewidth admits a polynomial-time algorithm.*

Furthermore, from Theorem 2, we directly obtain an approximation algorithm for ROBUST MATCHING AUGMENTATION by applying the greedy algorithm for SET COVER.

Corollary 3 ROBUST MATCHING AUGMENTATION *admits a polynomial-time* $\log n$*-factor approximation algorithm.*

Finally, we relax the requirement of having a perfect matching in the graph. In fact, all of our algorithmic results for ROBUST MATCHING AUGMENTATION generalize to the setting where we desire to have a matching of size k after deleting any single edge from a graph.

The main idea of the proof is to add new vertices and edges, such that we obtain a perfect matching in the new graph. An optimal set of edges to be added in the new graph then corresponds to an optimal set of edges to be added in the original graph and vice versa. The transformation preserves the approximation guarantee, the property of being chordal bipartite and increases the treewidth by at most 2.

References

[1] Adjiashvili, David and Bindewald, Viktor and Michaels, Dennis. *Robust Assignments via Ear Decompositions and Randomized Rounding.* ICALP, 55(1):71:1–71:14, 2016.

[2] Adjiashvili, David and Stiller, Sebastian and Zenklusen, Rico. *Bulk-robust combinatorial optimization.* Mathematical Programming, 149(1-2):361–390, 2015.

[3] Robert C. Brigham and Frank Harary and Elizabeth C. Violin, and Jay Yellen. *Perfect-matching preclusion..* Congressus Numerantium, 174(1):185–192, 2005.

[4] Dourado, Mitre C. and Meierling, Dirk and Penso, Lucia D. and Rautenbach, Dieter and Protti, Fabio and de Almeida, Aline Ribeiro. *Robust recoverable perfect matchings.* Networks, 66(3):210–213, 2015.

[5] Eswaran, Kapali P. and Tarjan, Robert E. *Augmentation problems.* SIAM Journal on Computing, 5(4):653–665, 1976.

[6] Garey, Michael R. and Johnson, David S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[7] András Frank and Tibor Jordán. Graph connectivity augmentation. In *Handbook of Graph Theory, Combinatorial Optimization, and Algorithms*, chapter 14, pages 313–346. CRC Press, 2015.

[8] Mathieu Lacroix, A. Ridha Mahjoub, Sébastien Martin, and Christophe Picouleau. On the np-completeness of the perfect matching free subgraph problem. *Theoretical Computer Science*, 423:25–29, 2012. doi:10.1016/j.tcs.2011.12.065.

[9] Schrijver, Alexander *Combinatorial Optimization - Polyhedra and Efficiency.* Algorithms and Combinatorics, 2003.

Successive Shortest Path Algorithm for Flows in Dynamic Graphs

Mathilde Vernet[1], Maciej Drozdowski[2], Yoann Pigné[1], Eric Sanlaville[1]

[1] Normandie Univ, UNIHAVRE, UNIROUEN, INSA Rouen, LITIS, 76600 Le Havre, France
`{mathilde.vernet,yoann.pigne,eric.sanlaville}@univ-lehavre.fr`

[2] Institute of Computing Science, Poznań University of Technology, Piotrowo 2, Poznań 60-965, Poland
`Maciej.Drozdowski@cs.put.poznan.pl`

Abstract

This work focuses on the minimum cost flow problem in dynamic graphs. We use a model in which there are no travel time on arcs, and storage on vertices is not allowed. Every other graph parameter is time-dependent. We propose a method using the successive shortest path algorithm that avoids using the time-expanded graph that significantly increases the complexity of the problem. Our method is implemented using the GraphStream library.

Keywords : *Dynamic graphs, minimum cost flow, successive shortest path algorithm.*

1 Introduction

1.1 Dynamic graphs

Static graph models are not always sufficient to describe some real-world problems and their evolution through time. Adding a time dimension to graphs offers extra modeling possibilities. This is a motivation to explore known problems on graphs in a dynamic context.

A dynamic graph is defined on a time horizon. Vertices and arcs can be modified over the time horizon as well as any parameter the graph has. Holme (2015) gives an overview of the different dynamic graph models and their applications.

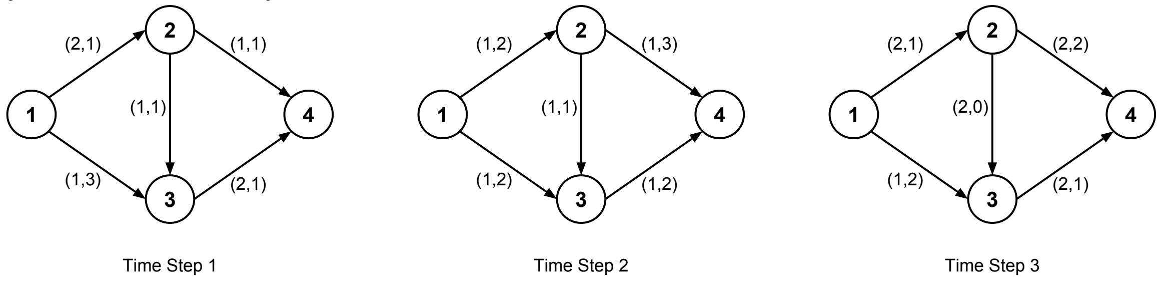
1.2 Flows in dynamic graphs

We focus on flow problems in dynamic graphs. Flow problems in graphs are modeled using various graph parameters such as arc capacity, arc cost. As we consider dynamic graphs, travel time on arcs or storage on vertices with a storage cost can be defined. All these parameters can vary over time or not.

Flow problems as defined in static graphs (Ahuja et al., 1993) have a dynamic version in dynamic graphs where a given amount of flow units has to be sent in the graph from a source to a sink over the time horizon. This work focuses on the minimum cost flow problem.

Though many different works on dynamic flows have been carried on (Wilkinson, 1971; Hoppe and Tardos, 2000; Skutella, 2009), most of the times, travel time is added on arcs and the graph stays unchanged over time. Here, travel time is negligible and flow cannot be stored in the nodes, while other parameters can change over time.

FIG. 1: Dynamic graph with 3 time steps. On arc ij, couple (u_{ij}, c_{ij}) represents the capacity of arc ij and the cost of arc ij.



1.3 This work's hypotheses

We assume a model without travel time on arcs and without storage on vertices. Flow units travel instantaneously from a vertex to its neighbor and when arrive at a vertex they are not allowed to "wait" before going to a neighbor. Arc capacities and arc costs are time-dependent and supposed integer. Without loss of generality, vertices and arcs are present all over the time horizon. Their accessibility is determined by arc capacities. If an arc capacity is 0, this arc cannot be used by flow units. If every arc going into a vertex has 0 capacity, this vertex is unreachable.

At each time step t, our graph can be seen as a static graph. The static graphs defined at each time step are independent. We call *t-graph* the static graph at time step t, noted G_t.

This set of hypotheses can, for instance, be used to model power grid systems.

2 Minimum cost flow problem in dynamic graphs

We consider the minimum cost flow problem in which, as in the static context, a predetermined amount of flow has to be sent in the graph from the source to the sink at the minimum possible cost. The amount of flow is defined globally for the whole time horizon and is not defined at each time step.

Although the t-graphs are independent, this problem cannot be solved by computing the minimum cost flow in each t-graph because this is not equivalent to the optimum solution in the dynamic graph. Figure 1 shows an example of a dynamic graph with 3 time steps, without travel time or storage. Sending 3 flow units in this graph from vertex 1 to vertex 4 would cost 6. To get the minimal cost, we send 1 flow unit with cost 2 through path $(1, 2, 4)$ at time step 1, and 2 flow units with cost 4 through path $(1, 2, 3, 4)$ at time step 3. Nothing is sent at time step 2. Note that the minimum cost to send 3 units in G_1 and G_3 is 9 and 8 respectively. G_2 admits a maximum flow of value 2 which can be sent a cost 9.

3 Solving methods

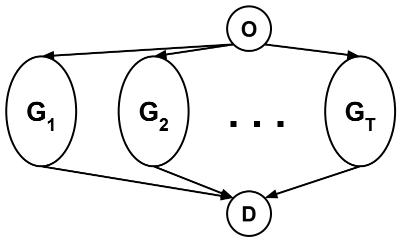
3.1 Using the time-expanded graph

The most popular method used to solve optimization problems on dynamic graphs is to build the time-expanded graph to have a static representation of it and use algorithms developed for static graphs on it (Minieka, 1973; Parpalea and Ciurea, 2011). Using the time-expanded graph increases the size of the problem. The number of vertices and arcs is multiplied by the number of time steps T.

Figure 2 represents the time-expanded graph for our model. Each t-graph G_t has n vertices and m arcs, therefore the time-expanded graph has $T \cdot n + 2$ vertices and $T \cdot m + 2 \cdot T$ arcs.

A regular Successive Shortest Path (SSP) algorithm (Ahuja et al., 1993) on a graph with a unique source and a unique sink using Dijkstra (Ahuja et al., 1993) to solve the shortest path problem has complexity $O(U \cdot (m + n \cdot log(n)))$, where U is the supply of the source, n is the number of vertices and m the number of arcs. As the expanded graph has $T \cdot n + 2$ vertices and $T \cdot m + 2 \cdot T$ arcs, SSP on the time-expanded graph has complexity $O(U \cdot T \cdot (m + n \cdot log(n \cdot T)))$.

FIG. 2: Time-expanded graph for our model. We add a source O connected to the source of each t-graph G_t and a sink D connected to the sink of each t-graph G_t. The new arcs all have infinite capacity and null cost.



3.2 Dynamic successive shortest path Algorithm

In order to improve the complexity, we propose a method that does not use the time-expanded graph. It adapts SSP to a dynamic graph on T time steps without travel time nor storage. At each iteration, the algorithm performs one search of a shortest path (as in classical SSP) but does it on one specific t-graph, which had the shortest path (*i.e.* the minimum cost augmenting path) at the previous iteration. It works as such :

1. Initialization : Execute Dijkstra using the cost as arc length on each t-graph G_t, $1 \leq t \leq T - 1$ and let $\bar{t} = T$. *(Example of Figure 1, 3 units to send : Shortest path in G_1 is $(1, 2, 4)$ cost 2, shortest path in G_2 is $(1, 3, 4)$ cost 4, $\bar{t} = 3$)*

2. Execute Dijkstra on $G_{\bar{t}}$'s residual graph *(Example, 1^{st} iteration : Shortest path in G_3 is $(1, 2, 3, 4)$ cost 2. 2^{nd} iteration : Shortest path in G_1 is $(1, 2, 3, 4)$ cost 3)*

3. Compare the real cost of the shortest path from the source to the sink in each G_t *(Example, 1^{st} iteration: Shortest path is $(1, 2, 4)$ in G_1. 2^{nd} iteration: Shortest path is $(1, 2, 3, 4)$ in G_3)*

4. Let $\bar{t}$ be the time step which has the shortest path from the source to the sink and update reduced costs, vertex potentials, flow and capacity on shortest path on $G_{\bar{t}}$'s residual graph *(Example, 1^{st} iteration : $\bar{t} = 1$, update G_1's residual graph. 2^{nd} iteration : $\bar{t} = 3$, update G_3's residual graph)*

5. Until there are no flow units left to be sent, go back to step 2 *(Example, 1^{st} iteration : 2 units left to be sent. 2^{nd} iteration : 0 units left to be sent)*

We implemented this algorithm using the GraphStream library[1] (Dutot et al., 2007).

3.3 Complexity

The initialization phase needs $O(T \cdot (m + n \cdot log(n)))$ because we execute Dijkstra's algorithm on T graphs with n vertices and m arcs.

One iteration takes $O(m + n \cdot log(n) + log(T))$ because we execute Dijkstra's algorithm on one t-graph, the comparison of T paths costs can be done in $O(log(T))$ and the residual graph update can be done in $O(n + m)$. As for a regular SSP algorithm with integer capacities, the number of iteration is bounded by U, which is the source supply.

[1]http://graphstream-project.org

Our algorithm has complexity $O((U+T)\cdot(m+n\cdot log(n))+U\cdot log(T))$ which improves the complexity of the algorithm using the time-expanded graph.

3.4 Correctness

We can prove the correctness of this algorithm using a swapping argument. Suppose f^* is an optimum flow on the graph and f the flow given by our algorithm. Due to the nature of the algorithm, it is possible, starting from f^*, to repeatedly swap one unit of flow from some graph G_i for one unit in some graph G_j so that the cost is non-increasing. The sequence of swaps eventually transforms f^* into f. The complete proof is not given in this abstract.

4 Conclusion

We presented a dynamic graph model without travel time nor storage and proposed for this model a method to solve the minimum cost flow problem based on the successive shortest path algorithm. Unlike methods widely used to solve optimization problems on dynamic graphs, we do not use the time-expanded graph, allowing us to work on much smaller graphs and therefore significantly improve the complexity. An experimental study to validate our approach is a work in progress. Furthermore, the same principle could be used for other dynamic flow problems.

Acknowledgment

The project is co-financed by the European Union with the European regional development fund (ERDF) and by the Normandie Regional Council (CLASSE2 project). The French-Polish collaboration is possible thanks to a Polonium Project.

References

Ahuja, R. K., T. L. Magnanti, and J. B. Orlin (1993). Network flows: theory, algorithms, and applications.

Dutot, A., F. Guinand, D. Olivier, and Y. Pigné (2007). Graphstream: A tool for bridging the gap between complex systems and dynamic graphs. In *Emergent Properties in Natural and Artificial Complex Systems. Satellite Conference within the 4th European Conference on Complex Systems (ECCS'2007).*

Holme, P. (2015). Modern temporal network theory: a colloquium. *The European Physical Journal B 88*(9), 234.

Hoppe, B. and É. Tardos (2000). The quickest transshipment problem. *Mathematics of Operations Research 25*(1), 36–62.

Minieka, E. (1973). Maximal, lexicographic, and dynamic network flows. *Operations Research 21*(2), 517–527.

Parpalea, M. and E. Ciurea (2011). The quickest maximum dynamic flow of minimum cost. *International Journal of Applied Mathematics and Informatics 3*(5), 266–274.

Skutella, M. (2009). An introduction to network flows over time. In *Research trends in combinatorial optimization*, pp. 451–482. Springer.

Wilkinson, W. L. (1971). An algorithm for universal maximal dynamic flows in a network. *Operations Research 19*(7), 1602–1612.

Fully leafed induced subtrees (extended abstract)[*]

A. Blondin Massé[1], J. de Carufel[2], A. Goupil[2], M. Lapointe[1], É. Nadeau[1], É. Vandomme[1]

[1] Laboratoire de Combinatoire et d'Informatique Mathématique, Université du Québec à Montréal, Canada

[2] Laboratoire Interdisciplinaire de Recherche en Imagerie et en Combinatoire, Université du Québec à Trois-Rivières, Canada

Abstract

Subtrees of graphs, as well as their number of leaves, have been investigated by various communities: from discrete mathematics to data mining and information retrieval. We consider a variant where we require the subtrees to be induced and compute their maximal number of leaves. The problem, which is NP-complete in general, becomes polynomial in the case of trees. The leaf function associates to a number n the maximal number of leaves an induced subtree of size n can have. To compute the leaf function, we provide an efficient branch and bound algorithm. In the particular case of trees, we provide a polynomial algorithm using the dynamic programming paradigm.

Keywords : *graph theory, induced subtrees, optimization problem, number of leaves*

1 Introduction

In the past decades, many researchers coming from various communities extensively studied subtrees of graphs and their number of leaves. For instance in 1984, Payan *et al.* [9] discussed the maximum number of leaves, called the *leaf number*, that can be realized by a spanning tree of a given graph. This problem, called the *Maximum Leaf Spanning Tree problem* (MLST), is known to be NP-complete even in the case of regular graphs of degree 4 [8] and has attracted interest in the telecommunication network community [3, 4]. The *frequent subtree mining problem* [5] investigated in the data mining community, has applications in biology. The detection of subgraph patterns such as induced subtrees is useful in information retrieval [12] and requires efficient algorithms for the enumeration of induced subtrees. In this perspective, Wasa *et al.* [11] proposed an efficient parametrized algorithm for the generation of induced subtrees in a graph. Note that the *induced* property requirement brings an interesting constraint on subtrees, yielding distinctive structures with respect to other constraints such as in the MLST problem. A first result given by Erdös *et al.* in 1986, showed that the problem of finding an induced subtree of a given graph G with more than i vertices is NP-complete [7].

Among induced subtrees of simple graphs, we focus in particular on those with a maximal number of leaves. We call these objects *fully leafed induced subtrees* (FLIS). Particular instances of the FLIS recently appeared in a paper of Blondin Massé *et al.* [2], where the authors considered the maximal number of leaves that can be realized by tree-like polyominoes, respectively polycubes. Their investigation led to the discovery of a new 3D tree-like polycube structure that realizes the maximal number of leaves constraint. The observation that tree-like polyominoes and polycubes are induced subgraphs of the lattices $\mathbb{Z}^2$ and $\mathbb{Z}^3$ respectively leads naturally to the investigation of FLIS in general simple graphs, either finite or infinite.

[*]This document is an extended abstract of the paper [1] available on arXiv.

2 Fully leafed induced subtrees

We introduce the decision problem called *Leafed Induced Subtree problem (*LIS) and its associated optimization problem MLIS:

LIS. Given a simple graph G and two positive integers i and ℓ, does there exist an induced subtree of G with i vertices and ℓ leaves?

MLIS. Given a simple graph G on n vertices, what is the maximum number of leaves, $L_G(i)$, that can be realized by an induced subtree of G with i vertices, for $i \in \{0, 1, \ldots, n\}$?

We believe that induced subtrees with the maximal number of leaves are interesting candidates for the representation of structures appearing in nature and in particular in molecular networks. Indeed, in chemical graph theory, subtrees are known to be useful in the computation of the *Wiener index* of chemical graph, that corresponds to a topological index of a molecule [10]. The results of [2] and [10] suggest that a thorough investigation of subtrees, and in particular induced subtrees with many leaves, could lead to the discovery of combinatorial structures relevant to chemical graph theory.

Definition 1 (Leaf function) *For a graph $G = (V, E)$, the* leaf function *of G, denoted by L_G, is the function with domain $\{0, 1, 2, \ldots, |V|\}$ which associates to i the maximum number of leaves that can have an induced subtree of size i of G. As is customary, we set* $\max \emptyset = -\infty$. *An induced subtree T of G with i vertices is called* fully leafed *when its number of leaves is exactly $L_G(i)$.*

The following observations are immediate. Consider a graph G with at least 3 vertices. The sequence $(L_G(i))_{i=0,1,\ldots,|G|}$ is non-decreasing if and only if G is a tree. Moreover, we have $L_G(0) = 0 = L_G(1)$ and $L_G(2) = 2$ if G contains at least one edge. If G is connected and non-isomorphic to a complete graph, then $L_G(3) = 2$.

While it is easy to determine the leaf function for some well-known families of graphs (such as complete graphs, wheels etc.), in general the problem is much harder.

3 Complexity and algorithms

First, we prove that the problem LIS is NP-complete by reducing it from the *Independent Set* problem. To tackle the MLIS problem and compute the leaf function, we provide a non trivial branch and bound algorithm. The algorithm is based on a data structure that we call an *induced subtree configuration.*

Definition 2 *Let $G = (V, E)$ be a simple graph and $\Gamma = \{\text{green}, \text{yellow}, \text{red}, \text{blue}\}$ be a set of colors with coloring functions $c : V \to \Gamma$. An* induced subtree configuration *of G is an ordered pair $C = (c, H)$, where c is a coloring and H is a stack of colorings called the* history *of C.*

All colorings $c : V \to \Gamma$ must satisfy the following conditions for any $u, v \in V$:

(i) The subgraph induced by $c^{-1}(\text{green})$ is a tree;

(ii) If $c(u) = \text{green}$ and $\{u, v\} \in E$, then $c(v) \in \{\text{green}, \text{yellow}, \text{red}\}$;

(iii) If $c(u) = \text{yellow}$, then $|c^{-1}(\text{green}) \cap N(u)| = 1$, where $N(u)$ denotes the set of neighbors of u.

The initial induced subtree configuration *of a graph G is the pair (c_{blue}, H) where $c_{\text{blue}}(v) =$* blue *for all $v \in G$ and H is the empty stack. When the context is clear, C is simply called a* configuration.

Roughly speaking, a configuration is an induced subtree enriched with information that allows one to generate other induced subtrees either by extension, by exclusion or by backtracking. The colors assigned to the vertices can be interpreted as follow. The *green* vertices are the confirmed vertices to be included in a subtree. Since each *yellow* vertex is connected to exactly one *green* vertex, any *yellow* vertex can be safely added to the *green* subtree to create a new induced subtree. The *red* vertices are those that are excluded from any possible tree extension. The exclusion of a *red* vertex is done either because it is adjacent to more than one *green* vertex and its addition would create a cycle or because it is explicitly excluded for generation purposes. Finally, the *blue* vertices are available vertices that have not been considered yet and that could be considered later. It is convenient to save in the stack H the colorations from which C was obtained.

Contrary to a naive algorithm that considers all induced subtrees to compute the maximal number of leaves, the strategy prunes the search space by discarding induced subtrees that cannot be extended to fully leafed subtrees (see Figure 1). Therefore, given an induced subtree configuration of n *green* vertices, we define a function $C.\textsc{LeafPotential}(n')$, for $n \leq n' \leq |V|$, which computes an upper bound on the number of leaves that can be reached by extending the current configuration C to a configuration of n' vertices. To compute this upper bound we consider an optimistic scenario in which all available *yellow* and *blue* vertices that are close enough can be safely colored in *green* without creating a cycle, whatever the order in which they are selected.

Proposition 1 *Let C be any configuration of a simple graph $G = (V, E)$ with $n \geq 3$* green *vertices and let n' be an integer such that $n \leq n' \leq |V|$. Then any extension of C to a configuration of n' vertices has at most $C.\textsc{LeafPotential}(n')$ leaves, where $C.\textsc{LeafPotential}(n')$ is the operator described above.*

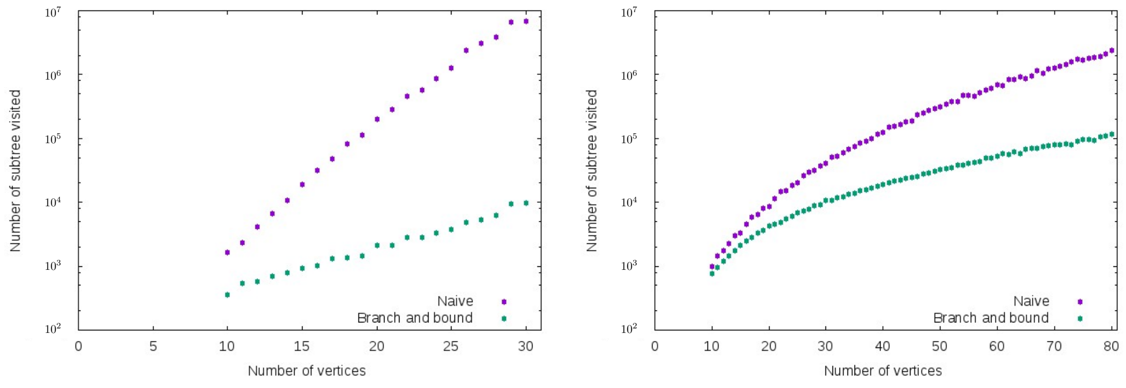


FIG. 1: Number of induced subtrees visited for samples of 10 random graphs with edge probability 0.3 (on the left) and with edge probability 0.8 (on the right).

When restricted to the case of trees, we show that the MLIS problem is polynomial using a dynamic programming strategy.

Theorem 1 *Let $T = (V, E)$ be a tree with $n \geq 2$ vertices. Then L_T can be computed in $\mathcal{O}(n^3\Delta)$ time and $\mathcal{O}(n^2)$ space where Δ denotes the maximal degree of a vertex in T.*

Notice that a naive greedy approach cannot work, even in the case of trees, because a fully leafed induced subtree with n vertices is not necessarily a subtree of a fully leafed induced subtree with $n + 1$ vertices. Both algorithms are available, with examples, in a public GitHub repository[1].

[1]`https://github.com/enadeau/fully-leafed-induced-subtrees`

4 Perspectives

There seems to be some room for improving and specializing the branch and bound algorithm. For example, we are able to speed up the computations for the hypercube Q_6 by taking into account some symmetries, but significant improvements could be done by discarding more configurations by exploiting the complete automorphism group of the hypercube. It seems reasonable to expect similar speed up for other highly symmetric graphs.

From a theoretical perspective, it is not clear if the algorithm described for the trees is optimal. As a last observation, we believe that it would be interesting to investigate the natural problems of counting and generating related to the concept of fully leafed induced subtrees.

References

[1] Alexandre Blondin Massé, Julien de Carufel, Alain Goupil, Mélodie Lapointe, Émile Nadeau, and Élise Vandomme. Fully leafed induced subtrees. `https://arxiv.org/abs/1709.09808` *arXiv preprint.*

[2] Alexandre Blondin Massé, Julien de Carufel, Alain Goupil, and Maxime Samson. Fully leafed tree-like polyominoes and polycubes. In *Combinatorial Algorithms*, LNCS 28th International Workshop, IWOCA 2017, New-Castle, Australia, Springer. To appear.

[3] Azzedine Boukerche, Xuzhen Cheng, and Joseph Linus. A performance evaluation of a novel energy-aware data-centric routing algorithm in wireless sensor networks. *Wireless Networks*, 11(5):619–635, 2005.

[4] Si Chen, Ivana Ljubič, and Subramanian Raghavan. The generalized regenerator location problem. *INFORMS Journal on Computing*, 27(2):204–220, 2015.

[5] Akshay Deepak, David Fernández-Baca, Srikanta Tirthapura, Michael J. Sanderson, and Michelle M. Evominer: frequent subtree mining in phylogenetic databases. *Knowledge and Information Systems*, 41(3):559–590, 2014.

[6] Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, Heidelberg, fourth edition, 2010.

[7] Paul Erdős, Michael Saks, and Vera T. Sós. Maximum induced trees in graphs. *J. Combin. Theory Ser. B*, 41(1):61–79, 1986.

[8] Michael R. Garey and David S. Johnson. *Computers and intractability.* W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[9] Charles Payan, Maurice Tchuente, and Nguyen Huy Xuong. Arbres avec un nombre maximum de sommets pendants. *Discrete Math.*, 49(3):267–273, 1984.

[10] Lászlò A. Székely and Hua Wang. On subtrees of trees. *Advances in Applied Mathematics*, 34(1):138–155, 2005.

[11] Kunihiro Wasa, Hiroki Arimura, and Takeaki Uno. Efficient enumeration of induced subtrees in a K-degenerate graph. In *Algorithms and computation*, LNCS 8889, 94–102. Springer, Cham, 2014.

[12] Mohammed J. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, 71–80, New York, NY, USA, 2002. ACM.

Graphs with at most one crossing

André C. Silva[1*], Alan Arroyo[2†], R. Bruce Richter[2‡], Orlando Lee[1§]

[1] Universidade Estadual de Campinas, Campinas, Brazil
{andre.silva,lee}@ic.unicamp.br
[2] University of Waterloo, Waterloo, Canada
{amarroyo,rbruce}@uwaterloo.ca

Abstract

The crossing number of a graph G is the least number of crossings over all possible drawings of G. We present a structural characterizations of graphs with crossing number one.

Keywords : *Graph Theory, Graph Drawing, Crossing Number.*

1 Introduction

For a graph G let $V(G)$ and $E(G)$ denote its vertex set and edge set respectively. We assume that graphs may have multiples edges but no loops. In the context of crossing number, we can interpret Kuratowski's classic characterization of planar graphs as: a graph has crossing number at least one if and only if it contains a subdivision of K_5 or $K_{3,3}$. In this paper we answer a similar question: when does a graph have crossing number at least 2? We answer this question by characterizing graphs with crossing number one.

This problem was already studied by Arroyo and Richter [AR17] in the context of *peripherally 4-connected graphs.* A graph G is *peripherally 4-connected* if G is 3-connected and for every vertex 3-cut X of G, and for any partition of the components of $G - X$ into two non-null subgraphs H and K, at least one of H or K has just one vertex. Two edges $e = x_1y_1$ and $f = x_2y_2$ are *linked* if either e and f are incident to a common vertex or there is a 3-cut X in G such that $X \subset \{x_1, y_1, x_2, y_2\}$ and the vertex in $\{x_1, y_1, x_2, y_2\}\backslash X$ is a vertex of a trivial component of $G - X$. Otherwise, e and f are *unlinked.* Arroyo and Richter [AR17] proved that a peripherally 4-connected non-planar graph G has crossing number at least two if and only if, any pair of unlinked edges e, f in G, there exists two vertex-disjoint cycles C_e and C_f in G with $e \in C_e$ and $f \in C_f$ (we say in this case that e, f are *separated by cycles* in G).

We assume the reader is familiar with the concept of drawings of graphs. We make no distinction between the elements of a graph and their representations in the drawing.

Also, we only concern ourselves with good drawings, that is any pair of edges may cross at most once, no adjacent edges cross and no three edges share a crossing point.

A *1-drawing* of a graph G is a drawing D of G with $cr(D) = 1$. A subgraph H of G is a 1-*subgraph* if $cr(H) = 1$.

A pair of edges e, f of a graph G is a *crossing pair* of G if there exists a 1-drawing of G in which e and f cross. A *potential crossing pair* of a graph G is a pair of edges e, f which is a crossing pair for every 1-subgraph H of G. Note that this does not mean that e and f cross in every 1-drawing of H – only that there exists some 1-drawing of H in which e and f cross. This also implies that e and f belong to every 1-subgraph of G.

The following is the main result of this paper.

[*] Research supported by FAPESP Proc. 2015/04385-0, 2014/14375-9 and 2015/11937-9, CNPq Proc. 311373/2015-1 [†] Supported by CONACyT [‡] Supported by NSERC [§] Research supported by CNPq Proc. 311373/2015-1, CNPq Proc. 425340/2016-3 and FAPESP Proc. 2015/11937-9.

Theorem 1 *Let G be a non-planar graph. Then $e, f \in E(G)$ is a crossing pair of G if and only if e, f is a potential crossing pair not separated by cycles.*

In Section 2 we give some terminology and state a result from Mohar that will be key in the proof of Theorem 1. In Section 3 we present the proof of Theorem 1.

2 Preliminaries

In this section we present some definitions and results which are used throughout the text. Through this section, let G be a graph.

For a subset S of $V(G)$, let $G-S$ denote the subgraph of G induced by $V(G)-S$. If $S = \{v\}$ we simply write $G - v$ instead of $G - \{v\}$. Let E denote a set of edges between vertices of G (possibly $E \nsubseteq E(G)$). Let $G + E$ and $G - E$ to denote the graphs $(V(G), E(G) \cup E)$ and $(V(G), E(G) \setminus E)$, respectively. If $E = \{e\}$, we simply write $G + e$ or $G - e$ instead.

Given a subgraph H of G, a path P is *H-avoiding* or *avoids H* if no internal vertex of P is in $V(H)$ and has both ends in $V(H)$.

Given graphs H, K and G, we say that H is a subdivision of K in G if H is a subgraph of G isomorphic to a subdivision of K. A *Kuratowski subgraph* of G is a subdivision of $K_{3,3}$ or K_5 in G.

Let G be a connected graph. A *node* of G is any vertex with degree different from 2. A *branch* of G is any path between main vertices of G that does not contain any node as an internal vertex.

A *k-separation* of a graph G is a pair of subgraphs $\{G_1, G_2\}$ such that $G = G_1 \cup G_2$, $E(G_1) \cap E(G_2) = \varnothing$, $|V(G_1) \cap V(G_2)| = k$, and for $i = 1, 2$, $|E(G_i)| \geq k$ and $V(G_i) - V(G_{3-i}) \neq \varnothing$.

A graph G is *nonseparable* if has no 0- or 1-separation and *separable* otherwise. Let G be a nonseparable graph and let $\{G_1, G_2\}$ be a 2-separation of G with $V(G_1) \cap V(G_2) = \{x, y\}$. A 2-separation is *elementary* if: either $G_1 - \{x, y\}$ or $G_2 - \{x, y\}$ is nonempty and connected; and either G_1 or G_2 is nonseparable. Graphs without elementary 2-separations are either 3-connected graphs, cycles, parallel edges, or rather small [Tut66].

Suppose $\{G_1, G_2\}$ is an elementary 2-separation of a nonseparable graph G with $\{x, y\} = V(G_1) \cap V(G_2)$. Let G_1' and G_2' obtained from G_1 and G_2 by adding an extra edge between x and y, respectively. This new edge is called a *virtual edge.* If G_i has no elementary 2-separation itself then G_i' is a *3-connected component* of G, for $i = 1, 2$. Otherwise, the 3-connected components of G are the 3-connected components of G_1' and G_2'. If G is separable, then the 3-connected components of G are the 3-connected components of its blocks. The 3-connected components of G are uniquely determined [Tut66]. They are also called *cleavage units* [Tut66].

A useful observation is that, by construction, any edge of G is in exactly one 3-connected component. Thus, for any 3-connected component H of G, there exists a corresponding subgraph H of G in which each virtual edge of H is replaced by an H-avoiding path in G.

Let C a cycle of G. Let P_1 and P_2 be a pair of disjoint paths both internally disjoint from C and whose ends are in $V(C)$. They are called a *pair of disjoint crossing paths* if the ends of P_1 and P_2 alternate in C.

A *tripod* in G with respect to C is a subdivision H of a $K_{2,3}$ in G together with three disjoint paths (possibly trivial) joining C with the part of size 3 in H. The tripod itself is edge-disjoint from C. We denote by $Aux(G, C)$ the graph obtained from G by adding a new vertex v and an edge vw for each $w \in V(C)$.

For a disc D in the plane, let $\partial(D)$ denote its boundary. We need the following result of Mohar.

Theorem 2 *[Moh94] Let G be a graph, C a cycle of G and D a disk. Let $\tilde{G} = Aux(G, C)$. There is a linear time algorithm that either finds an embedding of G in D with C on $\partial(D)$, or:*

(1) a pair of disjoints crossing paths (w.r.t. C),

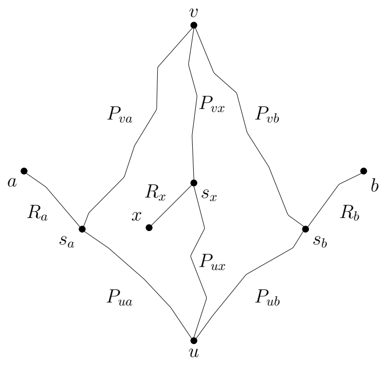


FIG. 1: The tripod in proof of Theorem 1.

(2) a tripod (w.r.t. C) or

(3) a Kuratowski subgraph contained in a 3-connected component of $\tilde{G}$ distinct from the 3-connected component of $\tilde{G}$ containing C.

3 Proof of Theorem 1

Proof : The sufficiency is a direct consequence of the next two lemmas. Their proofs are straightforward.

Lemma 1 *Let G be a graph and let $e, f \in E(G)$. If e, f are separated by cycles then e, f is not a crossing pair of G.*

Lemma 2 *If e, f is a crossing pair of G, then it is a potential crossing pair.*

We focus on the necessity. Let G be a graph and let e, f be a potential crossing pair not separated by cycles in G. Let $\{a, b\}$ and $\{x, y\}$ be the (all distinct) ends of e and f, respectively. As G is nonplanar, let H be a Kuratowski subgraph of G. We may assume that G is connected.

Our goal is to show that there exists an embedding of $G - \{e, f\}$ in a disk D on the plane with $\partial(D)$ containing a, x, b, y in this cyclic order. If so, we can simply draw e and f crossing on the exterior of D obtaining a 1-drawing of G.

Let ax, xb, by and ya be new edges and let C be the cycle induced by them. Let $G' = (G - \{e, f\}) \cup C$. If we obtain a disk embedding of G', we can delete the edges of C to obtain our desired embedding of $G - \{e, f\}$. So suppose there is no such embedding. By Theorem 2 one of (1)-(3) does holds.

Suppose (3) holds. Let $\tilde{G} = Aux(G', C)$. Let J be the 3-connected component of $\tilde{G}$ containing C. Let K be a Kuratowski subgraph of $\tilde{G}$ not in J. Let $\{G_1, G_2\}$ be an elementary 2-separation in $\tilde{G}$, with u, w the vertices of $G_1 \cap G_2$, such that $J \subseteq G_1 + uw$ and $K \subseteq G_2 + uw$, where uw is a virtual edge. There is a uw-path P in G_1 such that we can exchange the virtual edge uw of K for P to obtain a Kuratowski subgraph in $G_2 \cup P$. We will also call this subgraph K.

Since H is non-separable and $e, f \in E(H)$, there are vertex-disjoint paths in H from the two ends of e to the two ends of f; we may choose the labelling so that these are ax- and by-paths P_{ax} and P_{by}, respectively. At most one of these paths has an edge in G_2.

Suppose that P_{ax} contains an edge of G_2. Then P_{by} is disjoint from $\{u, w\}$ and we may replace P with $((P_{ax} \cap G_1) \cup P_{by}) + \{e, f\}$ to get a Kuratowski subgraph of G in which e, f are in the same branch. This contradicts the fact that e, f is a potential crossing pair. Therefore, we may assume that P_{ax} and P_{by} are contained in G_1.

We may assume P uses at most two edges of C. Since consecutive edges of C can be replaced by either e or f, in this case K converts to a Kuratowski subgraph of G containing only one of e and f, a contradiction. Thus, we may assume that, if there are two edges of C in P, then they are either ax and by or ay and bx.

In the first case, we may replace ax with P_{ax} and by with P_{by} to get a uw-walk W in G_2 that uses no edge of C. Thus, W contains a uw-path P' and $K \cup P'$ is a Kuratowski subgraph of $G - \{e, f\}$, a contradiction.

In the second case, we replace ay with $P_{ax} + f$ and bx with $P_{by} + e$ to get a uw-walk W in G_2 that uses no edge of C. In this case, G contains a Kuratowski subgraph K' in which any of e and f that are in K' are in the same K'-branch, again a contradiction.

If (1) holds, then the crossing paths together with e and f form disjoint cycles in G. Suppose (1) from Theorem 2 holds. If $G - \{e, f\}$ contains a pair of disjoint crossing pairs, but then these paths joined with e and f show that e and f are separated by cycles, a contradiction.

Finally, suppose (2) from Theorem 2 holds. We refer the reader to Figure 1 for a visual aid in the following definitions. Let T be a minimal (in the number of vertices) tripod in G'. As the tripod is edge-disjoint from C, $T \subseteq G - \{e, f\}$. Let K be the subdivision of $K_{2,3}$ in T. We may assume that $\{a, x, b\}$ are the vertices of C connected to K in T. Let $\{u, v\}$ and $S = \{s_a, s_x, s_b\}$ be the parts of K. For $i \in S$, let P_{ui} be the ui-subpath in $K \setminus (S \setminus \{i\})$. Similarly, define P_{vi} for v. For $j \in \{a, x, b\}$, let Q_j be the js_j-paths connecting C to K in T.

Because H is a Kuratowski subgraph and e and f are in different branches of H, $H - \{e, f\}$ is a connected. It follows that there is a yT-path in $H - \{e, f\}$ and, therefore, a yT-path in $G - \{e, f\}$. The reader may verify that if P is a $V(T)$-avoiding path from y to $V(T)$ which ends in $R_a - s_a$ or $R_b - s_b$ then $(T \cup y) + e + f$ contains a Kuratowski subgraph in which e and f is not a crossing pair; and, for $i \in \{a, b, c\}$, if y ends in $P_{ui} - \{s_a, s_b\}$ or $P_{vi} - \{s_a, s_b\}$, then e and f are separated by cycles in $(T \cup P) + e + f$.

So any $V(T)$-avoiding path from y to $V(T)$ in $G - \{e, f\}$ ends in $\{s_a, s_b\}$. Note that by symmetry, the same holds for a in place of y. Let P_y be a path in $G - \{e, f\}$ from y to $\{s_a, s_b\}$, say s_a. By the minimality of T, P_y is nontrivial. If there exists some $z \in (P_y - s_a)$ (respectively, $R_a - s_a$) with a zs_b-path Q that is internally disjoint from P_y (respectively, R_a) then $(K \cup P_y \cup Q) + f$ (respectively, $(K \cup R_a \cup Q) + e$) is a subdivision of $K_{3,3}$ in $G - e$ $(G - f)$, a contradiction.

So, we may assume that s_a separates y (a) from $V(T)$ in $G - f$ $(G - e)$. If R_a is trivial (that is, $s_a = a$), then $\{a, x\}$ separates y from b in H. This implies that either e and f are in the same or adjacent branches of H, a contradiction. If R_a is not trivial, then $(G - \{e, f\}) - s_a$, and consequently $(H - \{e, f\}) - s_a$, has at least three components: one for each vertex in $\{y, a, x\}$. Assuming $H - \{e, f\}$ is connected, since otherwise e and f are in the same branch of H, the only way this can happen is if s_a is a node in H and e and f are in adjacent branches, a contradiction. □

References

[AR17] Alan Arroyo and R. Bruce Richter. "Characterizing graphs with crossing number at least 2." In: *J. of graph theory* 85.4 (2017), pp. 738–746.

[Moh94] Bojan Mohar. "Obstructions for the disk and the cylinder embedding extension problems." In: *Combin. probab. comput.* 3.3 (1994), pp. 375–406.

[Tut66] William Thomas Tutte. *Connectivity in Graphs.* Vol. 15. University of Toronto Press, 1966.

Rigidity of 1-coordinated frameworks in 2 dimensions

Bernd Schulze[1], Hattie Serocold[1], Louis Theran[2]

[1] Lancaster University, Lancaster, UK
b.schulze@lancaster.ac.uk, h.serocold@lancaster.ac.uk
[2] University of St Andrews, St Andrews, Scotland
louis.theran@st-andrews.ac.uk

Abstract

A bar-joint framework (G, p) is a graph G with an embedding of its vertices into $\mathbb{R}^d$. Coordinated frameworks are those with a subset of bars that may extend or retract, but must all do so at the same time, in contrast to standard frameworks in which the bar lengths are fixed by the embedding p. We wish to extend characterisations of standard frameworks to the coordinated context.

Keywords : *rigidity, infinitesimal rigidity, Henneberg construction*

1 Introduction

We extend the definition of a bar-joint framework to include frameworks where some collection of the bars are required to extend or contract in a coordinated manner. This could be considered as a set of pistons connected to a central pump, for example. It is natural to model this type of coordinated framework with an edge-coloured graph, wherein edges with the same colour form such a collection. The concept of a motion of a standard framework is similarly extended to contain motions that involve an extension or contraction of such a set of bars.

This extended abstract uses some basic results from geometric rigidity, and Henneberg-style inductive moves, to give a sparsity-based characterisation of 2-dimensional frameworks with one class of coordinated edges, along with an inductive construction of such frameworks.

2 Definitions

We give the following definitions as extensions of definitions from the existing rigidity literature.

Definition 1 *Let $G = (V, E)$ be a graph. We define the* edge-colouring function $c : E \to \{0, 1, \dots, k\}$*. We use $E_0 := c^{-1}(0)$ to denote the set of* uncoloured edges *of the graph G, and define the k* colour classes *to be the induced partitions $c^{-1}(\ell)$ for $1 \le \ell \le k$. We denote each colour class by $E_\ell := c^{-1}(\ell)$.*

We may then refer to (G, c) as a k edge-coloured graph*, and refer to edges in $E_1 \cup \cdots \cup E_\ell$ as the* coloured edges *of (G, c).*

Definition 2 *For a graph $G = (V, E)$ with k edge-colouring c, we define a* configuration *of G by $(p, r) \in \mathbb{R}^{d|V|+k}$, where $p \in \mathbb{R}^{d|V|}$ is such that $p(i) \neq p(j)$ for any $\{i, j\} \in E$ and $r \in \mathbb{R}^k$ is a vector representing the colour classes $1 \le \ell \le k$.*

*We refer to (G, c, p, r) as a k-*coordinated framework.

Definition 3 *If a k edge-coloured graph (G, c) has two potential placements (p, r) and (q, s), the k-coordinated frameworks (G, c, p, r) and (G, c, q, s) are considered to be* equivalent *if*

$$\|p(i) - p(j)\|^2 = \|q(i) - q(j)\|^2 \text{ for all } \{i, j\} \in E_0 \tag{1}$$

$$\|p(i) - p(j)\|^2 + r(\ell) = \|q(i) - q(j)\|^2 + s(\ell) \text{ for all } \{i, j\} \in E_\ell,\ \ell \in \{1, \dots, k\} \tag{2}$$

The k-coordinated frameworks (G, c, p, r) and (G, c, q, s) are congruent *if the frameworks are equivalent and the placements (p, r) and (q, s) are congruent.*

$$\|p(i) - p(j)\|^2 = \|q(i) - q(j)\|^2 \text{ for all } i, j \in V \tag{3}$$

Definition 4 *A k-coordinated framework (G, c, p, r) is* locally rigid *if there is a neighbourhood U of $(p, r) \in \mathbb{R}^{d|V|+k}$ such that, if $(q, s) \in U$ and the frameworks (G, c, p, r) and (G, c, q, s) are equivalent, then the frameworks (G, c, p, r) and (G, c, q, s) are congruent.*

If a framework is not locally rigid, we refer to it as being flexible.

It is equivalent for a framework (G, p) to be locally rigid, and for the only continuous motions of the framework (G, p) to be isometries of $\mathbb{R}^d$. We refer to the isometries of $\mathbb{R}^d$ as the *trivial motions* of any framework (G, c, p, r) in $\mathbb{R}^d$.

Rather than consider continuous motions of frameworks, we differentiate the constraints to linearise the problem. This allows us to consider infinitesimal motions, defined as follows.

In the infinitesimal case, we may often refer to a k-coordinated framework as (G, c, p).

Definition 5 *An* infinitesimal motion *of the k-coordinated d-dimensional framework (G, c, p) is $(p', r') \in \mathbb{R}^{d|V|+k}$, where $p' \in \mathbb{R}^{d|V|}$ is a velocity field supported on p and $r' \in \mathbb{R}^k$ is a vector, such that*

$$[p(i) - p(j)] \cdot [p'(i) - p'(j)] = 0 \text{ for all } \{i, j\} \in E_0 \tag{4}$$

$$[p(i) - p(j)] \cdot [p'(i) - p'(j)] + r'(\ell) = 0 \text{ for all } \{i, j\} \in E_\ell, \ell \in \{1, \dots, k\} \tag{5}$$

A trivial infinitesimal motion *(p', r') of a k-coordinated framework (G, c, p) will be $(p', \mathbf{0})$, where p' is an isometry of $\mathbb{R}^d$. These span a space of dimension $d + \binom{d}{2} = \binom{d+1}{2}$ and satisfy $[p(i) - p(j)] \cdot [p'(i) - p'(j)] = 0$ for all $i, j \in V$.*

Definition 6 *A k-coordinated framework (G, c, p) is considered to be* infinitesimally rigid *if all the infinitesimal motions of (G, c, p) are trivial infinitesimal motions.*

We define the coordinated rigidity matrix using the following notions, which is a natural extension of the standard rigidity matrix (see, for example, [4]).

Definition 7 *Let (G, c, p) be a k-coordinated framework. The* characteristic vector *of the colour class E_ℓ for $1 \leq \ell \leq k$ is denoted by $\mathbb{1}_\ell \in \mathbb{R}^{|E|}$, where $\mathbb{1}_\ell(e) = 1$ if $e \in E_\ell$ and $\mathbb{1}_\ell(e) = 0$ if $e \notin E_\ell$. The characteristic vectors for each colour class may be combined into the $|E|$ by k matrix $\mathbb{1}(c) := [\mathbb{1}_1, \dots, \mathbb{1}_k]$.*

The edge-length function *for a graph G is $f_G : \mathbb{R}^{d|V|} \to \mathbb{R}^{|E|}$, where $f_G(p)_{\{i,j\}} = \|p(i) - p(j)\|^2$ for a placement $p \in \mathbb{R}^{d|V|}$ with $p(i) \neq p(j)$ for any edge $\{i, j\} \in E$. We derive the $|E|$ by $d|V|$ matrix $R(G, p)$, referred to as the* rigidity matrix *of the d-dimensional framework (G, p), from the edge-length function by $df_G(p) = 2R(G, p)$. Each row of $R(G, p)$ will have $p(i) - p(j)$ in the d columns associated to the vertex i, and $p(j) - p(i)$ in the columns associated to the vertex j.*

The coordinated rigidity matrix *of the k-coordinated framework (G, c, p) is $R(G, c, p) := [R(G, p), \mathbb{1}(c)]$. This gives the following matrix condition for an infinitesimal motion (p', r'), equivalent to Equations (4) and (5):*

$$R(G, c, p)\,[p', r'] = \mathbf{0}. \tag{6}$$

We may in fact check whether or not a coordinated framework is infinitesimally rigid by checking whether the rank of $R(G, c, p)$ is $d|V| + k - \binom{d+1}{2}$. We state the following definition.

Definition 8 *A k-coordinated framework (G, c, p, r) is* isostatic in $\mathbb{R}^d$, *or* d-isostatic, *if it is infinitesimally rigid in $\mathbb{R}^d$, and the rows of $R(G, c, p)$ are independent.*

We may consider the infinitesimal rigidity of a framework (G, c, p, r) as being a property of the coloured graph (G, c) by restricting the class of placements that we consider to the following dense subset of $\mathbb{R}^{d|V|}$.

Definition 9 *A placement $(p, r) \in \mathbb{R}^{d|V|+k}$ of a k edge-coloured graph (G, c) is* regular *if* $\operatorname{rank} R(G, c, p) \geq \operatorname{rank} R(G, c, q)$ *for all $q \in \mathbb{R}^{d|V|}$.*

This leads to the following analogue to a standard result, given by Asimow and Roth [1].

Theorem 1 *Let (G, c, p) be a d-dimensional framework with a regular placement $p \in \mathbb{R}^{d|V|}$. Then (G, c, p) is rigid if and only if (G, c, p) is infinitesimally rigid.*

3 Main result

We wish to state an analogue to the standard result known as Laman's Theorem [3], for 2-dimensional frameworks with one class of coordinated edges. Laman characterises the class of graphs that are generically isostatic in 2 dimensions, and we use an extension of this class.

Definition 10 *A graph $G = (V, E)$ is referred to as a* (2,3)-tight graph*, or a* Laman graph*, if G has $|E| = 2|V| - 3$, with $|E'| \leq 2|V'| - 3$ for every subgraph $(V', E') \subset (V, E)$ with $|V'| \geq 2$.*

A graph $G = (V, E)$ is Laman-plus-one *if there exists an edge $e \in E$ such that $G - e$ is a Laman graph. A Laman-plus-one graph $G = (V, E)$ is a* rigidity circuit *if $G - e$ is a Laman graph for every edge $e \in E$.*

The proof method used by Laman [3] gives an inductive construction of Laman graphs, using 0-extensions and 1-extensions. We extend these constructions to our coordinated frameworks as follows, and note that such coloured extensions will preserve isostaticity.

Definition 11 *In 2 dimensions, the* 0-extension *is applied to a 1 edge-coloured graph (G, c) by creating a new vertex x, along with a pair of new edges $\{x, u_1\}, \{x, u_2\}$ for some $u_1, u_2 \in V$. When both edges are uncoloured, this is equivalent to the standard 0-extension used by Laman [3]. The edges may also both be coloured, or one may be coloured while the other is not.*

Definition 12 *In 2 dimensions, the* 1-extension *is applied to a 1 edge-coloured graph (G, c) by removing an edge $\{u_1, u_2\} \in E$, and replacing it with a new vertex x, along with three new edges, $\{x, u_1\}, \{x, u_2\}$, $\{x, u_3\}$ for some other vertex $u_3 \in V \setminus \{u_1, u_2\}$. If the edge $\{u_1, u_2\}$ is removed from E_0, we require that both of the edges $\{x, u_1\}, \{x, u_2\}$ be added to E_0, and the third edge may be added arbitrarily to either E_0 or E_1. When $\{x, u_3\}$ is also added to E_0, this is equivalent to the standard 1-extension. If instead $\{u_1, u_2\} \in E_1$, we require only that one of $\{x, u_1\}$ and $\{x, u_2\}$ be added to E_1. The other, and the third edge $\{x, u_3\}$, may be added to either E_0 or E_1 arbitrarily.*

These definitions allow us to state the following main result, which characterises the isostatic frameworks with one class of coordinated edges in 2 dimensions. We give a sketch of the proof below.

Theorem 2 *Let (G, c, p) be a 1-coordinated framework with a regular configuration $p \in \mathbb{R}^{2|V|}$. The following are equivalent:*

1. *(G, c, p) is an isostatic framework;*

2. *The graph G is Laman-plus-one, and at least one edge in E_1 is in the rigidity circuit of (G, c);*

3. *The graph G is Laman-plus-one, and $|D| \leq 2|V(D)| - 3$ for any $D \subseteq E_0$.*

4. *The edge-coloured graph (G, c) can be constructed from a copy of K_4 with at least one edge in E_1, by a sequence of coloured 0-extensions and 1-extensions.*

We note that the equivalence of the second and third conditions is straightforward. Proving that the third condition implies the fourth makes up the main body of the proof, and uses induction on the number of vertices of (G, c). To confirm that the fourth condition implies the first, it is straightforward to check that any regular coloured (K_4, c, p) with $|E_1| \geq 1$ is isostatic. As the coloured 0-extensions and 1-extensions preserve isostaticity of a 1-coloured framework, any framework constructed in this way will be isostatic.

We shall now prove that the first condition implies the third. If the graph (G, c) is not Laman-plus-one, we may remove any edge e from a rigidity circuit of G to create $G - e$, which is flexible as an uncoloured graph. The edge e will be replaced within a Laman subgraph of $G - e$, so the flex of $G - e$ will remain as a flex of G, and so (G, c) will be flexible. It is also clear from Laman's Theorem [3] that an uncoloured subgraph of (G, c) with $|D| > 2|V(D)| - 3$ cannot be independent, and so we require that $|D| \leq 2|V(D)| - 3$ for any $D \subseteq E_0$.

We apply induction on $|V|$ to prove that any framework satisfying the third condition may be constructed as described in the fourth condition. As the average degree within a Laman-plus-one graph is strictly less than four, there will be vertices of degree two or degree three. A vertex of degree two may be viewed as the result of a 0-extension, whether the adjacent edges are uncoloured or coloured, and will lie outside the rigidity circuit of (G, c). We may apply the reverse to a 0-extension to remove this vertex, resulting in a smaller graph containing an identical rigidity circuit, which will still satisfy the third condition.

If instead the minimum degree is three, and the Laman-plus-one graph is not a rigidity circuit, there will be a degree three vertex in the subgraph outside the rigidity circuit of (G, c). We may apply the reverse of a 1-extension at this vertex, as a vertex outside the rigidity circuit may have at most two neighbours within the rigidity circuit. The rigidity circuit of the reduced graph will contain the rigidity circuit of the larger graph.

If there are no vertices outside the rigidity circuit of (G, c), we consider the two cases of whether the rigidity circuit (G, c) is 2-connected or 3-connected. We apply results proved by Berg and Jordán [2] to prove that a coloured 3-connected circuit can be reduced to a smaller 3-connected circuit that still contains at least one coloured edge. When G is a 2-connected rigidity circuit, we identify a cut pair and a coloured edge $e \in E_1$, and apply the reverse of the 1-extension move to a degree three vertex on the other side of the cut pair. The rigidity circuit of the reduced graph will contain the whole subgraph on the same side of the cut pair as e, and so the rigidity circuit will still contain at least one coloured edge as required.

4 Further work

We have a characterisation of isostatic 2-dimensional frameworks with two classes of coordinated bars, with a similar structure to Theorem 2, along with characterisations of isostatic 1-dimensional frameworks with one and two classes of coordinated edges. Similar coloured sparsity conditions will clearly be necessary for any regular k-coordinated framework to be isostatic, though there may also be others. We aim to extend our coordinated class of frameworks to those including symmetry, which is currently work in progress.

References

[1] L. Asimow and B. Roth. *The rigidity of graphs I.* Trans. of the AMS, 245:279–289, 1978.

[2] A. R. Berg and T. Jordán. *A proof of Connelly's conjecture on 3-connected circuits of the rigidity matroid.* Journal of Combinatorial Theory, Series B, 88(1):77–97, 2003.

[3] G. Laman. *On graphs and rigidity of plane skeletal structures.* Journal of Engineering mathematics, 4(4):331–340, 1970.

[4] B. Schulze and W. Whiteley. *Rigidity and Scene Analysis, Handbook of Discrete and Computational Geometry, Third Edition*, CRC, 2017.

New advances on the branch-and-prune algorithm for the discretizable molecular distance geometry problem

C. Lavor[1], L. Mariano[2], M. Souza[3]

[1] University of Campinas, Brazil
clavor@ime.unicamp.br
[2] State University of Rio de Janeiro, Brazil
[3] Federal University of Ceará, Brazil

Abstract

The discretizable molecular distance geometry problem (DMDGP) is associated to protein structure calculations using data from Nuclear Magnetic Resonance (NMR) experiments, where the problem is how to calculate the 3D protein structure using NMR distance information. Protein geometry allow us to define atomic orders such that a discrete method, called Branch-and-Prune (BP), can be applied to the problem. We will present a new atomic order and discuss its theoretical properties that can be useful to model the uncertainties in NMR data.

Keywords : *distance geometry, protein conformation*

Extremal Graphs with respect to the Modified First Zagreb Connection Index

Guillaume Ducoffe[2,3], Ruxandra Marinescu-Ghemeci[1], Camelia Obreja[1] ,
Alexandru Popa[1,2], Rozica Maria Tache[1]

[1] University of Bucharest, Romania
[2] National Institute for Research and Development in Informatics, Romania
[3] The Research Institute of the University of Bucharest ICUB, Romania

Abstract

Topological indices (TIs) have an important role in studying properties of molecules. A main problem in mathematical chemistry is finding extreme graphs with respect to a given TI. In this paper extremal graphs with respect to the modified first Zagreb connection index for trees, unicyclic graphs with or without a fixed girth and connected graphs are determined. These graphs are relevant since they are chemical graphs.

Keywords : *topological index, Zagreb indices, trees, unicyclic graphs*

1 Introduction

Molecular descriptors are numerical values that characterize different properties of molecules. They are used mainly in the construction of Quantitative Structure-Activity Relationship (QSAR) or Quantitative Structure-Property Relationship (QSPR) models to generate predictions about the biological activity or physico-chemical properties of new untested or even yet to be synthetized substances. Such models have an accuracy up to 90% in predicting the effects of different medicines on some species of parasites which cause a large number of deaths or viruses known for their aggressivity, like Hepatitis B and C, Cytomegalovirus and HIV-1 ([4, 5]). Moreover, these models provides new drugs much more active than those already existing on the market ([2]).

A special family of molecular descriptors is represented by the topological indices (TIs), invariants that characterize the topology of a graph, indicating its ramification. TIs are mainly studied in connection with chemical graphs, which are undirected, connected graphs that model organic compounds by considering only the carbon-carbon and carbon-hydrogen type atomic bonds. Analysed in this context, the TIs appeared from the desire of studying structural properties of chemical compounds. Since the structures of substances, even modelled as graphs, cannot be readily investigated as such, numerical quantities associated with these structures have to be considered for successful studies in this area ([2, 4, 5]). Thus TIs participate, along with other molecular descriptors, in the construction of QSAR or QSPR models that generate predictions about the biological activity or physico-chemical properties of new substances. Consequently, more and more topological indices are being introduced in the literature with the purpose of being used mainly in applications within different areas of chemistry.

One of the most important and intensely studied TIs are the Zagreb indices, introduced in [6] from the desire to examine the dependence of the total π-electron energy on the molecular structure. They are defined by

$$M_1(G) = \sum_{v\in V(G)} d_G(v)^2 \qquad \text{and} \qquad M_2(G) = \sum_{(u,v)\in E(G)} d_G(u)d_G(v),$$

where $G = (V(G), E(G))$ is a connected graph and $d_G(v)$ denotes the degree of the vertex v.

Starting from these, a whole family of Zagreb indices appeared, including the multiplicative Zagreb indices ([8]), the reformulated Zagreb indices ([9]) and the modified Zagreb connection index ([6]).

This class of indices has been used to study molecular complexity, chirality, *ZE*-isomerism, heterosystems and has been used with good results in QSAR and QSPR models. Thus, mathematical chemistry experts note that the best descriptor models considered in the literature contains the Zagreb M_2-index. ([7])

An important direction in the field of mathematical chemistry is given by the determination of the extreme graphs with respect to a given TI. This field started with the leading article of Bollobás and Erdös ([3]), in which they determine the extremal graphs with respect to the Randić index, a well-known and intensely used in QSAR and QSPR models TI.

In this paper we study a modified Zagreb index introduced by Gutman and Trinajstić in [6], but studied for the first time in [1], and named the modified first Zagreb connection index:

$$ZC_1^*(G) = \sum_{v \in V(G)} d_G(v)\tau_G(v),$$

where $\tau_G(v)$ denotes the number of vertices at distance 2 from v, also called the connection number of v. The latter article determines the extremal graphs with respect to the $ZC_1^*(G)$-index in the class of chemical trees (in which every vertex has degree at most four). In this paper we extend the above study by considering general trees and unicyclic graphs with given or arbitrary girth, all of them representing classes of graphs of interest for the field of mathematical chemistry, due to their topology that closely resembles molecular structures.

The main method used in determining the extremal graphs is defining graph transformations that, under certain conditions, strictly increase/decrease the value of the topological index. This leads us to the extremal graph(s) by the following principle: whenever a graph is not yet of the desired shape, there is a way to apply one of the transformations mentioned above to yield a new graph; since the class of graphs under investigation is finite, this process is finite; since the process gives the same end result regardless of the graph it starts with, it exhibits the extremal graph(s) of the class in question.

We propose a new method with higher degree of generality with respect to the transformation techniques usually used in such context. It allows to find transformations that strictly increases the index by seeing the graph as a result of join operations of subgraphs and study the changes in the index when replacing a subgraph with another. This is useful, since various types of decomposition of graphs exists in literature, such as block decompositions.

2 Results on graph operations and transformations

Let G, H be two connected graphs and consider one vertex from each graph: $a \in V(G)$, $b \in V(H)$. Denote by $(G,a) \odot (H,b)$ the graph obtained from the union of graphs G and H by identifying vertices a and b. Note that if a graph has cut vertices, it can be seen as a vertex-joins of its biconnected components.

For a graph G denote by $d(G)$ the set of the degrees of its vertices and, for a vertex $v \in V(G)$, by $s_G(v)$ the sum of the degrees of its neighbors. In order to determine the ZC_1^*-index for graphs obtained from vertex-join operations, we define the following constants:

$$\epsilon_n = \begin{cases} 4, & \text{if } n \geq 5, \\ 2, & \text{if } n = 4, \\ 0, & \text{if } n = 3. \end{cases} \qquad \alpha_n = \begin{cases} 6, & \text{if } n \geq 5, \\ 5, & \text{if } n = 4, \\ 4, & \text{if } n = 3. \end{cases}$$

Lemma 1 *Let G, H be two connected graphs and two vertices $a \in V(G)$, $b \in V(H)$. Let $\Omega = (G,a) \odot (H,b)$. Then*

$$ZC_1^*(\Omega) = ZC_1^*(G) + ZC_1^*(H) + (s_G(a) + \tau_G(a))d_H(b) + d_G(a)(s_H(b) + \tau_H(b)).$$

Since for the classes of graphs considered in this paper are results of applying vertex-join operations on trees and cycles or of moving pendant edges from one vertex to another, we derive formulas for the case when one of the graphs is a path with one edge or a cycle.

Corollary 1 *Let G, H be two connected graphs and two vertices $a \in V(G)$, $b \in V(H)$. Let $\Omega = (G,a) \odot (H,b)$.*

1. *If $H \simeq P_2$, then $ZC_1^*(\Omega) = ZC_1^*(G) + d_G(a) + \tau_G(a) + s_G(a)$.*
2. *If $H \simeq C_n$, $n \geq 3$, then $ZC_1^*(\Omega) = ZC_1^*(G) + \epsilon_n n + \alpha_n d_G(a) + 2\tau_G(a) + 2s_G(a)$.*

We define the following transformation on a connected graph G. If G contains a pendant edge yx with y of degree at least 2 (non-pendant) and u is a vertex of G, $u \neq y$ and $u \neq x$, define transformation t_1 through which we obtain the graph

$$t_1(G, yx, u) := G - yx + ux$$

by removing the pendant edge yx from y and attaching it to vertex u.

A similar transformation can be defined by removing a cycle C from a vertex y and attaching it to a vertex u. We calculate the effect of these transformations on $ZC_1^*(G)$ by seeing G and the obtained graph as results of vertex-join operations.

Lemma 2 *Let G be a connected graph and t_1 the transformation defined above. Then*

$$ZC_1^*(t_1(G,yx,u)) - ZC_1^*(G) = 2(s_G(u) - s_G(y)) + 2(\tau_G(u) - \tau_G(y)) + 2(d_G(u) - d_G(y)) + 2\gamma_{G,y,u}$$

where

$$\gamma_{G,y,u} = \begin{cases} 0, & \text{if } u \text{ and } y \text{ are adjacent },\\ 1, & \text{otherwise.} \end{cases}$$

3 Trees

In [1] it is proven that, for $n \geq 5$, the path graph P_n has minimum ZC_1^* value among all trees with n vertices and this minimum value is $4n - 10$. In this section the trees with n vertices having maximum ZC_1^* value are determined. Let $\mathcal{S}_n$ be the family of trees with n vertices and diameter at most 3. Note that the only trees in $\mathcal{S}_n$ are the star and double stars.

Theorem 1 *The trees with $n \geq 4$ vertices having the maximum value of modified first Zagreb connection index are the trees in $\mathcal{S}_n$ and this value is $(n-2)(n-1)$.*

Proof : For a tree $T \notin \mathcal{S}_n$ there are two vertices u, y at distance at least 2 in which are incident pendant edges denoted such that $s_T(u) \geq s_T(y)$. Consider the tree $T' = t_1(T, yx, u)$. By Lemma 2 we obtain $ZC_1^*(T') - ZC_1^*(T) = 2(s_T(u) - s_T(y)) + 2 > 0$ □

4 Unicyclic graphs

A unicyclic graph is a connected graph containing exactly one cycle. Let $n \geq 5$ and $g < n$. In this section we consider first unicyclic graphs with n vertices and fixed girth g, that is having the length of the cycle equal to g.

Denote by $\mathcal{U}_{n,g}$ the family of unicyclic graphs with n vertices and girth g such that any two non-pendant vertices incident to pendant edges are adjacent vertices on the cycle.

Theorem 2 *Let $n \geq 5$ and $g < n$. The unicyclic graphs with n vertices and girth g having the maximum value of modified first Zagreb connection index are the graphs in $\mathcal{U}_{n,g}$ and this value is $(n-g)^2 + (\alpha_g + 1)(n-g) + \epsilon_g g$.*

Proof : Let $G \notin \mathcal{U}_{n,g}$. We construct a graph with higher index using previous results. More exactly, if there are in G two non-pendant vertices u, y both incident to at least one pendant edge, such that $d(u,y) \geq 2$ and $s_T(u) + d_T(u) + \tau_T(u) \geq s_T(y) + d_T(y) + \tau_T(y)$, we consider the graph $G' = t_1(G, yx, u)$. Otherwise the vertices of the cycle of G have degree 2, with one exception, a vertex denoted y. Then $G = (T_G, y) \odot (C_g, y)$ where T_G is a tree. In this case, we consider $G' = (S_{n-g}, a) \odot (C_g, y)$ where a is the center of star S_{n-g} (G' is obtained from G by replacing tree T_G rooted in y with star S_{n-g} rooted in the center). □

Let G be a unicyclic graph and C its unique cycle. Note that the connected components of $G - E(C)$ are trees rooted in vertices belonging to C. In order to obtain unicyclic graphs with smaller ZC_1^* index we study the changes of this index when replacing such a component with paths (with the same number of vertices) rooted in one extremity. Moreover, if one such component is a path, consider the graph obtained by attaching this component to a pendant vertex of G instead of a vertex from cycle C. The following result holds.

Theorem 3 *Let $n \geq 5$ and $g < n$. The unicyclic graph with n vertices and girth g having the minimum value of modified first Zagreb connection index is $T_{g,n-g} = (C_g, a) \odot (P_{n-g+1}, v)$, where v is a pendant vertex in P_{n-g+1} (the $(g, n-g)$-tadpole graph). Moreover,*

$$ZC_1^*(T_{g,n-g}) = \begin{cases} 4(n-g) + \epsilon_g g + \alpha_g, & \text{if } g \leq n-2, \\ \epsilon_g g + \alpha_g + 2, & \text{if } g = n-1. \end{cases}$$

We use the results on extremal unicyclic graph with fixed girth to study how the maximum and minimum value of ZC_1^* evolves when girth is variable, in particular to find the extremal unicyclic graphs with n vertices.

Theorem 4 *Let $n \geq 5$.*

1. *The unicyclic graphs with n vertices having the maximum value of the ZC_1^*-index are the graphs in $\mathcal{U}_{n,5}$ if $n < 8$, in $\mathcal{U}_{n,5} \cup \mathcal{U}_{n,3}$ if $n = 8$ and in $\mathcal{U}_{n,3}$ if $n > 8$.*
2. *The unicyclic graph with n vertices having the minimum value of the ZC_1^*-index is $T_{3,n-3}$.*

5 Connected graphs with n vertices

One important problem is to find the structure of an extremal connected graph with respect to ZC_1^*, having n vertices and no other constrains, that is to know the possible structure of molecules with n atoms that have minimum or maximum value of ZC_1^*.

Theorem 5 *Let $n \geq 5$.*

1. *The complete graph K_n is the unique graph that minimizes the index ZC_1^* among all n-vertex connected graphs.*
2. *A connected graph G with n vertices maximizes the index ZC_1^* among all n-vertex connected graphs if and only if it has diameter 2 and:*
   - *$d(G) \subseteq \{\frac{n}{2} - 1, \frac{n}{2}\}$, if $n \in \{4k, 4k+2 | k \in \mathbb{N}\}$*
   - *$d(G) = \{\frac{n-1}{2}\}$, if $n \in \{4k+1 | k \in \mathbb{N}\}$*
   - *there exists a vertex $x \in V(G)$ such that $d(G) \setminus \{d(x)\} = \{\frac{n-1}{2}\}$ and $d(x) \in \{\frac{n-1}{2} - 1, \frac{n-1}{2} + 1\}$, if $n \in \{4k+3 | k \in \mathbb{N}\}$*

References

[1] A. Ali and N. Trinajstić. *A novel/old modification of the first Zagreb index.* https://arxiv.org/pdf/1705.10430.pdf.

[2] A. T. Balaban and O. Ivanciuc. *Historical Development of Topological Indices.* Topological Indices and Related Descriptors in QSAR and QSPR, J. Devillers and A. T. Balaban (Eds.), Gordon and Breach Science Publisher, Amsterdam: 21–57, 1999.

[3] B. Bollobás and P. Erdös. *Graphs of extremal weights.* Ars Comb., 50: 225–233, 1998.

[4] I. Garcia, Y. Vall and G. Gomez. *Using Topological Indices to Predict Anti-Alzheimer and Anti-Parasitic GSK-3 Inhibitors by Multi-Target QSAR in Silico Screening.* Molecules, 15: 5408–5422, 2010.

[5] H. Gonzales-Diaz and C.R. Munteanu. *Topological indices for Medical Chemistry, Biology, Parasitology, Neurological and Social Networks.* Transworld Research Network, Kerala, 2010.

[6] I. Gutman and N. Trinastić. *Graph theory and molecular orbitals. Total π - electron energy of alternant hydrocarbons.* Chem Phys. Lett., 17: 535–538, 1972.

[7] S. Nikolić, G. Kovačević, A. Miličević and N. Trinajstić *The Zagreb Indices 30 Years After.* Croat. Chem. Acta, 76: 113–124, 2003.

[8] R. Todeschini and V. Consoni *New local vertex invariants and molecular descriptors based on functions of the vertex degrees.* MATCH, 64: 359–372, 2010.

[9] B. Zhou and N. Trinajstić *Some properties of the reformulated Zagreb indices.* J. Math. Chem., 48(3): 714–719, 2010.

A branch-and-price framework for decomposing graphs into relaxed cliques

Timo Gschwind[1], Stefan Irnich[1], Fabio Furini[2], Roberto Wolfler Calvo[3]

[1] Gutenberg School of Management and Economics, University of Maintz
gschwind@uni-mainz.de irnich@uni-mainz.de
[2] PSL, UniversitÃ© Paris-Dauphine, Paris, France,
fabio.furini@dauphine.fr
[3] Laboratoire d'Informatique de Paris Nord,
Université de Paris 13 and Sorbonne Paris Cité
CNRS (UMR 7538), 93430 Villetaneuse, France
wolfler@lipn.fr

Abstract

Relationships between objects can be modeled with graphs, where nodes represent the different objects and edges express the relationship. Social network analysis is an example where clusters, e.g., formed by members of a community, are studied using cliques and clique relaxations. A clique is a subgraph with pairwise directly connected nodes, i.e., a subgraph with diameter one. Several relaxations have been defined either in terms of distance (k-clique), degree (k-plex, k-core), diameter (k-club), or density. The majority of the literature deals with identifying such subgraphs of maximum cardinality or weight. In this presentation, we consider the generic problem of covering or partitioning a graph with a set of relaxed cliques. We present an exact solution fromework for solving the relaxed clique partitioning and covering problems based on branch-and-price. Herein, the subproblem consists of finding a relaxed clique of maximum weight. We present heuristics and a new combinatorial branch-and-bound algorithm for its resolution. The current state of the art in the field of complexity theory is a demonstrated lack of proof methods against problems still open. The combination of three separate results, called barriers (Relativisation, Natural Proofs and Algebrization), implies that none of the currently known proof methods for separation will successfully settle the remaining open problems. A single research program – Geometric Complexity Theory (GCT) – is considered viable by the community. However, according to its initiator and major contributor K. Mulmuley, GCT will not provide new results within our lifetimes; recent results have moreover closed the easiest path to GCT. As a consequence, complexity theory is in dire need of new tools and methods as such advances should require "fundamentally new methods" to paraphrase S. Aaronson and A. Widgerson. This talk will be about how such methods may be founded upon some recent developments in logic, and more precisely some specific models of proofs introduced under the name "Interaction Graphs".

The interplay between logic and computational complexity has been the subject of research for more than 50 years, but it has arguably failed to provide insights on the classification problem. Nevertheless, it has shown how logic is tightly bound to computation, clearly circumscribing the limits of the different approaches. The framework of Interaction Graphs, although taking its roots in logic, offers a mathematical model of programs that bypasses these limits and accounts for subtle aspects of computation. Moreover, it unveils deep connections with methods from geometry and dynamical systems that one may hope to exploit to enable potent proof methods from mathematics to be used by researchers against open problems in complexity theory.

A characterization for binary signed-graphic matroids

Konstantinos Papalamprou, Leonidas Pitsoulis, Eleni-Maria Vretta
School of Electrical and Computer Engineering,
Aristotle University of Thessaloniki, Thessaloniki, Greece
papalamprou@auth.gr, pitsouli@auth.gr, emvretta@auth.gr

Abstract

Binary matroids can be represented by $\{0,1\}$-matrices, while signed-graphic matroids can be represented by signed graphs. We provide a characterization for the class of binary signed-graphic matroids, generalizing a known result for graphic matroids. The latter characterization constitutes the first step towards the construction of an algorithm for recognizing binary signed-graphic matroids.

Keywords : *algorithm, binary matroids, signed-graphic matroids, tangled signed graphs, totally unimodular matrices*

1 Introduction

The natural connection of matroids with Combinatorial Optimization is established via their algorithmic definition[1], since matroids are precisely the structures for which the Greedy algorithm works. Furthermore, there are several results of Matroid Theory that have significant consequences for Combinatorial Optimization. One of them is the Regular Matroid Decomposition Theorem, which resulted in a polynomial time algorithm for recognizing totally unimodular (TU) matrices [3]. The latter class of matrices plays a central role in Combinatorial Optimization, since it defines a class of integer programs that are solved in polynomial time.

In this work, we present some results on signed graphs and matrices and prove a characterization for binary signed-graphic and nongraphic matroids. Moreover, we characterize the class of binary signed-graphic matroids, generalizing a known characterization for graphic matroids. The extended abstract is structured as follows. In section 2, some definitions and notations are given. In section 3, we introduce the main tools and present the results as well as some future perspectives for the utility of the characterization.

2 Preliminaries

A *graph* $G = (V, E)$ consists of a finite set of vertices V and a set of edges $E \subseteq V \cup V^2$. We shall denote by $V(G)$ and $E(G)$ the vertex set and edge set of a graph G, respectively. Given two distinct vertices u, v in a graph G we define four types of edges: $e = \{u, v\}$ is called a link, $e = \{v, v\}$ a loop, $e = \{v\}$ a half-edge, while $e = \emptyset$ is a loose-edge. Each edge has a set of one, none or two vertices associated with it, which are called its *endvertices*.

[1]An *independence system* $(E, \mathcal{I})$ is a pair of a finite set E and a collection $\mathcal{I}$ of subsets of E closed under inclusion. An independence system $(E, \mathcal{I})$ is a matroid if and only if the Greedy algorithm is optimal for the maximum-weight problem associated with $(E, \mathcal{I})$.

This research was implemented with a scholarship from the State Scholarships Foundation of Greece which was funded by the «Scholarship Program for Postgraduate Studies of the Second Cycle Studies» with resources from the Operational Program «Human Resources Development, Education and Lifelong Learning» 2014-2020 which was co-financed by the European Social Fund and the Greek State.

A *signed graph* $\Sigma = (G, \sigma)$ is a graph $G = (V, E)$ together with a sign function $\sigma : E(G) \to \{+1, -1\}$ such that $\sigma(e) = -1$ if e is a half-edge and $\sigma(e) = +1$ if e is a loose-edge. The graph G is called the underlying graph of Σ. We shall denote by $V(\Sigma)$ and $E(\Sigma)$ the set of vertices and the set of edges of a signed graph Σ, respectively. We consider a graph G to be a signed graph whose edges are all positive. Thus, signed graphs constitute a generalization of graphs. A signed graph is *connected* if and only if its underlying graph is connected. Any operation or term on signed graphs is defined via a corresponding operation or term on the underlying graph and the sign function. The *incidence* matrix of a signed graph $\Sigma = (G, \sigma)$ is a $|V(G)| \times |E(G)|$ matrix $A_\Sigma \in GF(3)$ with columns $a_e = (\alpha_{ie})_{i \in V(G)}$ for each $e \in E(\Sigma)$ defined as follows: $\alpha_{ue} = -\alpha_{ve}$ if $e = \{u, v\}$ is a positive link, $\alpha_{ue} = \alpha_{ve}$ if $e = \{u, v\}$ is a negative link, $\alpha_{ue} = 1$ or -1 if $e = \{u\}$ is a half-edge or $e = \{u, u\}$ is a negative loop, $\alpha_{ve} = 0$ if $e = \{u, u\}$ is a positive loop and $\alpha_{ie} = 0$ if $i \neq u, v$ [2].

In a signed graph, each walk $W = e_1, e_2, \dots, e_n$ has a sign $\sigma(W) := \sigma(e_1)\sigma(e_2)\dots\sigma(e_n)$. Therefore, a positive (resp. negative) cycle is a cycle that contains an even (resp. odd) number of negative edges. Negative loops and half-edges are considered negative cycles and are called *joints.* A signed graph without negative cycles is called *balanced*, otherwise it is called *unbalanced.* A vertex v in an unbalanced signed graph that belongs to every negative cycle is called a *balancing vertex.* An unbalanced signed graph is called *tangled*, if it has no balancing vertex and no two vertex-disjoint negative cycles. Given a signed graph Σ, if G is a tree, then Σ is a signed tree. *Negative* 1-*tree* of Σ is a signed tree with one more edge (link or joint) that forms a negative cycle with the signed tree. A connected subgraph of Σ consisting of a negative cycle and a path that has exactly one common vertex with the cycle is called *negative* 1-*path.* A signed graph whose connected components are either negative 1-trees or signed trees is called 1-*forest.*

Matroids have several equivalent axiomatic definitions; the following is in terms of circuits. A *matroid* M is an ordered pair $(E, \mathcal{C})$ of a finite set E and a collection $\mathcal{C}$ of subsets of E satisfying the following three conditions: (i) $\emptyset \notin \mathcal{C}$, (ii) If C_1 and C_2 are members of $\mathcal{C}$ and $C_1 \subseteq C_2$, then $C_1 = C_2$ (iii) If C_1, C_2 are distinct members of $\mathcal{C}$ and $e \in C_1 \cap C_2$, then there is a $C_3 \in \mathcal{C}$ such that $C_3 \subseteq (C_1 \cup C_2) - e$. The set E is called the ground set of M, denoted also by $E(M)$, while $\mathcal{C}$ is the family of circuits of M. The independent sets of a matroid M are all the subsets of E that do not contain a circuit $C \in \mathcal{C}$. A maximal independent set in M is a *basis* of M and is denoted by B. Given a basis B of a matroid M, for each $f \in E(M) - B$, we define $P_f \subseteq B$ such that $C_f = P_f \cup \{f\}$ is the unique circuit contained in $B \cup \{f\}$, that is C_f is the *fundamental circuit* of f with respect to B. A matroid M is *connected* if and only if any two distinct elements of $E(M)$ belong to a circuit. Two matroids M_1 and M_2 are *isomorphic* and we write $M_1 \cong M_2$, if there is a bijection $\phi : E(M_1) \to E(M_2)$ such that $X \in \mathcal{C}(M_1)$ if and only if $\phi(X) \in \mathcal{C}(M_2)$.

A major class of matroids, the class of $\mathbb{F}$-representable matroids, where $\mathbb{F}$ is some finite field, arises from matrices in the following way. Let E be the set of columns of a $m \times n$ matrix A in some field $\mathbb{F}$ and let $\mathcal{C}$ be the collection of all subsets of E which are minimal linearly dependent sets of vectors in the vector space $V(m, \mathbb{F})$; then $M = (E, \mathcal{C})$ is a matroid called vector matroid, denoted by $M[A]$. Any matroid isomorphic to $M[A]$ is called $\mathbb{F}$-*representable* matroid. Matroids that are representable over the finite fields $GF(2)$ and $GF(3)$ are called binary and ternary, respectively. The elements of $GF(3)$ are written as $0, 1, -1$. Furthermore, a matroid that is representable over every field is called *regular.* Equivalently, a regular matroid is one that can be represented by a TU matrix, the latter being a matrix over $\mathbb{R}$ for which every square submatrix has determinant in $\{0, 1, -1\}$.

The class of signed-graphic matroids and the class of graphic matroids are two fundamental classes of matroids that derive from signed graphs and graphs, respectively as follows. Let $E(\Sigma)$ be the set of edges of a signed graph Σ and $\mathcal{C}$ be the family of minimal edgesets inducing a subgraph that is either: (a) a positive cycle, or (b) two negative cycles which have exactly one common vertex, or (c) two vertex-disjoint negative cycles connected by a path, that has no common vertex with the cycles apart from its endvertices. Then $M(\Sigma) = (E(\Sigma), \mathcal{C})$ is a matroid on $E(\Sigma)$ with circuit family $\mathcal{C}$ and it is called the *signed-graphic* matroid of Σ. The connected subgraphs of Σ which are described in cases (b) and (c) in the above definition are called type I and type II handcuff, respectively. If, instead of a signed graph Σ, we have a graph G and the collection $\mathcal{C} \subseteq 2^{E(\Sigma)}$ of edgesets of cycles of G, then $M(G) = (E(G), \mathcal{C})$ is a matroid on $E(G)$ with circuit family $\mathcal{C}$, which is called the cycle matroid of G. A matroid that is isomorphic to the cycle matroid of a graph is called *graphic*.

3 Results

In this section, we present the main tools and the results of the work. Given a subset A of $E(M(\Sigma))$, where Σ is a signed graph and $M(\Sigma)$ is the associated signed-graphic matroid, then A corresponds to a subset of edges in Σ. Frequently, we shall refer to the induced subgraph of the edgeset of A in Σ, instead of the edgeset of A. By [[5], Theorem 5.1], every basis of a signed-graphic matroid $M(\Sigma)$ is a spanning 1-forest in a signed graph Σ. Hence the following two propositions are immediate consequences of the definition of a connected signed graph and the definition of a tangled signed graph. We note that a tangled signed graph is jointless.

Proposition 1 *If Σ is a connected tangled signed graph then every basis of $M(\Sigma)$ is a spanning negative 1-tree in Σ.*

Proposition 2 *If Σ is a connected tangled signed graph then there is no circuit of $M(\Sigma)$ which is a type II handcuff in Σ.*

Let M be a matroid represented by a TU matrix D. If D is viewed to be a matrix over an arbitrary field $\mathbb{F}$, then M is represented by D over $\mathbb{F}$. The *binary support* of a matrix D, denoted by $B_S(D)$, is defined to be a matrix which is obtained from D by replacing each non-zero entry of D by a 1. Since regular and signed-graphic matroids are $GF(3)$-representable, we use the following two propositions from [1] in the proof of Proposition 5.

Proposition 3 ([1], Proposition 6.4.1) *Let $[I_r|D_1]$ and $[I_r|D_2]$ be matrices over the fields $\mathbb{K}_1$ and $\mathbb{K}_2$ with the columns of each labelled, in order, by $e_1, e_2, \ldots, e_n$. If the identity map on $\{e_1, e_2, \ldots, e_n\}$ is an isomorphism from $M[I_r|D_1]$ to $M[I_r|D_2]$, then $B_S(D_1) = B_S(D_2)$.*

Proposition 4 ([1], Proposition 10.2.4) *Let D_1 and D_2 be $\{0, \pm 1\}$-matrices. If $B_S(D_1) = B_S(D_2)$ and $[I_r|D_1]$ and $[I_r|D_2]$ represent the same matroid over $GF(3)$, then D_2 can be obtained from D_1 by multiplying some rows and columns by -1.*

The incidence matrix A_Σ of a signed graph Σ is a representation of the signed-graphic matroid $M(\Sigma)$ over $GF(3)$ [2]. Furthermore, total unimodularity is maintained when we apply the following operations to a TU matrix: (1) pivots (2) row and column interchanges (3) scalings of rows or columns by -1. On combining the above, we obtain the following proposition, which highlights the connection between regular signed-graphic matroids and TU matrices.

Proposition 5 *Let $B = \{e_1, e_2, \ldots, e_r\}$ be a basis of a connected signed-graphic and non-graphic matroid $M(\Sigma)$, then $M(\Sigma)$ is regular if and only if the incidence matrix of the signed graph Σ is TU.*

Proof : Let us assume first that $M(\Sigma)$ is a regular matroid, then by definition there is a TU matrix A such that $M(\Sigma) \cong M[A]$. By a sequence of pivots and row and column interchanges, A can be transformed into a TU matrix of the form $[I_r|D_1]$, where the first r columns are labelled $e_1, e_2, \ldots, e_r$. Therefore $[I_r|D_1]$ has $\{0, \pm 1\}$ entries and it represents $M(\Sigma)$ over $GF(3)$. Since the incidence matrix A_Σ of Σ is over $GF(3)$, by applying pivots, row and column scalings to A_Σ, it can be transformed into $[I_r|D_2]$ in which the first r columns are labelled $e_1, e_2, \ldots, e_r$. By Proposition 3, it follows that $B_S(D_1) = B_S(D_2)$. Moreover, combining the fact that D_1 is TU with Proposition 4, we have that D_2 is TU.

For the converse, the incidence matrix A_Σ of Σ is TU and therefore, it is a representation matrix of $M(\Sigma)$ over $\mathbb{R}$. $\square$

Proposition 6 *If B is a basis of a binary matroid M, then M is graphic if and only if there is a tree T with $E(T) = B$ such that each of the sets $(P_f : f \in E(M) - B)$ is a path in T.*

The following theorem characterizes the class of binary signed-graphic and nongraphic matroids. Our main tools for its proof are matrices and signed graphs that represent binary and signed-graphic matroids.

Theorem 1 *Let B be a basis of a connected binary matroid M, then M is signed-graphic and nongraphic if and only if*

1. *there is a negative 1-tree T with negative cycle C_T, that is not a half-edge and $E(T) = B$ such that each of the sets $(P_f : f \in E(M) - B)$ is either a path or a negative 1-path in T*
2. *the signed graph obtained from T by adding each $f \in E(M) - B$ as a link with endvertices the ends of P_f (resp. $P_f - C_T$) if P_f is a path (resp. negative 1-path) and with the same sign with P_f, has an incidence matrix that is TU.*

Proof : For the "if" part. Since M is a connected binary signed-graphic and nongraphic matroid, there is a connected and tangled signed-graph Σ such that $M = M(\Sigma)$ [[4], Theorem 1.4]. By Proposition 1, there is a negative 1-tree T_Σ in Σ such that $E(T_\Sigma) = B$ with negative cycle C_{T_Σ} that is not a half-edge. Moreover, by Proposition 2, the fundamental circuits of $M(\Sigma)$ with respect to B are either positive cycles or type I handcuffs in Σ. Therefore each set $(P_f : f \in E(\Sigma) - T_\Sigma)$ is either a path or a negative 1-path in T_Σ. Furthermore the incidence matrix of Σ is TU by Proposition 5, since $M(\Sigma)$ is regular.

For the "only if" part. Let us assume that there is a negative 1-tree T with negative cycle C_T that is not a joint and it holds that $E(T) = B$. Moreover, for each $f \in E(M) - E(T)$ the set P_f is either a path or a negative 1-path in T. We construct a signed graph Σ from T by adding each $f \in E(M) - E(T)$ to T in the following manner. Suppose that P_f is a path in T, then we add edge f to T as a link with endvertices the two ends of P_f. Furthermore we attribute to f the sign of the path P_f. Therefore $P_f \cup \{f\}$ is a positive cycle in $T \cup \{f\}$. Suppose that P_f is a negative 1-path in T, then we add edge f to T as a link with endvertices the two ends of the path $P_f - C_T$. Furthermore we assign to f the opposite sign of the path $P_f - C_T$. Thereby $(P_f - C_T) \cup \{f\}$ is a negative cycle and $P_f \cup \{f\}$ is a type I handcuff in $T \cup \{f\}$. By assumption the incidence matrix of Σ is TU, which implies that $M(\Sigma)$ is regular and therefore, $M(\Sigma)$ is binary. Since both of M and $M(\Sigma)$ are binary and the fundamental circuits of $M(\Sigma)$ and M coincide with respect to the basis B, it follows that $M \cong M(\Sigma)$. □

On combining Theorem 1 and the proposition preceding it, we obtain the following characterization for the class of binary signed-graphic matroids. Given a basis B of a binary matroid M, the existence of a tree or a negative 1-tree T with $E(T) = B$ such that each set $(P_f : f \in E(M) - B)$ is either a path or a negative 1-path in T and (ii) of Theorem 1, are necessary and sufficient conditions for M to be signed-graphic. As a future step, it is desirable to obtain a polynomial time algorithm, which determines whether a given binary matroid is signed-graphic and returns also the graphical representation in the affirmative.

References

[1] J. Oxley. *Matroid Theory 2nd edition.* Oxford University Press, Oxford, 2011.

[2] L. Pitsoulis. *Topics in Matroid Theory.* Springer, 2014.

[3] P. Seymour. *Decomposition of regular matroids.* Journal of Combinatorial Theory Series, B 28 305-359, 1980

[4] D. Slilaty and H. Qin. *Decompositions of signed-graphic matroids.* Discrete Mathematics, 307 (17-18), 2187–2199, 2007.

[5] T. Zaslavsky. *Signed graphs.* Discrete Applied Mathematics 4, 47–74, 1982.

A Characterization of Interval Orders with Semiorder Dimension Two

Alexander Apke[1], Rainer Schrader[1]

University of Cologne, Germany

Abstract

Given a partial order Q, its semiorder dimension is the smallest number of semiorders whose intersection is Q. When we look at the class of partial orders of fixed semiorder dimension k, no characterization is known, even in the case $k = 2$. In this paper, we give a characterization of the class interval orders of semiorder dimension two.

Keywords : *partially ordered sets, interval order, semiorder, dimension*

1 Introduction

A partial order $Q = (V, \leq)$ is an *interval order* if

$$a < b,\ c < d \Rightarrow a < c \ \text{ or } \ b < d \qquad \forall\ a, b, c, d \in V.$$

Every interval order Q has a representation by intervals $I(v)$ on the real line, i.e. $u < v$ iff $I(u)$ lies completely to the left of $I(v)$. An interval order is a *semiorder* if it has a representation where all intervals have length 1.

Given a class $\mathcal{K}$ of orders, the *$\mathcal{K}$-dimension* of a partial order P is the smallest number k such that P is the intersection of k orders in $\mathcal{K}$. This concept was introduced by Dushnik and Miller [5] as (linear) dimension for the class of linear orders. Dagan et al. [4] introduced the class of *trapezoid graphs* and observe that they are exactly the co-comparability graphs of partial orders with *interval dimension* two. Langley [8] and Ma and Spinrad [9] proved that partial orders of interval dimension two and the associated class of trapezoid graphs can be recognized in polynomial time. Rabinovich [12] and Bogart et al. [2], [3] investigate the *semiorder dimension* and the associated class of co-comparability graphs. Since then, several related notions of order dimension and associated graph classes have been investigated (see, for example, [6], [7], [10], [11], [13]).

For partial orders of a fixed semiorder dimension k no characterization is known, even in the case that $k = 2$. In this paper, we give a characterization of the class interval orders of semiorder dimension two. It follows by our results that partial orders that are interval orders of semiorder dimension two can be recognized efficiently.

2 Preliminaries

Let C_l be a chain of length l. Given two chains C_l and C'_l such that no two elements $u \in C_l$ and $v \in C'_l$ are comparable, we denote their union by $C_{l,l'}$. We call $C_{3,1}$ the *claw order* since its complement induces a claw. Let $C' = (u, v, w, z)$ be a claw order with $u < v < w$. Then we call z the *center* of C', u the *tail of a claw with center z* or simply a *tail of z*, and w a *head of z*. Let $Z \subseteq V$ be the set of all centers of Q. In [1] we showed that for every interval order there is an interval representation such that $|I(z)| > 1$ for $z \in Z$ and $|I(v)| = 1$ for $v \in V \setminus Z$. We call $U = V \setminus Z$ the set of *unit elements* of Q.

We call $z, z' \in V$ *queued* if z is a tail of z' and z' is a head of z. Observe that if z and z' are queued then z is the center of a claw a leaf z' and vice versa.

For an interval order Q we introduce the undirected *queue graph* $G'(Q) = (Z, E)$. The set of vertices of G' corresponds to the set of centers in Q and we add an edge zz' to E if z and z' are queued.

3 Our Main Result

Theorem 1 *Let $Q = (V, \leq)$ be an interval order. Then, Q is of semiorder dimension two if and only if $G'(Q)$ is bipartite.*

For an interval order $Q = (V, \leq)$ the queue graph $G'(Q)$ can be constructed in time $O(|V|^2)$. Further, interval orders can be recognized in linear time. Since it can also be checked in linear time whether a graph is bipartite, it follows from Theorem 1 that interval orders of semiorder dimension two can be recognized in time $O(|V|^2)$.

References

[1] A. Apke and R. Schrader. On the non-unit count of interval graphs. *Discrete Appl. Math.*, 195(C):2–7, November 2015.

[2] Kenneth P. Bogart, Peter C. Fishburn, Garth Isaak, and Larry J. Langley. Proper and unit tolerance graphs. *Discrete Applied Mathematics*, 60(1 - 3):99 – 117, 1995.

[3] Kenneth P. Bogart, Rolf H. Möhring, and Stephen P. Ryan. Proper and unit trapezoid orders and graphs. *Order*, 15(3):325–340, 1998.

[4] Ido Dagan, Martin Charles Golumbic, and Ron Yair Pinter. Trapezoid graphs and their colorings. *Discrete Applied Mathematics*, 21(1):35 – 46, 1988.

[5] Ben Dushnik and Edwin W. Miller. Partially ordered sets. *American Journal of Mathematics*, 63(3):600 – 610, 1941.

[6] Peter C. Fishburn and William Trotter. Split semiorders. *Discrete Mathematics*, 195(1):111 – 126, 1998.

[7] Martin C. Golumbic and Clyde L. Monma. A generalization of interval graphs with tolerances. In *Proceedings of the 13th Southeastern Conference on Combinatorics, Graph Theory and Computing, Congressus Numerantion 35*, pages 321–331, 1982.

[8] Larry J. Langley. A recognition algorithm for orders of interval dimension two. *Discrete Applied Mathematics*, 60(1):257 – 266, 1995.

[9] T.H. Ma and J.P. Spinrad. On the 2-chain subgraph cover and related problems. *Journal of Algorithms*, 17(2):251 – 268, 1994.

[10] George B. Mertzios, Ignasi Sau, and Shmuel Zaks. A new intersection model and improved algorithms for tolerance graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1800 – 1813, 2009.

[11] George B. Mertzios, Ignasi Sau, and Shmuel Zaks. The recognition of tolerance and bounded tolerance graphs. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 585 – 596, 2010.

[12] I Rabinovitch. The dimension of semiorders. *Journal of Combinatorial Theory, Series A*, 25(1):50 – 61, 1978.

[13] Stephen P. Ryan. Trapezoid order classification. *Order*, 15(1):341 – 354, 1998.

Nash Equilibria in Mixed Stationary Strategies for m-Player Cyclic Games on Networks

Dmitrii Lozovanu[1], Stefan Pickl[2]

[1] Institute of Mathematics and Computer Science of Academy of Sciences of Moldova;
dmitrii.lozovanu@math.md

[2] Institute for Theoretical Computer Science, Mathematics and Operations Research,
Universität der Bundeswehr, Germany
stefan.pickl@unibw.de

Abstract

We consider non-zero cyclic games on networks with average payoffs and show that this class of games possesses Nash equilibria in mixed stationary strategies. An approach for determining the optimal strategies of the players in the considered games is proposed.

Keywords : *Cyclic game, average payoffs, Nash equilibrium, mixed stationary strategy*

1 Introduction and problem formulation

Let $G = (V, E)$ be a finite directed graph in which every vertex $u \in V$ has at least one leaving edge $e = (u, v) \in E$. The vertex set V of G is divided into m disjoint subsets $V_1, V_2, \dots, V_m$ ($V = V_1 \cup V_2 \cup \dots \cup V_m$; $V_i \cap Vj = \emptyset, i \neq j$) which we regard as positions sets of m players. Additionally, on edge set m functions $c^i : E \to R,\ i = 1, 2, \dots, m$ are given that assign to each directed edge $e = (u, v) \in E$ the values $c^1_e, c^2_e, \dots, c^m_e$ that we treat as the rewards for the corresponding players $1, 2, \dots, m$.

On G we consider the following m-person dynamic game. The game starts at given position $v_0 \in V$ at the moment of time $t = 0$ where the player $i \in \{1, 2, \dots, m\}$ who is owner of starting position v_0 makes a move from v_0 to a neighbor position $v_1 \in V$ through the directed edge $e_0 = (v_0, v_1) \in E$. After that players $1, 2, \dots, m$ receive the corresponding rewards $c^1_{e_0}, c^2_{e_0}, \dots, c^m_{e_0}$. Then at the moment of time $t = 1$ the player $k \in \{1, 2, \dots, m\}$ who is owner of position v_1 makes a move from v_1 to a position $v_2 \in V$ through the directed edge $e_1 = (v_1, v_2) \in E$, players $1, 2, \dots, m$ receive their rewards $c^1_{e_1}, c^2_{e_1}, \dots, c^m_{e_1}$, and so on, indefinitely. Such a play of the game on G produces the sequence of positions $v_0, v_1, v_2, \dots, v_t \dots$ where each v_t is the position at the moment of time t. In this game players make moves through the directed edges in their positions in order to maximize their average rewards per move

$$\omega^i_{v_o} = \lim_{t\to\infty} \inf \frac{1}{t} \sum_{\tau=0}^{t-1} c^i_{e_\tau}, \quad i = 1, 2, \dots, m.$$

If $m = 2$ and $c^1_e = -c^2_e = c_e,\ \forall e \in E$ then we obtain an antagonistic game with an average payoff. This case of the game has been studied in [2, 3] where the existence of the value of the game and the optimal pure stationary strategies of the players is proven. In [3] such an antagonistic game is called cyclic game with a mean payoff. In general, for a m-player cyclic game with average payoffs Nash equilibria in pure stationary strategies may not exist. This fact has been shown in [3] where an example of a two-player nonzero cyclic game that has no Nash equilibrium in pure strategies is constructed. A special class of m-player cyclic games for which Nash equilibria in pure stationary strategies exist is presented in [5, 6].

In this paper, we show that an arbitrary m-player cyclic game with mean payoffs possesses a Nash equilibrium in mixed stationary strategies. Based on a constructive proof of this result we propose an approach for determining the optimal mixed stationary strategies of the players.

2 Cyclic games in pure and mixed stationary strategies

A strategy of moves of player $i \in \{1, 2, \dots, m\}$ in a cyclic game is a mapping s^i that provides for every position $v_t \in V_i$ and every moment of time t a probability distribution over the set of moves $E(u_t) = \{e = (u_t, v) \in E | v \in V\}$. If these probabilities take only values 0 and 1, then s^i is called *a pure strategy*, otherwise s^i is called *a mixed strategy.* If these probabilities depend only on the state $u_t = u \in V_i$ (i. e. s^i do not depend on t), then s^i is called *a stationary strategy*, otherwise s^i is called a *non-stationary strategy.*

Denote by $V(u) = \{v \in V | (u, v) \in E\}$ the set of neighbor vertices for vertex u in G. Then the set of mixed stationary strategies $\mathbf{S}^i$ of player $i \in \{1, 2, \dots, m\}$ we can identify with the set of solutions of the system

$$\begin{cases} \sum\limits_{v \in V(u)} s^i_{u,v} = 1, & \forall u \in V_i; \\ s^i_{u,v} \geq 0, & \forall u \in V_i, \ \ \forall v \in V(u), \end{cases} \tag{1}$$

where each basic solution of system (1) correspond to a pure stationary strategy. For a mixed stationary strategy of player i the value $s^i_{u,v}$ for a given $u \in V_i$ and an arbitrary $v \in V$ can be treated as the probability that player i will make a move from $u \in V_i$ to $v \in V$ at every time when the position u is reached by any route in the dynamic game.

Let $\mathbf{s} = (s^1, s^2, \dots, s^m)$ be a profile of stationary strategies (pure or mixed strategies) of the players. Then the moves in the cyclic game from an arbitrary $u \in V$ to $v \in V$ induced by $\mathbf{s}$ are made according to probabilities of a stochastic matrix $P^s = (s_{u,v})$, where

$$s_{x,y} = \begin{cases} \mathbf{s}^i_{u,v} & \text{if } \ e = (u, v) \in E, u \in V_i, v \in V; \ \ i = 1, 2, \dots, m; \\ 0 & \text{if } \ e = (u, v) \notin E, \end{cases} \tag{2}$$

This means that for given $\mathbf{s}$ we obtain a Markov process with a probability transition matrix $P^{\mathbf{s}} = (\mathbf{s}_{x,y})$ and the corresponding rewards $c^i_{u,v}, i = 1, 2, \dots, m$ for $u, v \in E$. Therefore, if $Q^{\mathbf{s}} = (q^{\mathbf{s}}_{x,y})$ is the limiting probability matrix of $P^{\mathbf{s}}$ then the average rewards per transition $\omega^1_{v_0}(\mathbf{s}), \ \omega^2_{v_0}(\mathbf{s}), \dots, \omega^m_{v_0}(\mathbf{s})$ for the players can be determined as follows

$$\omega^i_{v_0}(\mathbf{s}) = \sum_{k=1}^{m} \sum_{u \in V_k} q^{\mathbf{s}}_{v_0,u} \mu^i(u, s^k), \quad i = 1, 2, \dots, m, \tag{3}$$

where

$$\mu^i(u, s^k) = \sum_{v \in V(u)} s^k_{u,v} c^i(u, v), \quad \text{for} \ \ u \in V_k, \ k \in \{1, 2, \dots, m\} \tag{4}$$

expresses the average step reward of player i in the state $u \in V_k$ when player k use the strategy s^k. The functions $\omega^1_{v_0}(\mathbf{s}), \ \omega^2_{v_0}(\mathbf{s}), \ \dots, \ \omega^m_{v_0}(\mathbf{s})$ on $\mathbf{S} = \mathbf{S}^1 \times \mathbf{S}^2 \times \dots \times \mathbf{S}^m$, defined according to (3),(4), determine a game in normal form that we denote by $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}}, \{\omega^i_{v_0}(\mathbf{s})\}_{i=\overline{1,m}} \rangle$. This game corresponds to the *cyclic game in mixed stationary strategies* on G.

In the paper we will consider also cyclic games in which the starting state is chosen randomly according to a given distribution $\{\theta_u\}$ on V. So, for a given stochastic positional game we will assume that the play starts in the states $u \in V$ with probabilities $\theta_u > 0$ where $\sum\limits_{u \in X} \theta_u = 1$. If the players use mixed stationary strategies of a selection of the actions in the states then the payoff functions

$$\psi^i_\theta(\mathbf{s}) = \sum_{x \in X} \theta_u \omega^i_u(\mathbf{s}), \quad i = 1, 2, \dots, m$$

on $\mathbf{S}$ define a game in normal form $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}}, \ \{\psi^i_\theta(\mathbf{s})\}_{i=\overline{1,m}} \rangle$. In the case $\theta_u = 0, \forall u \in V \setminus \{v_0\}$, $\theta_{v_0} = 1$ the considered game becomes a cyclic game with fixed starting state v_0.

3 Some auxiliary results

Let $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}},\ \{\psi^i(\mathbf{s})\}_{i=\overline{1,m}}\rangle$ be an m-player game in normal form, where $\mathbf{S}^i \subseteq \mathbf{R}^{n_i}$, $i = 1, 2, \ldots, m$ represent the corresponding sets of strategies of the players $1, 2, \ldots, m$ and $\psi^i : \prod\limits_{j=1}^{m} \mathbf{S}^j \to \mathbf{R}^1$, $i = 1,\ 2,\ \ldots,\ m$ represent the corresponding payoffs of these players. Additionally, let $\mathbf{s} = (s^1, s^2, \ldots, s^m)$ be a profile of strategies of the players where $\mathbf{s} \in \mathbf{S} = \prod\limits_{j=1}^{m} \mathbf{S}^j$, and define $\mathbf{s}^{-i} = (s^1, s^2, \ldots, s^{i-1}, s^{i+1}, \ldots, s^m)$, $\mathbf{S}^{-i} = \prod\limits_{j=1(j\neq i)}^{m} \mathbf{S}^i$ where $\mathbf{s}^{-i} \in \mathbf{S}^{-i}$. Thus, for an arbitrary $\mathbf{s} \in \mathbf{S}$ we can write $\mathbf{s} = (s^i, \mathbf{s}^{-i})$.

In [1] it is considered a class of games with upper semi-continuous, quasi-concave and graph-continuous payoffs. The payoff $\psi^i : \prod\limits_{j=1}^{m} \mathbf{S}^j \to \mathbf{R}^1$ of the game $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}},\ \{\psi^i(\mathbf{s})\}_{i=\overline{1,m}}\rangle$ is *graph-continuous* if for all $\overline{\mathbf{s}} = (\overline{s}^i, \overline{\mathbf{s}}^{-i}) \in \mathbf{S}$ there exists a function $F^i : \mathbf{S}^{-i} \to \mathbf{S}^i$ with $F^i(\overline{\mathbf{s}}^{-i}) = \overline{s}^i$ such that $\psi^i(F^i(\mathbf{s}^{-i}), \mathbf{s}^{-i})$ is continuous at $\mathbf{s}^{-i} = \overline{\mathbf{s}}^{-i}$.

In [1] it is proven the following theorem.

Theorem 1 *Let $\mathbf{S}^i \subseteq \mathbf{R}^{n_i}$, $i = 1, 2, \ldots, m$, be non-empty, convex and compact sets. If each payoff $\psi^i : \mathbf{S} \to \mathbf{R}^1, i \in \{1, 2, \ldots, m\}$, is quasi-concave with respect to s^i on $\mathbf{S}^i$, upper semi-continuous with respect to $\mathbf{s}$ on $\mathbf{S}$ and graph-continuous, then the game $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}},\ \{\psi^i(\mathbf{s})\}_{i=\overline{1,m}}\rangle$ possesses a Nash equilibrium.*

In the following we shall use this theorem for the case when each payoff $\psi^i(s^i, \mathbf{s}^{-i})$, $i \in \{1, 2, \ldots, m\}$ is quasi-monotonic with respect to s^i on $\mathbf{S}^i$ and graph-continuous. In this case the reaction correspondences of the players

$$\phi^i(\mathbf{s}^{-i}) = \{\hat{s}^i \in \mathbf{S}^i | f^i(\hat{s}^i, \mathbf{s}^{-i}) = \max_{s^i \in \mathbf{S}^i} \psi^i(s^i, \mathbf{s}^{-i})\}, \quad i = 1, 2, \ldots, m$$

are compact and convex valued and therefore the upper semi-continuous condition for the functions $f^i(\mathbf{s})$, $i = 1, 2, \ldots, m$ in Theorem 1 can be released. So, in this case the theorem can be formulated as follows.

Theorem 2 *Let $\mathbf{S}^i \subseteq \mathbf{R}^{n_i}$, $i = \overline{1,m}$ be non-empty, convex and compact sets. If each payoff $\psi^i : \mathbf{S} \to \mathbf{R}^1, i \in \{1, 2, \ldots, n\}$, is quasi-monotonic with respect to s^i on $\mathbf{S}^i$ and graph-continuous, then the game $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}},\ \{\psi^i(\mathbf{s})\}_{i=\overline{1,m}}\rangle$ possesses a Nash equilibrium.*

4 The main results

We have shown that a m-player cyclic game in mixed stationary strategies can be formulated as a game in normal form $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}},\ \{\psi^i_\theta(s)\}_{i=\overline{1,m}}\rangle$ where $\mathbf{S}^i$ and $\psi^i_\theta(s)$ for $i \in \{1, 2, \ldots, m\}$ are defined as follows: $\mathbf{S}^i$ represents a set of the solution of the system

$$\begin{cases} \sum\limits_{a \in A(x)} s^i_{u,v} = 1, \quad \forall x \in X_i; \\ \quad s^i_{u,v} \geq 0, \quad \forall u \in V_i,\ v \in V(u) \end{cases} \tag{5}$$

and

$$\psi^i_\theta(s^1, s^2, \ldots, s^m) = \sum_{k=1}^{m} \sum_{u \in V_k} \sum_{v \in V(u)} s^k_{u,v} c^i(u,v) q_u, \qquad i = 1, 2, \ldots, m, \tag{6}$$

where q_x for $x \in X$ are determined uniquely (via $s^k_{u,v}$) from the following system of equations

$$\begin{cases} q_v - \sum\limits_{k=1}^{m} \sum\limits_{u \in V_k} \sum\limits_{v \in V(u)} s^k_{u,v}\ q_u = 0, & \forall v \in V(u); \\ q_v + w_v - \sum\limits_{k=1}^{m} \sum\limits_{v \in V} \sum\limits_{a \in V(u)} s^k_{u,v}\ w_u = \theta_v, & \forall v \in V(u). \end{cases} \tag{7}$$

Here, θ_v for $v \in V$ represent arbitrary fixed positive values such that $\sum\limits_{v \in V} \theta_v = 1$.

For the game $\langle\{\mathbf{S}^i\}_{i=\overline{1,m}},\ \{\psi^i_\theta(s)\}_{i=\overline{1,m}}\rangle$, where $\mathbf{S}^i$ and $\psi^i_\theta(s)$ are defined according to (5)-(7), we proved the following two properties: each payoff $\psi^i_\theta(s)$ is quasi-monotonic with respect to s^i on $\mathbf{S}^i$; each payoff $\psi^i_\theta(s^i, \mathbf{s}^{-i}),\ \ i \in \{1, 2, \ldots, m\}$ is graph-continuous. The first property we derived from Theorem 4 is proven in [4] . The second property can be easily checked if for each payoff $\psi^i_\theta(s^i, \mathbf{s}^{-i}),\ \ i \in \{1, 2, \ldots, m\}$ we consider the function $F^i : \mathbf{S}^{-i} \to \mathbf{S}^i$ such that

$$F^i(\mathbf{s}^{-i}) = \hat{s}^i \in \phi^i(\mathbf{s}^{-i}) \ \text{ for } \ \mathbf{s}^{-i} \in \mathbf{S}^{-i},\ i \in \{1, 2, \ldots, m\}$$

where

$$\phi^i(\mathbf{s}^{-i}) = \{\hat{s}^i \in \mathbf{S}^i |\ \psi^i_\theta(\hat{s}^i, \mathbf{s}^{-i})) = \max_{s^i \in \mathbf{S}^i} \psi^i_\theta(s^i, \mathbf{s}^{-i})\}.$$

We can observe that $\psi^i_\theta(F^i(\mathbf{s}^{-i}), \mathbf{s}^{-i})$ is continuous at $\mathbf{s}^{-i} = \overline{\mathbf{s}}^{-i}$ for an arbitrary $(\overline{s}^i, \overline{\mathbf{s}}^{-i}) \in \mathbf{S}$ and consequently each payoff $\psi^i_\theta(\mathbf{s})$ is graph-continuous.

Thus, based on Theorem 2, we obtain the following result.

Theorem 3 *For an arbitrary cyclic game on G the corresponding game in normal form $\langle\{\mathbf{S}^i\}_{i=\overline{1,m}},\ \{\psi^i_\theta(s)\}_{i=\overline{1,m}}\rangle$ possesses a Nash equilibrium $\mathbf{s}^* = (s^{1*}, s^{2*}, \ldots, s^{m*}) \in \mathbf{S}$ which is a Nash equilibrium in mixed stationary strategies for the cyclic game on G with an arbitrary starting position $v_0 \in V$.*

So, the optimal mixed stationary strategies of the players in a cyclic game can be found if we determine the optimal stationary strategies of the players in the game $\langle\{\mathbf{S}^i\}_{i=\overline{1,m}},\ \{\psi^i_\theta(s)\}_{i=\overline{1,m}}\rangle$ where $\mathbf{S}^i$ and $\psi^i_\theta(s)$ for $i \in \{1, 2, \ldots, m\}$ are defined according to (5)-(7).

5 Conclusion and Perspectives

A Nash equilibrium in pure stationary strategies for a cyclic game with m players may not exist. However an arbitrary cyclic game with average payoffs possesses a Nash equilibrium in mixed stationary strategies. For an antagonistic cyclic game with an average payoff there exists an equilibrium in pure stationary strategies. The optimal mixed stationary strategies in a m-player cyclic game can be found using the game model from Section 4. The proposed approach for determining mixed stationary Nash equilibria in cyclic games can be extended for average stochastic positional games.

References

[1] Dasgupta P., Maskin E. *The existence of equilibrium in discontinuous economic games.* The Review of economic studies: 53, 1-26, 1986.

[2] Ehrenfeucht A., Mycielski J. *Positional strategies for mean payoff games.* Int. J. of Game Theory: 8, 109–113, 1979.

[3] Gurvich V., Karzaniv A., Khachyan L. *Cyclic games and an algorithm to find minimax mean cycles in directed graphs.* USSR Comput. Math. Phis.: 28, 5-91, 1988.

[4] Lozovanu D., Pickl S. *An approach for determining the optimal strategies for an average Markov decision problem with finite state and action spaces.*Bul. of ASM, ser. Math.: 1(86), 19-34, 2018.

[5] Lozovanu D., Pickl S. *Nash equilibria conditions for cyclic games with p players.* Electronic Notes in Discrete Mathematics: 25, 117-124, 2006.

[6] Lozovanu D., Pickl S. *Optimization of Stochastic Discrete Systems and Control on Complex Networks.* Springer Verlag, 2015.

On the One-Cop-Moves Game on Graphs

Boting Yang*

Department of Computer Science, University of Regina, Canada
boting@uregina.ca

Abstract

In this paper, we consider one-cop-moves game, in which the cop number of a graph G, denoted as $c_1(G)$, is the minimum number of cops required to capture a robber on G. We give a characterization of graphs with $c_1(G) \leq k$. We investigate the cop number of several classes of special graphs, including graphs with treewidth at most 2, Halin graphs, nested-wheel graphs, and Cartesian product graphs.

Keywords : *Graphs, Cops and Robbers, one-cop-moves game, lazy cops and robbers.*

1 Introduction

Quilliot [10] and Nowakowski and Winkler [8] introduced Cops and Robbers independently. In this paper, we study a variant of the Cops and Robbers game, known alternately as the *one-active-cop* game [9], *lazy cops and robbers* game [2, 3, 12] or the *one-cop-moves* game [13, 5]. The one-cop-moves game is played on a graph containing a set of cops and a robber, where the cops try to capture the robber. Both cops and the robber have perfect information, meaning that all of them know the whole graph and everyone's location at any moment. Initially, the cops choose a set of vertices to occupy; then the robber chooses a vertex to occupy. At even ticks of a clock (starting at 0), one of the cops moves to an adjacent vertex, and on odd ticks of the clock, the robber moves to an adjacent vertex or stays at his current vertex. Cops win if after some finite number of turns, one of the cops occupy the same vertex as the robber. This is called a *capture*. The robber wins if he can evade capture indefinitely. We want to find the minimum number of cops required to capture the robber in G. Such a number is called the *cop number* of G, denoted by $c_1(G)$.

The one-cop-moves cop number has been studied for various special families of graphs such as hypercubes [2, 9], generalized hypercubes [11], random graphs [3] and Rook's graphs [12]. On the other hand, relatively little is known about the behaviour of the one-cop-moves cop number of planar graphs. Aigner and Fromme [1] proved that for every connected planar graph, its cop number in the Cops and Robbers game is at most 3. Gao and Yang [5] showed that Aigner and Fromme's result does not generalize to the one-cop-moves game by constructing a connected planar graph whose structure is specifically designed for a robber to evade 3 cops indefnitely.

2 Main Results

For a graph G with vertex set $V(G)$ and edge set $E(G)$, we use $\{u, v\}$ to denote an edge with endpoints u and v. The vertex set $\{u : \{u, v\} \in E(G)\}$ is the *neighborhood* of v, denoted as $N_G(v)$, and the vertex set $N_G(v) \cup \{v\}$ is the *closed neighborhood* of v, denoted as $N_G[v]$. The *distance* between two vertices u, v, denoted by $\mathrm{dist}_G(u, v)$, is the length of the shortest path

*Research supported in part by an NSERC Discovery Research Grant, Application No.: RGPIN-2013-261290.

between u and v. The *distance* between a vertex v and a subset U of vertices in G, denoted by $\mathrm{dist}_G(v, U)$, is defined as $\mathrm{dist}_G(v, U) = \min\{\mathrm{dist}_G(v, u) : u \in U\}$.

2.1 Characterization of k-searchable graphs

If S is a collection of vertices of G in which vertices are allowed to appear more than once, then we say that S is a *multiset* of vertices. For a graph G and a positive integer k, we use $V_{G,k}$ to denote a multiset of vertices of G such that each vertex of G occurs exactly k times. It is easy to see that $V_{G,1} = V(G)$. The *cardinality* of S, denoted as $|S|$, is the total number of occurrences of vertices in S. So $|V_{G,k}| = k|V(G)|$. The notation of $S \subseteq V_{G,k}$ means that S is a multisubset of $V_{G,k}$ such that each vertex of S can occur at most k times. If $v \in S$, we use $S \setminus \{v\}$ to denote a multisubset of S such that the number of occurrences of v is reduced by one (if this number is reduced to 0, then v is removed from the multisubset). If $v \in V(G)$, we use $S \cup \{v\}$ to denote a multisuperset of S such that the number of occurrences of v is increased by one.

Let G be a finite connected graph and k be a positive integer. We define a relation $\mathbb{R}^0_{G,k}$ by letting $\mathbb{R}^0_{G,k} = \{(u, S) : u \in S \subseteq V_{G,k} \text{ and } |S| = k\}$. Assume that the relation $\mathbb{R}^i_{G,k}$ has been defined for each integer $i \in [0, t]$. Define $\mathbb{R}^{t+1}_{G,k}$ by saying $(u, S) \in \mathbb{R}^{t+1}_{G,k}$, where $u \in V(G)$, $S \subseteq V_{G,k}$ and $|S| = k$, if and only if for every $u' \in N[u]$, there are vertices $v \in S$ and $v' \in N[v]$ such that $(u', S') \in \mathbb{R}^i_{G,k}$ for some $i \in [0, t]$, where $S' = (S \setminus \{v\}) \cup \{v'\}$. The relation $\mathbb{R}^i_{G,k}$ has the following property.

Lemma 1 *For any $i \geq 0$, $\mathbb{R}^i_{G,k} \subseteq \mathbb{R}^{i+1}_{G,k}$.*

From the definition of the relation $\mathbb{R}^i_{G,k}$, we know that whenever we reach a t such that $\mathbb{R}^t_{G,k} = \mathbb{R}^{t-1}_{G,k}$, then $\mathbb{R}^i_{G,k} = \mathbb{R}^{t-1}_{G,k}$ for all $i \geq t$. From Lemma 1, there must be a smallest t for which $\mathbb{R}^t_{G,k} = \mathbb{R}^{t-1}_{G,k}$. We then define the relation $\mathbb{R}_{G,k}$ to be $\mathbb{R}^{t-1}_{G,k}$. Since the graph G is finite and $k \leq |V(G)|$, we know that $\mathbb{R}_{G,k}$ has only finitely many pairs (u, S), where $u \in V(G)$, $S \subseteq V_{G,k}$ and $|S| = k$.

The relation $\mathbb{R}_{G,k}$ is built recursively starting with the relation $\mathbb{R}^0_{G,k}$ as the initial step. In forming $\mathbb{R}^1_{G,k}$, we first include $\mathbb{R}^0_{G,k}$. Consider a pair (u, S) with $u \notin S$. In order for (u, S) to be in $\mathbb{R}^1_{G,k}$, for every vertex $u' \in N[u]$, we must find a vertex $v \in S$ and $v' \in N[v]$ such that $(u', S') \in \mathbb{R}^0_{G,k}$, where $S' = (S \setminus \{v\}) \cup \{v'\}$. However, the latter relation forces $u' \in S'$. Thus, every vertex in $N[u]$ must be in $N[S]$. In particular, $u \in N[S]$. Hence, $(u, S) \in \mathbb{R}^1_{G,k}$ implies $\mathrm{dist}_G(u, S) \leq 1$. In general, we have the following property.

Lemma 2 *For any $i \geq 0$, if $(u, S) \in \mathbb{R}^i_{G,k}$, then* $\mathrm{dist}_G(u, S) \leq i$.

Note that the relation $\mathbb{R}_{G,k}$ is a subset of the Cartesian product $V_{G,1} \times \{S : S \subseteq V_{G,k}$ and $|S| = k\}$. When the equality holds, the relation $\mathbb{R}_{G,k}$ is called *complete.* A graph G is *k-searchable* if $c_1(G) \leq k$, which is an analog of k-copwin in the Cops and Robbers game [6, 4].

Theorem 1 *For a connected graph G and a positive integer k, G is k-searchable if and only if the relation $\mathbb{R}_{G,k}$ is complete.*

2.2 Graphs with treewidth at most 2

The class of *two-terminal series-parallel graphs* are defined inductively as follows: (1) The graph with the single edge $\{s, t\}$ is a two-terminal series-parallel graph with terminals $\{s, t\}$. (2) If G_1 is a two-terminal series-parallel graph with terminals $\{s_1, t_1\}$, and G_2 is a two-terminal series-parallel graph with terminals $\{s_2, t_2\}$, then either identifying $t_1 = s_2$ to create a new two-terminal series-parallel graph with terminals $\{s_1, t_2\}$, or identifying $s = s_1 = s_2$ and $t = t_1 = t_2$ to create a new two-terminal series-parallel graph with terminals $\{s, t\}$.

Theorem 2 *If G is a two-terminal series-parallel graph, then $c_1(G) \leq 2$.*

Given a graph G, a *tree decomposition* of G is a pair (T, W) with a tree $T = (I, F)$, $I = \{1, 2, \ldots, m\}$, and a family of non-empty subsets $W = \{W_i \subseteq V : i = 1, 2, \ldots, m\}$, such that (1) $\bigcup_{i=1}^m W_i = V(G)$, (2) for each edge $\{u, v\} \in E(G)$, there is an $i \in I$ with $\{u, v\} \subseteq W_i$, and (3) for all $i, j, k \in I$, if j is on the path from i to k in T, then $W_i \bigcap W_k \subseteq W_j$.

The *width* of a tree decomposition (T, W) is $\max\{|W_i| - 1 : 1 \leq i \leq m\}$. The *treewidth* of G is the minimum width over all tree decompositions of G.

Theorem 3 *Let G be a connected graph with treewidth at most 2. Then $c_1(G) \leq 2$.*

Corollary 1 *For a connected graph G with treewidth 2, if G contains an induced cycle of length at least 4, then $c_1(G) = 2$.*

A graph is *outerplanar* if it has a plane embedding with all vertices on the exterior face.

Corollary 2 *For any connected outerplanar graph G, $c_1(G) \leq 2$.*

2.3 Halin graphs

A *Halin graph* is a plane graph constructed from a plane embedding of a tree with at least 4 vertices and with no vertices of degree 2, by connecting all leaves with a cycle in the natural cyclic order defined by the embedding of the tree.

Theorem 4 *For a Halin graph H, $c_1(H) \leq 3$.*

3 Nested-wheel graphs

We say that a cop *protects* a subgraph H of G if for any sequence of robber moves leading to the robber moving to a vertex of H, the robber is immediately captured by the cop.

Let C be a cycle in a graph G such that for any vertex v of G, there is a unique vertex f_v of C satisfying $\text{dist}_G(v, C) = \text{dist}_G(v, f_v)$. The vertex f_v is called the *foot* of v on C. If $\text{dist}_G(v, u) \geq \text{dist}_C(f_v, u)$, for any pair of vertices $v \in V(G)$, $u \in V(C)$, then we say that C is a *tight cycle* of G.

Lemma 3 *In the one-cop-moves game on a graph G with at least two cops, if C is a tight cycle of G, then after a finite number of turns, a single cop can protect C.*

Definition 1 Let G be a connected planar graph which has a plane embedding that contains a collection of vertex-disjoint nested cycles $\{C^1, \ldots, C^k\}$ such that $\cup_{i=1}^k V(C^i) = V(G)$, where C^1 may be a single vertex. If for each $i \in \{1, \ldots, k-1\}$, C^i is tight in the subgraph induced by $\cup_{j=i}^k V(C^j)$, then G is called a *nested-wheel graph.*

Note that cylinder grids are nested-wheel graphs.

Theorem 5 *If G is a nested-wheel graph, then $c_1(G) \leq 2$.*

Since cylinder grids are nested-wheel graphs, by Theorem 5, we have $c_1(C_m \square P_n) = 2$ for $m \geq 3$ and $n \geq 2$, where P_n is a path with n vertices and C_m is a cycle with m vertices. If we replace each cycle C^i in Definition 1 by a path, Theorem 5 still holds. So it follows that $c_1(P_m \square P_n) = 2$ for $m \geq 2$ and $n \geq 2$.

3.1 Cartesian product graphs

Let G_1 and G_2 be graphs. The *Cartesian product* of G_1 and G_2, denoted $G_1 \Box G_2$, has vertex set $V(G_1) \times V(G_2)$, where two vertices (u_1, u_2) and (v_1, v_2) are adjacent if and only if there exists j, $1 \leq j \leq 2$, such that $\{u_j, v_j\} \in E(G_j)$ and $u_i = v_i$ for $i \neq j$.

Maamoun and Meyiel [7] studied the cop number of Cartesian products of trees in the Cops and Robbers game. The next result directly follows from Lemma 1 in [7].

Lemma 4 *Let T_1 and T_2 be two trees, each of which contains at least one edge. Then $c_1(T_1 \Box T_2) = 2$.*

We now consider the Cartesian product of a tree T and a cycle C. Note that each subgraph $P \Box C$ is a nested-wheel graph, where P is a path in T. We can use the strategy described in the proof of Theorem 5 to show the following result.

Theorem 6 *Let T be a tree with at least two vertices and C be a cycle with at least three vertices. Then $c_1(T \Box C) = 2$.*

References

[1] M. Aigner, M. Fromme, A game of cops and robbers, *Discrete Applied Mathematics*, 8:1–12, 1984.

[2] D. Bal, A. Bonato, W. B. Kinnersley, P. Pralat. Lazy cops and robbers on hypercubes. *Combinatorics Probability and Computing* 24(6):829–837, 2015.

[3] D. Bal, A. Bonato, W. B. Kinnersley, P. Pralat. Lazy cops and robbers played on random graphs and graphs on surfaces. *International Journal of Combinatorics* 7(4):627–642, 2016.

[4] N.E. Clarke, G. MacGillivray, Characterizations of k-copwin graphs, *Discrete Mathematics*, 312:1421–1425, 2012.

[5] Z. Gao, B. Yang, The Cop Number of the One-Cop-Moves Game on Planar Graphs, Proceedings of the 11th International Conference on Combinatorial Optimization and Applications (COCOA'17), Lecture Notes in Computer Science, Vol. 10628, Springer, Berlin, pp.199–213, 2017.

[6] G. Hahn, G. MacGillivray, A note on k-cop, l-robber games on graphs, *Discrete Mathematics* 306:2492–2497, 2006.

[7] M. Maamoun, H. Meyniel, On a game of policemen and robber, *Discrete Applied Mathematics*, 17: 307–309, 1987.

[8] R.J. Nowakowski, P. Winkler, Vertex-to-vertex pursuit in a graph, *Discrete Mathematics*, 43:235–239, 1983.

[9] D. Offner, K. Okajian. Variations of Cops and Robber on the hypercube. *Australasian Journal of Combinatorics* 59(2):229–250, 2014.

[10] A. Quilliot, Jeux et pointes fixes sur les graphes, *Thèse de 3ème cycle*, Université de Paris VI, 131–145, 1978.

[11] K. A. Sim, T. S. Tan, K. B. Wong. Lazy cops and robbers on generalized hypercubes. *Discrete Mathematics* 340(2017):1693–1704.

[12] B. W. Sullivan, N. Townsend, M. Werzanski. The 3×3 rooks graph is the unique smallest graph with lazy cop number 3. Preprint. https://arxiv.org/abs/1606.08485.

[13] B. Yang, W. Hamilton. The optimal capture time of the one-cop-moves game. *Theoretical Computer Science* 588(2015):96–113.

Attacking the Clique Number of a Graph

Fabio Furini[1], Ivana Ljubić[2], Sébastien Martin[3], Pablo San Segundo[4]

[1] PSL, Université Paris-Dauphine, Paris, France,
`fabio.furini@dauphine.fr`

[2] ESSEC Business School, Cergy-Pontoise, France
`ivana.ljubic@essec.edu`

[3] LCOMS, Université de Lorraine, Metz, France
`sebastien.martin@univ-lorraine.fr`

[4] UPM, Universidad Politécnica de Madrid, Madrid, Spain,
`pablo.sansegundo@upm.es`

Abstract

We study the two player zero-sum Stackelberg game in which the leader interdicts (removes) a limited number of vertices from the graph, and the follower searches for the maximum clique in the interdicted graph. The goal of the leader is to derive an interdiction policy which will result in the worst possible outcome for the follower.

Keywords : *Network Interdiction, Maximum Clique, Social Network Analysis.*

1 Introduction and Problem Definition

Interdiction games on networks are a special family of two-player zero-sum Stackelberg games, in which the first player (a *leader*) decides which vertices or edges to *interdict* without violating a given interdiction budget, and after that the second player (a *follower*) solves a network optimization problem (e.g., the maximum clique, the shortest-path, the maximum flow) on the remaining network. The goal of the leader is to choose an *interdiction policy* that will guarantee the worst possible outcome for the follower. In this article we study the interdiction game of the maximum clique in a network.

In the context of game theory, this problem can be seen as the two-player *Clique Interdiction Game* (CIG), in which the leader has a limited budget of vertices to interdict in a network and the follower finds the largest clique in the remaining network.

Definition 1 (The Maximum Clique Interdiction Game (CIG)) *Given a graph G and an interdiction budget k ($k \geq 1$), the* maximum clique interdiction game *is to find a subset of at most k vertices to delete from G so that the size of the maximum clique in the remaining graph is minimized. The associated set of interdicted vertices is called the* optimal interdiction policy*.*

All the material presented here are not intended to appear in any publication and they refer to a submitted paper with preprints here: [1].

References

[1] F. Furini, I. Ljubic, S. Martin, and P. San Segundo. The maximum clique interdiction game. *Optimization Online*, 2018.

Multimodal transportation plan adjustment with passengers behaviour constraints

Pierre-Olivier Bauguion[1] Claudia D'Ambrosio[2]

[1] IRT SystemX, 8 Avenue de la Vauve, 91120 Palaiseau, France
pierre-olivier.bauguion@irt-systemx.fr

[2] CNRS LIX, Ecole Polytechnique, 91128 Palaiseau, France dambrosio@lix.polytechnique.fr

Keywords : *Transportation optimization, network flows, passengers behavior, supervision.*

1 Introduction

For the last decades, the growth of population in high density areas pressed the need to enhance mobility services. Nowadays, most of the biggest cities have settled public transportation infrastructures to fluidize traffic and ease local mobility. Nevertheless, these infrastructures are likely to meet their saturation limits during rush hours, making the management of such a system a crucial point. This led to consider Transportation Network Optimization Problems (TNOP).

One can divide TNOPs into two main parts, each part corresponding to a different aspect of the problem.

1.1 Network flows and user equilibrium

It is usually more convenient to model passengers as flows. The practical interest of flows lies in its ability to aggregate each individual while still capturing the movements on the network. Network flows have been introduced long ago (see, e.g.,[4], enhanced later by [2]). Since then, these models have been generalized for multiple "commodities" [10] and grouped under the label of "multi-commodity" flow models. A lot of optimization problems aim at deciding the best routing solution of flows –by minimizing a cost function for instance (see e.g. [1]), whereas in transportation models these flows are used to describe a user equilibrium, namely a Wardrop's equilibrium. [11] sums up equilibrium models by describing its two competitive mechanisms: a "selfish" disutility minimization for each individual on one hand (the objective), and the increase of disutility generated by confronting the strategies of those individuals on the other hand (the constraints). Hence, he shows how to reformulate it with mathematical programming. This will become the base principle from which every other passengers behaviour model will derive.

1.2 Network design and Bi-level Programming

Whereas network flows problems consists in using the network, the network design problem consists in deciding how the network should be. The latter can be the case when it comes to choose whether we want to install a network component or not, considering a cost function (see e.g. [5]). More generally the impact of such decisions (in terms of cost and/or feasibility) is not always easily computed, which is typically the case in TNOPs. It is typically the case in TNOPs, where we have to solve a network flows problem (as [11]) to know what are the consequences in our transportation plan decisions. These two embedded problems (network flows problem and network design problem) can be seen as hierarchical in the sense that the decisions of the transportation plan will set the available itineraries of passengers, but the other way around is not true. From the game theory point of view, this can be seen as Stackelberg competition, as the transport operator decides the transport plan first (and hence is the so-called "leader"), and given that decision, the passengers will decide how to react (they represent the so-called "followers"). However, the leader have insights about the way the follower will react to developp a relevant strategy. To model it as an optimization problem, Bi-level Programming is often used. Bi-level Programming means that two level of decisions exist: one that becomes parameters for the second. [9] proposed a large survey of different transportation Bi-level models and algorithm techniques to solve them, completed later by [3]. [8] then [7] tackles the stochastic versions of user equilibrium, especially for road networks.

1.3 Multimodal transportation plan adjustment with passengers behaviour constraints

With the rise of digital technology and telecommunications, transportation operators can simultaneously be closer to the real time state of transportation infrastructure and have larger vision of the whole (even multimodal) transportation system. This allows transport operators to react at a disturbance dynamically and disseminate quickly a recourse strategy to face it off. This is what we call multimodal supervision. From this point of view, we propose to address a problem of transportation plan readjustment. Readjustment differs from classical optimization by four main points. Firstly, the problem has a variable spatial and temporal perimeter, as we do not necessarily know in advance how long and how far the issue will propagate along the network(s). In other words, the space and time needed to go back to a nominal state is unknown. Secondly, for the previous reason (propagation avoidance), the adjustment model must be as accurate as possible during the horizon it is deployed. It includes passengers strategies and problems caused by their overloads (delays, door blocking...). This can induce interdependency between different transportation systems. Thus, the multimodal network must be considered. Thirdly, the optimization horizon starts with a given infrastructure state. This includes the positions of vehicles and passengers, and whole infrastructure state. Finally, adjustment implies that the solution is a differential plan from the nominal one. In terms of optimizing process, it can induce a set of specific constraints related to the distance with the proposed plan (adjustment) and the nominal one.

Moreover, as the situation is extremely dynamical, to ensure the strategy to be deployed at the time it is provided, the adjustment must be made in a short time-window. This problem can be formalized as a particular TNOP and modelled using (Time-Expanded or Time-Dependent) graphs (see e.g. [6, 12]). We discuss here a first mathematical formulation to tackle this problem in one single Mixed Integer Linear Program, and future methods to achieve these operational objectives.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications.* Prentice Hall Inc., 1993.

[2] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, **19**(2):248–264, 1972.

[3] R. Z. Farahani, E. Miandoabchi, W. Y. Szeto, and H. Rashidi. A review of urban transportation network design problems. *European Journal of Operational Research*, **229**(2):281–302, 2013.

[4] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[5] E. Gourdin, M. Labbé, and H. Yaman. Telecommunication and location. In Z. Drezner and H.W. Hamacher, editors, *Facility Location: Applications and Theory*, pages 275–305. Springer, 2002.

[6] C. Liebchen and L. Peeters. Some practical aspects of periodic timetabling. In *Operations Research Proceedings 2001*, pages 25–32, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[7] H. Liu and D. Z. W. Wang. Global optimization method for network design problem with stochastic user equilibrium. *Transportation Research Part B: Methodological*, 72:20–39, 2015.

[8] Q. Meng, D. H. Lee, H. Yang, and H. J. Huang. Transportation network optimization problems with stochastic user equilibrium constraints. *Transportation Research Record: Journal of the Transportation Research Board*, 1882:113–119, 2004.

[9] A. Migdalas. Bilevel programming in traffic planning: Models, methods and challenge. *Journal of Global Optimization*, **7**(4):381–405, 1995.

[10] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *Journal of the ACM*, **37**(2):318–334, 1990.

[11] Y. Sheffi. *Urban Transportation networks: Equilibrium Analysis with Mathematical Programming Methods.* Prentice Hall Inc., 1985.

[12] M. Skutella. *An Introduction to Network Flows over Time*, pages 451–482. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

Dynamic programming for the Electric Vehicle Orienteering Problem with multiple technologies

Dario Bezzi[1], Alberto Ceselli[1], Giovanni Righini[1]

Dept. of Computer Science, University of Milan, Italy

dario.bezzi,alberto.ceselli,giovanni.righini@unimi.it

Abstract

We describe a bi-directional dynamic programming algorithm to solve the Electric Vechile Orienteering Problem, arising as a pricing sub-problem in column generation algorithms for the Electric VRP with multiple recharge technologies.

Keywords : *Combinatorial optimization, dynamic programming, shortest path.*

1 Problem description

The Electric Vehicle Routing Problem (EVRP) has been introduced by Erdogan and Miller-Hooks under the name of Green Vehicle Routing Problem in [1]. Several variations have been studied, including problem with time windows, partial recharges, multiple technologies and both exact and heuristic algorithms have been developed. Examples of heuristic algorithms for the EVRP are given in Felipe et al. [2], Schneider et al. [3] and Koc and Karaoglan [4]. More references on VRP variants involving the use of electric vehicles can be found in a recent and extensive survey by Pelletier et al. [5].

The computation of exact solutions is more challenging than for the classical VRP, because of the additional subproblem of deciding the optimal recharges at some points along the routes. An additional source of complexity is the presence of different recharge technologies, each one characterized by a unit cost and a recharge speed. Schiffer and Walther [6] recently considered a similar problem in the context of location-routing. Sweda et al. [7] studied the optimal recharge policy when the route is given. As with many other variations of the VRP, the most common choice to design effective exact optimization algorithms is to rely upon branch-and-cut-and-price, starting from a reformulation of the routing problem as a set covering or set partitioning problem, where each column represents the duty of a vehicle. For instance, Desaulniers et al. [8] developed a branch-and-price-and-cut algorithm for the exact solution of the EVRP with time windows. In this study we investigate the Electric Vehicle Orienteering Problem, arising as a pricing sub-problem when the EVRP is solved by branch-and-price and in particular we consider a dynamic programming algorithm for the case with multiple technologies.

2 Formulation

Let $G = (N \cup R, E)$ be a given weighted undirected graph whose vertex set is the union of a set N of customers and a set R of recharge stations. A distinguished station in R is the depot, numbered 0. A fleet of V identical vehicles, located at the depot, must visit the customers. All customers in N must be visited by a single vehicle; split delivery is not allowed. Each customer $i \in N$ is characterized by a demand and each vehicle has a capacity as in the classical Capacitated VRP. Vehicles are equipped with batteries of given capacity B. Recharge stations can be visited at any time; multiple visits to them is allowed and partial recharge is also allowed. We consider a set of different technologies for battery recharge. For each technology we assume a given recharge speed. When visiting a station, only one of the available technologies can be used.

All vertices $i \in R \cup N$ are also characterized by a service time, representing the time taken by pick-up/delivery operations for $i \in N$ or a fixed time to be spent to set-up the recharge for $i \in R$. The distance d_e and the travel time are known for each edge $e \in E$. The energy consumption is assumed to be proportional to the distance through a given coefficient π. The duration of each route (including service time, travel time and recharge time) is required not to exceed a given limit.

A feasible route must comply with capacity and duration constraints. Furthermore the level of the battery charge must be kept between 0 and B at any time. A set of feasible routes is a feasible solution if all customers are visited once and no more than V vehicles are used. The objective to be optimized is given by the overall recharge cost, consisting of a fixed cost and a variable cost. Since batteries allow for a limited number of recharge cycles during their operational life, we associate a fixed cost f with each recharge operation. The variable cost associated with a recharge operation at any station $i \in R$ is proportional to the amount of energy recharged, but it also depends on the recharge technology.

We indicate with Ω the set of all feasible routes. We associate a binary variable x_r with each feasible route $r \in \Omega$: Binary coefficients y_{ir} take value 1 if and only if customer $i \in N$ is visited along route $r \in \Omega$. We indicate by c_r the cost of each route $r \in \Omega$. With these definitions and notation we obtain the following ILP model (master problem):

$$\text{minimize} \sum_{r\in\Omega} c_r x_r \tag{1}$$

$$\text{s.t.} \sum_{r\in\Omega} y_{ir} x_r \geq 1 \qquad \forall i \in N \tag{2}$$

$$\sum_{r\in\Omega} x_r \leq V \tag{3}$$

$$x_r \in \{0,1\} \qquad \forall r \in \Omega. \tag{4}$$

At each node of a branch-and-bound tree the linear relaxation of the master problem is solved by column generation. We indicate by λ_i the non-negative dual variables vector corresponding to the covering constraints (2) and by μ the scalar non-negative dual variable corresponding to constraints (3) restated in $\geq$ form. With this notation, the expression of the reduced cost of a generic column r is

$$\hat{c}_r = c_r - \sum_{i\in N} \lambda_i y_{ir} + \mu.$$

3 The pricing sub-problem

The pricing problem, whose ILP formulation is not reported here for brevity, is a variation of the Orienteering Problem and it requires to find a minimum cost closed walk from the depot to the depot, not visiting any customer vertex more than once and not consuming more than a given amount of available resources (capacity, time and energy). Edges between stations can be traversed more than once. This problem is also a variation of the Resource Constrained Elementary Shortest Path Problem, in which the elementary path constraints are imposed only on a subset of vertices, the resources are partly discrete and partly continuous and one of the resources (energy) is renewable.

3.1 The algorithm

We have devised an exact pricing algorithm based on dynamic programming, where labels are associated with paths emanating from the depot and have the following form:

$$L = (u, S, \phi, \underline{t}, \underline{\hat{c}}, \underline{\Delta}, \overline{\Delta}, \underline{\delta}, \overline{\delta}),$$

where u is the endpoint of the path different from the depot, S is the set of customer vertices visited along the path, $\underline{t}$ is the minimum time required to traverse the path, $\underline{\hat{c}}$ is the minimum reduced cost of the path, $\underline{\Delta}$ and $\overline{\Delta}$ (scalar values) are the minimum and the maximum amount of residual energy that can exist in the battery when the vehicle reaches u from the depot, $\underline{\delta}$ and $\overline{\delta}$ (vectors with one component for each technology) are the lower and upper bounds on the total amount recharged with each technology along the path. For brevity, we indicate by P the polytope defined by the lower and upper bounds. The information conveyed by $\underline{t}$ and $\underline{\hat{c}}$ is indicated for convenience but it can be obtained from the knowledge of P.

Relying upon these definitions we developed and tested a dynamic programming algorithm to price out columns. Besides fathoming dominated states, the algorithm also relies on acceleration techniques such as bounding and state space relaxation.

In our talk we will present computational results obtained on benchmark instances from the literature on the pricing problem for the EVRP.

References

[1] S. Erdogan and E. Miller-Hooks, *A Green Vehicle Routing Problem*, Transportation Research Part E 48, 100-114, 2012.

[2] Á. Felipe, M.T. Ortuño, G. Righini and G. Tirado, *A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges*, Transportation Research Part E 71, 111-128, 2014.

[3] M. Schneider, A. Stenger and D. Goeke, *The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations* Transportation Science 48(4), 500-520, 2014.

[4] C. Koc and I. Karaoglan, *The green vehicle routing problem: A heuristic based exact solution approach*, Applied Soft Computing 39, 154-164, 2016.

[5] S. Pelletier, O. Jabali and G. Laporte, *Goods distribution with electric vehicles: review and research perspectives*, Transportation Science 50(1), 3-22, 2016.

[6] M. Schiffer and G. Walther, *The electric location routing problem with time windows and partial recharging*, European Journal of Operational Research 260(3), 995-1013, 2017.

[7] T.M. Sweda, I.S. Dolinskaya and D. Klabjan, *Optimal Recharging Policies for Electric Vehicles*, Transportation Science 51(2), 457-479, 2017.

[8] G. Desaulniers, F. Errico, S. Irnich and M. Schneider, *Exact Algorithms for Electric Vehicle-Routing Problems with Time Windows*, Operations Research 64(6), 1388-1405, 2016.

A Green Energy Grid Coupling Problem (GEGCP)

Andreas Schwenk[1,2,*], Hubert Randerath[1,2]

[1] Institute of Telecommunications Technology, TH Köln, Germany
[2] Institute of Computer Science, University of Cologne, Germany
andreas.schwenk@th-koeln.de, hubert.randerath@th-koeln.de

Abstract

We address the modeling and optimization of the coupling of energy sectors. Given a network infrastructure in form of a graph $G = (V, E)$ that consists of *a priori* unconnected components, the objective is to synthesize an optimal set of parameterized energy converters and energy storages. This enables cross–sectoral interconnection and facilitates to buffer otherwise wasted volatile energy from renewable sources. Since the underlying problem is polynomially reducible to the *Facility Location Problem (FLP)*, it is NP–hard [3]. We describe the modeling of the system and discuss approximations and context sensitive heuristics in the talk. We target to allow the computation of large–scale (real–world) problem instances in reasonable time.

Keywords : *Modeling, Mathematical Programming, Energy Management.*

1 Introduction

Conventionally, each energy sector (e.g. power, gas, heat, transport) operates independently. Produced energy is fed in, then transported via an energy network infrastructure of the same energy type and finally consumed. By the coupling of sectors, one may benefit from the equilibration and synergy effects between multiple networks. Energy of type a from regenerative sources that is not demanded in its original form at production time, can be converted into energy type b that has a higher demand probability. For example, power–to–gas (P2G) converters transform electrical power, e.g. from wind energy into gas fuel. The additional integration of energy storages into the infrastructure allows to buffer stochastic supplies. Noteworthy, the sole implementation of storages is not sufficient, since e.g. batteries imply significantly higher investment costs per energy unit than heat-storages. Integrated energy is one of the key technologies to progress the energy transition (*Energiewende*) [2].

Given an initial infrastructure, we introduce the *Green Energy Grid Coupling Problem (GEGCP)* to address the optimal integration of energy conversion and storage devices, such that both investment costs and greenhouse gas emissions are minimized, i.e. the loss of energy of regenerative sources is kept as low as feasible. Under the assumption that a sufficient number of network nodes of different energy sectors are geometrically close, a dedicated synthesis of transport edges can be omitted. In this work, we restrict to pure device synthesis.

2 Modeling

For simplicity, let $[n]$ declare the set $\{1, 2, \dots, n\}$ for $n \in \mathbb{N}$. An ordered set on M with key k is denoted by $(M, \leq_k)$, i.e. with $m_i \in M$ we have: $k(m_1) \leq k(m_2) \leq \dots$. A shortest path problem in a graph G from vertex $v_i \in V(G)$ to $v_j \in V(G)$, with respect to edge weight w, is abbreviated by $\rho := \mathrm{SPP}_w(v_i, v_j)$. Then ρ is the edge sequence $(e_1, e_2, \dots)$ from v_i to v_j.

[*]Funded by the European Regional Development Fund (EFRE-0800106)

We define a set of (initially independent) energy sectors by $[\xi]$. Let $G_k = (V_k, E_k)$ be an undirected and simple graph that describes the network of energy sector $k \in \xi$. We may assume the following graph properties for every instance k: G_k is always planar. The average degree $d(G_k)$ is approximatively 2. The degree sequence is bounded by minimum degree $\delta(G_k) = 1$ and maximum degree $\Delta(G_k) \approx 20$.

Independent Networks Let V_k be the finite set of vertices of G_k and let $T_i \in \{`n`, `p', `c'\} \in \mathbb{N}$ enumerate the type of vertex $v_i \in V_k$. The type T is either a passive connection node 'n' or an actor $\in \{`p', `c'\}$, that distinguishes between producers 'p' and consumers 'c'. A producer node $v_i \in V_k$ has an environmental factor $\alpha_i \in [0, 1] \in \mathbb{R}$ that evaluates the attractiveness of the producer for the environment: Green and volatile sources have a higher value than conservative sources.

The load (or demand resp.) of a node $v_i \in V_k$ is described by a time series $\tau_i := \{x_t\}_{t=1}^T$. We index the k-th sample of τ_i by $\tau_i^{(k)}$. For homogeneity, all samples of non–actors are equal to zero. Supply is indicated by a positive sign and demand is indicated by a negative sign. Let $p_i = (x, y) \subseteq \mathbb{R}^2$ be the geographic coordinates of vertex v_i.

The set of edges is defined by $E_k(G_k) = \{(v_i, v_j) \mid v_i \in V(G_k) \wedge v_j \in V(G_k)\}$. An edge represents a physically connection (an *energy grind line*) between two vertices. Let $l_{ij} = \|p_i - p_j\|_2$ be the length, $c_{ij} \in \mathbb{R}_0^+$ the capacity and η_{ij} the energy conversion factor of an edge $(v_i, v_j) \in E(G_k)$. The conversion factor η_{ij} unifies (a) the losses of physical energy lines and (b) interposed transformers, if applicable. Let $k : \mathbb{R}^n \to \mathbb{R}$ be a cost function.

The actual implementation of costs for vertices and edges (as seen later) is denoted by k_i for $v_i \in V_k$ and k_{ij} for $e_{ij} \in E_k$ respectively. Costs k_i (resp. k_{ij}) are set to 0 for all nodes and edges of the initial input infrastructure. We include costs of synthesized devices within the optimizing process.

Network Approximation The following proposition holds for energy graph instances.

Proposition 1 *If the transmission velocity in G_k is negligible*[1]*, then $G'_k := f_w(MST(G_k)) \approx G_k$, with $MST(G_k)$ the maximum spanning tree of G_k. If each cycle of G_k has semantics of failure safety*[2]*, then $f_w : G \to G$ is the identity function. Otherwise, f_w adjusts the edge capacities c_{ij}.*

In the adjusted graph G'_k, the energy flows from source to sink trivially and serves as preliminary work for a more simple formulation of energy flow constraints in section 2.1.

Composite Networks We define a coupled energy network $G := \bigcup_k G_{k \in \xi}$ that unifies all sectors. Energy is consistently expressed in watt hours [Wh] for all sectors, such that energy conversion does not require to change the unit. The inclusion of devices is described in the next paragraphs. Note that the optimal number of storages and converters is initially unknown.

The instantiation of a *storage device* that buffers fluctuating energy is implemented by appending a vertex v_s and an edge e_s ($s \in \mathbb{N}$) to G, i.e. $G := (V(G) \cup v_s, E(G) \cup e_s)$. The physical storage device itself is represented by v_s and the type of v_s is $T_s :=$'s'. We declare the device–type $D_{i,s} \in \mathbb{Z}$. The maximum charge is denoted by $c_{s,m}$. The cost function k_s is defined by $f_{k,s}(c_{s,m}, T_s)$ and depends on the former attributes. Let $S = (s_i) \in \mathbb{Z}_2^{|G|}$ be a vector that indicates the current instantiation of storage devices. Each $s_i \in S$ expresses whether an energy storage is instantiated close to $v_i \in G$. For simpler index notation, we assume $s := i$, as long as the uniqueness of indexing is maintained. The created edge $e_s = (v_i, v_s)$ has energy loss $\eta_{i,s}$ and a capacity $\phi_{i,s_m} = f(c_s, T_s)$ that symmetrically bounds the maximum input and output flow. An attractiveness–factor $\beta_{i,s} := f_\beta(\phi_{s,m}, T_s, v_i) : D \to [0, 1] \in \mathbb{R}$ estimates the *a priori* qualification of v_i to be a candidate for constructing a nearby storage of given properties.

[1]E.g. valid for energy sector power, since problem instances have a radius of only several kilometers.

[2]So called *(N-1) considerations* in the context of energy networks.

Greenhouse gas emissions are denoted by $\gamma_i := f_\gamma(\phi_s, T_s) : D \to \mathbb{R}^+$ and correlate with the chosen storage type.

The synthesis of *energy converters* implies the creation of an additional and directed[3] edge $e_c := e_{ij} = \{(v_i, v_j) \mid v_i \in G_k \;\wedge\; v_j \in G_l \;\wedge\; k \neq l\}$ for each converter. As a constraint, the geometric distance $\|v_i - v_j\|_2$ of any two select vertices v_i and v_j must be less than constant Φ. Otherwise, it would be required to extend the physical energy grid. We append e_c with type $T_c :=$ 'v' to G, i.e. $G := (V(G), E(G) \cup \{e_{ij}\})$. Let $C = (c_{ij}) \in \mathbb{Z}_2^{|G|\times|G|}$ be an adjacency matrix that indicates the current instantiation of converter devices. The upper bound for the number of actual converters is the number of edges of the ξ-partite graph $K_{|G_1|,|G_2|,\ldots,|G_\xi|}$. A converter device is exclusively described by e_{ij}. It is attributed by a cost function $k_{ij,v}(c_{ij}, D_{ji,v})$, a linear energy conversion loss $\eta_{ij,v}$ (with $\eta_{ij,v} \neq \eta_{ji,v} = 0$), involves a construction attractiveness function $\beta_{ij,v}(c_{ij}, D_{ij,v}, v_i)$ and greenhouse gas emissions $\gamma_{ij,v}(c_{ij}, D_{ij,v})$. The definition of properties is the same as for storages, apart from naming conventions. In case of compound conversion devices, i.e. $n \in \mathbb{N}$ input sources are converted to one output medium, the edges of a complete bipartite graph $K_{1,n}$ are added. All other constraints and properties hold. Only those vertices are selected that have a valid type combination, persisted in a device property database.

Previous work in [4] describes a domain–specific language that introduces syntactical and semantical structures to simplify modeling aspects.

2.1 Optimization Problem

The following equations define the optimization problem for GEGCP. The class is of Mixed-Integer-Linear-Programming (MINLP)[4]. The objective function (2) minimizes f_i from equation (1). Our first objective f_1 minimizes the construction costs for all storages and converters. The sum of greenhouse gas emissions is calculated in f_2. The sum of wasted energy w in form of unused energy from renewable sources is defined in f_3, with $eq(a, b)$ the binary predicate $equals(a, b) := (a == b)$.

$$f_1 = \sum_{v_i \in V} k_{i,s} + \sum_{e_{ij} \in E} k_{ij,v}, \quad f_2 = \sum_{v_i \in V} \gamma_{i,s} + \sum_{e_{ij} \in E} \gamma_{ij,v}, \quad f_3 = \sum_{\{v_i \in V \mid eq(T_i, `p`)\}} w_i \tag{1}$$

For multicriteria *a priori* methods (e.g. linear scalarization), all functions f_i can be monetized; e.g. via current energy marked prices and, if applicable, time factors.

Note that we omit the redeclaration of known variables, functions and assignments for the constraints in equations (3) to (16). We define $0 \Leftrightarrow False$, $1 \Leftrightarrow True$ and $\tilde{V}_t := \{v_i \in V \mid \mathrm{eq}(T_i, t)\}$. For better understanding, we partly use an algorithmic notation (indicated by assignments of the form ":="). Equations (3) to (12) ensure that all consumers are satisfied at any time. We iterate over all producers by priority α (and then over all [partly] charged storages ordered by β), so that renewable producers have the highest priority distributing energy. Energy flows from producers 'p' (resp. 's') to consumers 'c' via a shortest path that correlates with η. To simplify the temporal aspects and considerations of energy storages, we homogenize the device representation: a virtual current fill–level ℓ_i is defined for each vertex v_i. The level describes the current supply, demand or storage. Potential equalization tends to equilibrium. For a better overview, we omit listing the constraint that storages behave like consumers with a demand of $\phi_{s.m} - \ell_s$. Condition (13) ensures, that converters are only created, if two sectors are geometrically close to each other. Inequality (14) fulfills a given trade–off factor ψ that ensures that a given percentage of all consumed energy is supplied by renewable sources. Equations (15) and (16) bound the converter throughput and the maximum storage capacity, respectively.

Remark: one should add a tolerance to the determined storage capacity, outside of mathematical programming.

[3]Gen. we abstract edges to be undirected, since e.g. consumers can also feed photovol. energy into the grid.

[4]We address Mixed-Int.-Quadratic-Programming, if all devices and cost functions are def. restrictively.

$$\min \quad \Big[f_1, f_2, f_3\Big] \tag{2}$$

$$s.t. \quad \forall_{v_i \in V} : w_i := 0; \qquad \varrho := 0, \sigma := 0 \tag{3}$$

$$\forall_{[t]} : \tag{4}$$

$$\forall_{v_i \in V} : \quad \ell_i := \tau_i^{(t)} \tag{5}$$

$$\forall_{p_i \in \Big(\, (\, \tilde{V}_{`p'} \, , \, \geq_\alpha \, ) \, \cup \, ( \, \tilde{V}_{`s'} \, , \, \geq_\beta \,) \, \Big)} : \tag{6}$$

$$\forall_{c_j \in \, (\, \tilde{V}_{`c'} \, , \, \leq_{\mathrm{SPP}_{1-\eta}(p_i, c_j)} \,)} : \tag{7}$$

$$\tilde{\eta} := \Pi_{[e(v_i, v_j) \in \mathrm{SPP}_{1-\eta}(p_i, c_j)]} \, \eta_{ij} \tag{8}$$

$$\Delta_\ell := \min(\ell_{p_i}, -\ell_{c_j}/\tilde{\eta}) \tag{9}$$

$$\ell_{p_i} := \ell_{p_i} - \Delta_\ell \qquad \ell_{c_j} := \ell_{c_j} + \Delta_\ell \cdot \tilde{\eta} \tag{10}$$

$$\varrho := \varrho + \alpha_i \Delta_\ell \qquad \sigma := \sigma + (1 - \alpha_i)\Delta_\ell \tag{11}$$

$$w_i := w_i + f_{w\alpha}(\alpha_i, \ell_{p_i}) \tag{12}$$

$$\forall_{i \in [n(G)], j \in [n(G)]} : C_{ij} \leq 1 - \Big[\|v_i - v_j\|_2 \leq \Phi \wedge v_i \in G_l \wedge v_j \in G_m \wedge l \neq m\Big] \tag{13}$$

$$\varrho \leq \psi \cdot \sigma, \quad \psi \in [0, 1] \in \mathbb{R} \tag{14}$$

$$\forall_{\{v_i \in V, v_j \in V | C_{ij}\}} : f_{c,\min}(D_{ij,v}) \leq c_{ij,v} \leq f_{c,\max}(D_{ij,v}) \tag{15}$$

$$\forall_{\{v_i \in V | S_i\}} : f_{\phi,\min}(D_{i,s}) \leq \phi_{i,m} \leq f_{\phi,\max}(D_{i,s}) \tag{16}$$

3 Conclusion and Perspectives

Since the number of vertices per real–world energy sector is in the range $[10^3, 10^4]$ for (sub)urban scenarios, exact algorithms are too expensive. NP–completeness of similar problems is proven in [1]. In the talk, we will present a label correcting algorithm and heuristics as an approximation approach and also define the class of approximation (APX, PTAS/PAS, FPTAS).

An extension to the problem definition would be to also consider an energy grid expansion (adding edges for power lines), to bridge long distances. It is also conceivable to interconnect single storages to multiple vertices $v_i \in V$.

Acknowledgment

This work is part of the research project ES–FLEX–INFRA; funded by the European Regional Development Fund (EFRE–0800106).

EUROPEAN UNION
Investing in our Future
European Regional
Development Fund

EFRE.NRW
Investitionen in Wachstum
und Beschäftigung

References

[1] Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are np-complete. *Information processing letters*, 12(3):133–137, 1981.

[2] Brian Vad Mathiesen, Henrik Lund, David Connolly, Henrik Wenzel, Poul Alberg Østergaard, Bernd Möller, Steffen Nielsen, Iva Ridjan, Peter Karnøe, Karl Sperling, et al. Smart energy systems for coherent 100% renewable energy and transport solutions. *Applied Energy*, 145:139–154, 2015.

[3] Sanjay Melkote and Mark S Daskin. Capacitated facility location/network design problems. *European journal of operational research*, 129(3):481–495, 2001.

[4] Andreas Schwenk. Cempl: A new domain-specific language for rapid modeling of cross-energy systems. In *Energy and Sustainability Conference (IESC), 2017 International*, pages 1–6. IEEE, 2017.

Modeling and Solving Combinatorial Optimization Problems using Semidefinite Programming

Angelika Wiegele

Alpen-Adria-Universität Klagenfurt
Institut für Mathematik
Universitätsstr. 65-67, 9020 Klagenfurt am Wörthersee
angelika.wiegele@aau.at

Abstract

Semidefinite Programming (SDP) is an an extension of Linear Programming where a matrix-variable is optimized over the intersection of the cone of positive semidefinite matrices with an affine space. Semidefinite programming relaxations exist for a variety of combinatorial optimization problems, and many ways to tighten them have been proposed.

In this talk we will show how to apply SDP to efficiently approximate two important NP-hard combinatorial problems, namely the max-cut problem and the stable set problem. Besides modeling the problems as SDP, we will present a way to strengthen the relaxation using exact subgraph constraints.

Linked to the question of modeling a problem using semidefinite programming is the question of solving the resulting SDP. Standard methods like interior point algorithms are not applicable already to medium-sized problems due to the number of constraints or the size of the matrix. We will present alternative methods in order to obtain approximate solutions to the SDP in reasonable time and using affordable memory requirements.

Graph partitioning using matrix differential equations

Eleonora Andreotti[1], Dominik Edelmann[2], Nicola Guglielmi[1] and Christian Lubich[2]

[1] Università degli Studi di L'Aquila, L'Aquila, Italy
eleonora.andreotti@graduate.univaq.it
guglielm@univaq.it

[2] Eberhard Karls Universität, Tübingen, Germany
edelmann@na.uni-tuebingen.de
lubich@na.uni-tuebingen.de

Abstract

Given a connected undirected weighted graph, we are concerned with problems related to partitioning the graph. First of all we look for the closest disconnected graph (the minimum cut problem), here with respect to the Euclidean norm. We are interested in the case of constrained minimum cut problems, where constraints include cardinality or membership requirements, which leads to NP-hard combinatorial optimization problems. These problems are restated as matrix nearness problems for the weight matrix of the graph. A key element in the solution of these matrix nearness problems is the use of a constrained gradient system of matrix differential equations.

Keywords : *Constrained minimum cut, spectral graph partitioning, algebraic connectivity, Fiedler vector, matrix nearness problem, constrained gradient flow, matrix differential equation, graph theory*

1 Introduction and problem formulation

We present a novel approach to partitioning a connected weighted undirected graph. Here and in the following, a graph is a tuple $(\mathcal{V}, \mathcal{E}, W)$ of a vertex set $\mathcal{V} = \{1, \dots, n\}$, an edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and a matrix $W = (w_{ij})$ that collects positive weights w_{ij} associated with edges $(i, j) \in \mathcal{E}$ ($w_{ij} = 0$ if $(i, j) \notin \mathcal{E}$).

We consider the Frobenius-norm minimum cut problem and allow for constraints such as prescribing the minimum cardinality of connected components or assigning *a priori* selected vertices to a component. The matrix nearness problem is thus to find a cut $\mathcal{C}$, i.e., a set of edges that yield a disconnected graph when they are removed from the edge set $\mathcal{E}$ of the given graph, such that $\sum_{(i,j)\in\mathcal{C}} w_{ij}^2$ is minimized. This problem will further be considered with additional constraints:

- *Membership constraint:* It is required that a given set of vertices $\mathcal{V}^+ \subset \mathcal{V}$ is in one connected component and another given set of vertices $\mathcal{V}^- \subset \mathcal{V}$ is in the other connected component, where $\mathcal{V}$ denotes the vertex set of the given graph.
- *Cardinality constraint:* It is required that each of the connected components has a prescribed minimum number $\overline{n}$ of vertices.

It is known that cardinality constraints make the problem NP-hard [1, 2].

We use spectral graph theory as pioneered by Fiedler [3]: The second smallest eigenvalue λ_2 of the *Laplacian matrix* $L = \mathrm{Lap}(W)$ of the graph is zero if and only if the graph is disconnected, and the corresponding eigenvector indicates the membership of vertices to the connected components.

We formulate and use a gradient system of matrix differential equations to drive the smallest nonzero eigenvalue of the Laplacian matrix to zero. This approach can be extended to other partitioning problems beyond the constrained minimum cut problems considered here.

The approach takes basic ideas and techniques of recent algorithms for eigenvalue optimization via differential equations, as given for example in [5, 4, 7, 6], to another application area. A common feature is a two-level procedure, where on the inner level a gradient flow drives perturbations to the original matrix of a fixed size into a (local) minimum of a functional that depends on eigenvalues and possibly eigenvectors, and in an outer iteration the minimal perturbation size is determined such that the functional becomes zero. Our approach in the unconstrained case can be summarized as follows.

1. Given $\varepsilon > 0$, we look for a symmetric matrix $E = (e_{ij}) \in \mathbb{R}^{n\times n}$ with the same sparsity pattern as W (i.e., $e_{ij} = 0$ if $w_{ij} = 0$), of unit Frobenius norm, with $W + \varepsilon E \geq 0$ (with component-wise inequality) such that the second smallest eigenvalue of $\mathrm{Lap}(W + \varepsilon E)$ is minimal. The obtained minimizer is denoted by $E(\varepsilon)$.

2. We look for the smallest value of ε such that the second smallest eigenvalue of $\mathrm{Lap}(W + \varepsilon E(\varepsilon))$ equals 0.

Once the weight matrix $W^\star$ of the disconnected graph is computed, the eigenvector corresponding to the second smallest eigenvalue of $\mathrm{Lap}(W^\star)$ indicates the membership of vertices to the connected components and thus gives a partition of the graph.

In the constrained case, the functional in the inner level is more intricate: Let $x = (x_i) \in \mathbb{R}^n$ be the eigenvector to the second smallest eigenvalue λ_2 of $\mathrm{Lap}(W + \varepsilon E)$. Let $\mathcal{V}^-$ and $\mathcal{V}^+$ be the set of indices whose membership to different components of the cut graph is prescribed. Let $x^- = (x_i^-)$ with $x_i^- = \min(x_i, 0)$ and $x^+ = (x_i^+)$ with $x_i^+ = \max(x_i, 0)$ collect the negative and positive components of x, respectively. Let n^- and n^+ be the numbers of negative and nonnegative components of x, respectively. We denote the averages of x^- and x^+ by

$$\langle x^- \rangle = \frac{1}{n^-}\sum_{i=1}^{n} x_i^-, \qquad \langle x^+ \rangle = \frac{1}{n^+}\sum_{i=1}^{n} x_i^+.$$

Motivated by the special form of the eigenvectors as given in Fiedler's theorem, we consider the functional

$$F_\varepsilon(E) = \lambda_2(\mathrm{Lap}(W + \varepsilon E)) + \frac{\alpha}{2}\sum_{i\in\mathcal{V}^-}(x_i - \langle x^-\rangle)^2 + \frac{\alpha}{2}\sum_{i\in\mathcal{V}^+}(x_i - \langle x^+\rangle)^2,$$

where $\alpha > 0$ is a weight to be chosen. This functional is to be minimized under the inequality constraints $W + \varepsilon E \geq 0$, the norm constraint $\|E\| = 1$ and the symmetry and the sparsity pattern of E.

It is remarkable that the numerical approach as well as the methods remain the same with the only difference that it is more sophisticated to compute the gradient of F_ε in the constrained case.

2 Example

Figure 1 shows the graph of character co-occurrence in Les Miserables [8]. This graph consists of 77 vertices (representing characters). According to a standard Fiedler partitioning, 22 of these belong to one part and the remaining 55 belong to the other part. Asking for the partitioning of the graph with the cardinality constraint with threshold $\bar{n} = 35$, we obtain the result shown in Figure 1.

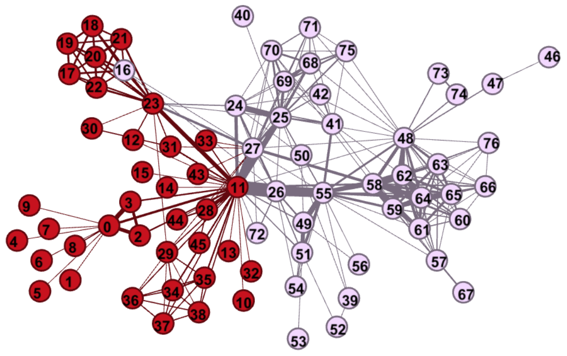


FIG. 1: Les Miserables, cardinality-constrained graph partitioning

3 Conclusion

The presented algorithm performs remarkably well in the examples from the literature on which we have tested it. As with the previous algorithms cited above, there are contrived examples where our algorithm could get stuck in a local minimum that is not a global minimum.

As opposed to combinatorial algorithms, the algorithm presented here modifies all weights of the graph as it proceeds, and only in the end arrives at the cut and the unchanged remaining weights.

The proposed algorithm is an iterative algorithm, where in each step the second eigenvalue and the associated eigenvector of the Laplacian of a graph with perturbed weights are computed. In the cardinality- or membership-constrained cases, additionally a linear system with an extended shifted Laplacian is solved in each step. For a large sparse connected graph (where the number of edges leaving any vertex is moderately bounded), these computations can be done in a complexity that is linear in the number of vertices. In the known (unconstrained) minimum cut algorithms, the computational complexity is at least quadratic [9]. It is thus conceivable that for large sparse graphs, the proposed algorithm can favorably compete with the classical unconstrained minimum cut algorithms. In constrained cases, it appears that the computational complexity is even more favorable in comparison with the existing heuristic combinatorial algorithms as proposed in [2]. However, as of now no detailed comparisons of the relative merits of the conceptually and algorithmically different approaches have been made.

References

[1] M. Bruglieri, M. Ehrgott, H. W. Hamacher, and F. Maffioli. An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints. *Discrete Appl. Math.*, 154(9):1344–1357, June 2006.

[2] M. Bruglieri, F. Maffioli, and M. Ehrgott. Cardinality constrained minimum cut problems: Complexity and algorithms. *Discrete Appl. Math.*, 137(3):311–341, March 2004.

[3] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Math. J.*, 23(98):298–305, 1973.

[4] N. Guglielmi, D. Kressner, and C. Lubich. Low rank differential equations for Hamiltonian matrix nearness problems. *Numer. Math.*, 129:279–319, 2015.

[5] N. Guglielmi and C. Lubich. Differential equations for roaming pseudospectra: paths to extremal points and boundary tracking. *SIAM J. Numer. Anal.*, 49:1194–1209, 2011.

[6] N. Guglielmi and C. Lubich. Matrix stabilization using differential equations. *SIAM J. Numer. Anal.*, page in press, 2018.

[7] N. Guglielmi, C. Lubich, and V. Mehrmann. On the nearest singular matrix pencil. *SIAM J. Matrix Anal. Appl.*, 38:776–806, 2017.

[8] D. E. Knuth. *Stanford GraphBase: A Platform for Combinatorial Computing, The.* Addison-Wesley Professional, 1st edition, 2009.

[9] M. Stoer and F. Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, 1997.

An algorithm for computing lower bounds for the Microaggregation problem

Jordi Castro[1] , Claudio Gentile[2] , Enrique Spagnolo[3]

[1] Department of Statistics and Operations Research, Universitat Politècnica de Catalunya, Jordi Girona 1-3, 08034 Barcelona, jordi.castro@upc.edu

[2] Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", Consiglio Nazionale delle Ricerche (IASI-CNR), Via dei Taurini 19, Rome, gentile@iasi.cnr.it

[3] Kernel Analytics, C/Balmes 89, 6o 4a, 08008, Barcelona enric.spagnolo@kernel-analytics.com

Abstract

Public use of microdata files requires preprocessing to protect privacy. Microaggregation consists in aggregating data into clusters of size at least k such that the spread between individuals' and centroid cluster values is minimized. This paper proposes an algorithm based on Column Generation to compute lower bounds on the spread.

Keywords : *Microaggregation, Statistical Disclosure Control, Column Generation.*

1 Introduction

Microdata consists of a file with individuals (people, companies, etc) and attributes for those individuals. A microdata file is a mapping $M : V \subseteq P \to D(V_1) \times \ldots \times D(V_t)$, where P is a population, V is a sample of the population and $D(V_i)$ is the domain of the attribute $i \in \{1, \ldots, t\}$. Microdata files must be treated or "protected" before being released, otherwise confidential individual information would be jeopardized. The set of methods for this task form the field of *Statistical Disclosure Control* (SDC). Microaggregation (MA)—introduced in [3]—is an SDC technique mainly for numeric variables related with *k-anonymity* [1].

Definition 1 *Given $k \in \mathbb{N}$, $k \geq 2$, let M be a microdata with n individuals $n \geq k$ and t attributes $V_1, \ldots, V_t$. Let $g = (V_{j_1}, \ldots, V_{j_m})$ be a subset of attributes, $j_q \in \{1, \ldots, t\}$, $q = 1, \ldots, m$. Then we say M is k-anonymous if for every possible value in $D(V_{j_1}) \times \ldots \times D(V_{j_m})$, there exist either 0 or at least k individuals in V with this value for the attributes in g.*

MA intends to modify the values of a subset of attributes g, called *quasi-identifier*, so that the microdata satisfies k-anonymity for g. Therefore, it first partitions the individuals into subsets of size at least k, called *clusters*, and then substitutes the values for g in the cluster with their centroid, to minimize the loss of information, called *spread.* In practical cases, the value of k is relatively small (e.g., $3 \leq k \leq 5$, see [3]). A widely used measure to evaluate the spread is the *sum of squared errors* (SSE) [3]:

$$SSE = \sum_{s=1}^{q} \sum_{j=1}^{n_s} (a_{sj} - \overline{a}_s)^T (a_{sj} - \overline{a}_s) = \sum_{s=1}^{q} \sum_{j=1}^{n_s} (a_{sj} - \overline{a}_s)^2, \tag{1}$$

where q denotes the number of clusters, n_s the size of cluster $C_s = \{a_{sj} \,|\, j = 1, \ldots, n_s\}$, and $\overline{a}_s = \frac{1}{n_s} \sum_{j=1}^{n_s} a_{sj}$ its centroid, for $s = 1, \ldots, q$. From now on, we will denote as *feasible clustering* a partition into clusters of size at least k. There exist different papers about heuristic algorithms that obtain feasible clusterings with reasonable SSE, e.g., [2, 3].

Proposition 1 *[3] Any cluster belonging to an optimal microaggregation must have size less than or equal to* $2k-1$.

Example 1 *Let* $g{=}(Employees, Surface)$ *be a quasi-identifier for a microdata of factories. Let* $k{=}2$ *and suppose that a MA procedure suggests us to join the factories in Table 1 in a cluster. Its centroid is* $\frac{55+48+41}{3}=48$ *employees and* $\frac{1410+1205+1120}{3}{=}1245$ m^2 *of surface. Thus, in the published microdata, factories* f_1*,* f_2*,* f_3 *will all have 48 employees and 1245*m^2 *of surface.*

Factory	Employees	Surface (m^2)
f_1	55	1410
f_2	48	1205
f_3	41	1120

TAB. 1: Values for the attributes in g for factories f_1, f_2, and f_3 in the original microdata.

2 A Formulation for the Microaggregation problem

We give an exact Integer Programming (IP) formulation for MA. Unfortunately, it is a fractional nonconvex program with integer variables. We need the following preliminary result:

Proposition 2 *Given a cluster* $C_s = \{a_{sj} \mid j = 1, \ldots, n_s\}$ *with centroid* $\overline{a}_s$*. Then,*

$$\sum_{j=1}^{n_s}(a_{sj}-\overline{a}_s)^2 = \frac{1}{2n_s}\sum_{i=1}^{n_s}\sum_{j=1}^{n_s}(a_{si}-a_{sj})^2. \tag{2}$$

Now, we can associate binary variables x_{ij} for all pairs $i,j \in V \times V, i \neq j$: $x_{ij} = 1$, if and only if a_i and a_j belong to the same cluster after microaggregation, and $x_{ij} = 0$ otherwise. To simplify the writing we will index with i the corresponding individual a_i and we will denote with n_i be the size of the cluster where a_i belongs to. Using relation (2) we can easily derive that

$$SSE = \frac{1}{2}\sum_{i=1}^{n}\frac{\sum_{j=1 j\neq i}^{n}(a_i-a_j)^2 x_{ij}}{n_i}. \tag{3}$$

Moreover, $n_i = \sum_{j=1,j\neq i}^{n} x_{ij} + 1$, thus we get that the MA objective function is:

$$SSE = \frac{1}{2}\sum_{i=1}^{n}\frac{\sum_{j=1,j\neq i}^{n}(a_i-a_j)^2 x_{ij}}{\sum_{j=1,j\neq i}^{n} x_{ij}+1}. \tag{4}$$

To force the variables to describe clusters, we must express two conditions. First, clusters must be *complete*: if i and r are in the same cluster (i.e., $x_{ir} = 1$) and j and r are in the same cluster too ($x_{jr} = 1$), then it must be that i and j are in the same cluster ($x_{ij} = 1$). This can be enforced by the following set of inequalities, named *triangle inequalities*:

$$x_{ir} + x_{jr} - x_{ij} \leq 1 \quad i,j,r \in V, \quad i \neq j, r \neq j, i \neq r. \tag{5}$$

Second, we must force the clusters to have size n_i at least k, for $i \in V$ (*size inequality*):

$$\sum_{j=1,j\neq i}^{n} x_{ij} \geq k-1 \quad i \in V. \tag{6}$$

Summing up, an IP formulation for MA is the following:

$$\min\ (4) : (5),\ (6),\ x_{ij} = x_{ji}, x_{ij} \in \{0,1\}\ i,j \in V, \quad i \neq j. \tag{7}$$

The most remarkable issue is that the objective function in (7) is nonconvex, so this model cannot be used to provide an effective lower bound on MA. On the other hand, by Proposition 1, a lower bound can be obtained by substituting n_i in (3) with $2k-1$, but it would be loose even for integer feasible solutions. With this transformation the problem can be reduced to the Clique Partioning Problem with Minimum Clique Size Requirement (CPPMIN) [4].

3 Column Generation for Microaggregation

In this section, we introduce a Column Generation (CG) approach for the MA problem inspired by [4]. We modify the weight of a cluster and more importantly the whole Pricing Problem scheme for cluster generation. By Proposition 1 we define $\mathcal{C}^* = \{C \subseteq V : k \leq |C| \leq 2k-1\}$ as the set of feasible clusters for MA. Let x_C be the binary variable that indicates whether cluster $C \in \mathcal{C}^*$ is or not in the solution for MA. From (2) the weight w_C of cluster C can be computed as:

$$w_C = \frac{1}{2}\sum_{i\in C}\frac{\sum_{j\in C}(a_i - a_j)^2}{|C|} = \sum_{i,j\in C}\frac{(a_i-a_j)^2}{|C|}. \tag{8}$$

On account of this we can formulate the Microaggregation problem as follows:

$$\begin{array}{ll} \min \sum_{C\in\mathcal{C}^*} w_C x_C & \\ \sum_{C\in\mathcal{C}^*: v\in C} x_C = 1 & v \in V \\ x_C \in \{0,1\} & C \in \mathcal{C}^*. \end{array} \tag{9}$$

From (9) we can set the Master Problem (MP) by considering a subset $\overline{\mathcal{C}} \subseteq \mathcal{C}^*$. The Pricing Problem looks for a cluster C of size at least k that minimizes (8) minus the sum of the dual prices λ_v associated with the individuals $v \in C$ in the continuous relaxation of (9). In order to take account of the size of the cluster, the problem is subdivided into k Pricing Subproblems (PSP) for fixed size cluster $\eta = k, \ldots, 2k-1$:

$$\min_{C\subset V:|C|=\eta} \overline{w}_C, \quad \overline{w}_C = w_C - \sum_{v\in C}\lambda_v = \sum_{u,v\in C}\frac{(a_u-a_v)^2}{\eta} - \sum_{v\in C}\lambda_v. \tag{10}$$

When none of these k PSPs adds a cluster with negative reduced cost, either the MP solution is binary and optimal for MA, or it is fractional and defines a lower bound on the optimal MA objective function value that can be used in a Branch&Price scheme. Up to our knowledge, this is the first approach in the literature for the computation of a lower bound for SSE.

In order to solve (10) we define a representation on the complete undirected graph $K_n = (V, E)$ with edge weights $(a_i - a_j)^2/\eta$, $e = ij \in E$, and node weights λ_i, $i \in V$. We propose a new ILP model considering only edge variables. Let $k \geq 2$; hence $\eta \geq 2$. Suppose C is the solution of (10), we define variables z_{ij}, $ij \in E$ such that: $z_{ij} = 1$ if and only if $i, j \in C$ and $z_{ij} = 0$ otherwise. The following *size constraint* enforces the cluster C to have size η:

$$\sum_{e\in E} z_e = \frac{\eta(\eta-1)}{2}. \tag{11}$$

Moreover, let i, $j \in V$, $i \neq j$, the following *node-to-node inequality* for i and j

$$\sum_{e\in\delta(i)\setminus(i,j)} z_e - (\eta-2)z_{ij} \geq 0, \tag{12}$$

if $z_{ij} = 1$, requires that i is connected to at least $\eta - 2$ nodes different from i and j.

Proposition 3 *[5] Let $\eta \geq 2$, the binary feasible solutions satisfying nonnegativity constraints and constraints (11), (12) represent clusters in K_n with size η.*

If a node $i \in C$, then i will be adjacent to $\eta-1$ nodes in C. It follows that $(\sum_{e\in\delta(i)} z_e)/(\eta-1)$ is 1 when $i \in C$ and 0 otherwise. In summary, the objective function in (10) is:

$$\sum_{ij\in E}\frac{(a_i-a_j)^2 z_{ij}}{\eta} - \sum_{i\in V}\lambda_i\frac{\sum_{ij\in\delta(i)} z_{ij}}{\eta-1}. \tag{13}$$

From Proposition 3 and (13) we derive an ILP model of the PSP with fixed cluster size $\eta \geq 2$:

$$\min\ (13) : (11),\ (12),\ z_e \in \{0,1\}\ e \in E. \tag{14}$$

In [5] the polyhedral properties of problem (14) have been deeply studied. We implemented in a C++ code the CG scheme to compute the lower bound on SSE. The MP has been started with the set of cluster $\overline{\mathcal{C}}$ defined by the solution of the heuristic MDAV [2]. Formulation (14) has been used to compute the solution of PSP when $\eta \geq 7$. For η such that $2 \leq \eta \leq 6$, complete enumeration results to be more effective.

4 Computational Tests

The code of the CG scheme for Microaggregation (CGM), discussed in Section 3, has been tested with two standard data sets in the Microaggregation literature: "Tarragona" (834 records and 13 attributes) and "Census" (1080 records and 13 attributes). See, e.g., [3] for reference. We extracted subsets of individuals in both microdata sets of size n equal to 30, 40, and 50 individuals and we considered three different minimal cluster sizes k equal to 3, 4, and 5.

For each combination n, k, we run CGM and obtained a lower bound on the SSE for MA. On the other hand, we also applied CG to CPPMIN based model (substituting n_i with $2k-1$ in (3)) and obtained another lower bound. We then computed the GAP of both lower bounds by comparing each one against the SSE obtained by MDAV heuristic solution. Those GAPs are noted as $\mathrm{GAP}_{\mathrm{CGM}}$ and $\mathrm{GAP}_{\mathrm{CPPM}}$. In Table 2 the results are summarised. Column "Int?" indicates with a "YES" or a "NO" if the eventual solution of the MP for CGM is integer or not. In case the MP solution is integer, then it corresponds to an actual clustering and, thus, the optimal MA has been achieved.

		Tarragona Data Set			Census Data Set		
n	k	$\mathrm{GAP}_{\mathrm{CGM}}$	Int?	$\mathrm{GAP}_{\mathrm{CPPM}}$	$\mathrm{GAP}_{\mathrm{CGM}}$	Int?	$\mathrm{GAP}_{\mathrm{CPPM}}$
30	3	14.25	NO	48.55	28.54	NO	57.12
30	4	22.96	NO	55.98	26.73	NO	58.13
30	5	23.94	YES	57.74	7.34	YES	48.52
40	3	8.49	NO	45.10	15.97	NO	49.58
40	4	18.38	NO	53.36	16.22	YES	52.13
40	5	16.75	NO	53.75	8.66	YES	49.26
50	3	12.35	NO	47.41	16.61	NO	49.97
50	4	15.55	NO	51.74	24.7	NO	56.97
50	5	12.77	YES	51.54	26.13	NO	58.96

TAB. 2: GAP comparison between new CG-based relaxation and CPPMIN-based relaxation

The GAP computation for the MDAV heuristic is a first result of this work, given that this method is the first approach providing an effective lower bound for MA. Although we need further developments to perform with larger instances, we have observed that the CGM algorithm finds the optimal integer solution for MA concretely in 5 cases out of 18. Therefore we expect that the lower bound provided by this approach is very close to the optimal integer solution value. The assessment of this statement needs further work. On the contrary, the lower bound provided by CPPMIN solution results to be very poor with respect to our algorithm certainly because the cluster sizes are actually smaller than $2k-1$.

References

[1] L. Sweeney, k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10, 557–570, 2002.

[2] J. Domingo-Ferrer, V. Torra, Ordinal, Continuous and Heterogeneous k-Anonymity Through Microaggregation *Data Mining and Knowledge Discovery*, 11(2), 195–212, 2005.

[3] J. Domingo-Ferrer, J. M. Mateo-Sanz, Practical data-oriented microaggregation for statistical disclosure control, *IEEE Trans. Knowl. Data Eng.*, 14, 189–201, 2002.

[4] X. Ji, J. E. Mitchell, Branch-and-price-and-cut on the clique partitioning problem with minimum clique size requirement. *Discrete Optimization*, 4, 87–102, 2007.

[5] E. Spagnolo, *On the use of Integer Programming to pursue Optimal Microaggregation*, BSc Thesis, Univ. Politecnica de Catalunya, Sch. of Math. and Statistics, Barcelona, 2016.

Some polynomial special cases for the Minimum Gap Graph Partitioning Problem

Maurizio Bruglieri[1], Roberto Cordone[2], Isabella Lari[3], Federica Ricca[3], Andrea Scozzari[4]

[1] Politecnico di Milano
maurizio.bruglieri@polimi.it
[2] Università degli Studi di Milano
roberto.cordone@unimi.it
[3] Sapienza Università di Roma
{isabella.lari,federica.ricca}@uniroma1.it
[4] Università degli Studi Niccolò Cusano, Roma
andrea.scozzari@unicusano.it

Abstract

We study various polynomial special cases for the problem of partitioning a vertex-weighted undirected graph into p connected subgraphs with minimum gap between the largest and the smallest vertex weight.

Keywords : *Graph partitioning, min-sum gap optimization, min-max gap optimization.*

1 Introduction

The *Minimum Gap Graph Partitioning Problem* (*MGGPP*) is a graph partitioning problem [1] introduced in [2]. Let $G = (V, E)$ be an undirected connected graph, w_v an integer *weight* coefficient defined on each vertex $v \in V$, and $p \leq |V|$ a positive integer. Given a vertex subset $U \subseteq V$, we denote by $m_U = \min_{u \in U} w_u$ and $M_U = \max_{u \in U} w_u$ the minimum and maximum weight in U, respectively, and define the *gap* of U as $\gamma_U = M_U - m_U$ (if U is a singleton, $\gamma_U = 0$). The *MGGPP* consists in partitioning G into p vertex-disjoint connected subgraphs $G_r = (V_r, E_r)$, $r = 1, \ldots, p$. We consider also the *nondegenerate* problem (*MGGPPnd*), in which all subgraphs must have at least two vertices. The *min-sum* version of both problems minimizes the sum of all gaps $f^{MS} = \sum_{r=1}^{p} \gamma_{V_r}$, while the *min-max* version minimizes $f^{MM} = \max_{r=1,\ldots,p} \gamma_{V_r}$. The *MGGPP* is related to uniform graph partitioning [4] which has applications, for example, in agriculture (divide a land into parcels with small height difference [7]) and in social network analysis.

The computational complexity and the approximability of the *MGGPP* are studied in [2]. A Tabu Search metaheuristic and a Mixed Integer Linear Programming (MILP) formulation for the *min-sum* version are proposed in [3]. We note that the *min-max MGGPP* can be seen as a special case of the following more general graph partitioning problem. Given a graph G and a $n \times n$ matrix of *dissimilarities* between any pair of vertices, find a partition of G into p connected components that minimizes the maximum dissimilarity of a pair of vertices in the same component. This problem is $\mathcal{NP}$-complete even on star graphs [5]. However, the *min-max MGGPP* might be easier because the dissimilarity for any given pair of vertices u and v is computed as $|w_u - w_v|$.

In this paper, we investigate some polynomial cases of the *MGGPP* concerning special graph topologies, such as paths, spiders, stars, caterpillars and complete graphs.

2 Polynomial cases

First of all, we introduce two useful properties of the *MGGPP* (but not of the *MGGPPnd*):

P1. the optimal value does not increase as the number p of components increases;

P2. given a partition π in p components with maximum gap equal to γ, another partition π' with $p' > p$ components and a gap less than or equal to γ always exists.

These properties hold for both objectives, because a singleton with zero gap can always be disconnected from a subgraph, increasing the number of components, but not the objective value. On the basis of the above properties we are able to solve the *min-max MGGPP* by a binary search over all the $O(n^2)$ possible values γ of the optimal maximum gap, that correspond to the differences between pair of vertices' weights. Therefore, we can follow an approach based on the solution of a polynomial number of instances of the following auxiliary problem.

Definition 1 (Feasibility problem) *Find, if any, a connected partition of G having the minimum number q of components and such that each component has gap at most γ.*

If q is greater than p, the value of γ must be increased; otherwise it must be decreased. At the end of the binary search, the partition having the maximum $q \leq p$ is considered and in case $q < p$, a partition in exactly p components with the same gap value is found by further dividing some components of the partition until p components are obtained. The binary search requires a pre-sorting of the $O(n^2)$ possible values of γ implying an overall time complexity of $O(n^2 \log n)$. This complexity can be reduced as follows. First sort the weights of the vertices, in $O(n \log n)$; then consider implicitly the ordered matrix of the differences between weights and apply the $O(n)$ time procedure for finding the k^{th} smallest value in a $n \times n$ matrix of reals with sorted rows and columns [6]. With this approach the whole solution procedure requires $O(n \log n + T_F(G) \log n)$ time, where $T_F(G)$ is the time for solving the feasibility problem. We apply this approach to paths and spiders.

2.1 Paths

If G is a path, we assume its vertices to be numbered progressively considering an arbitrary direction along the path: $P = \{v_1, \dots, v_n\}$. Every feasible solution is a partition into p vertex-disjoint subpaths. Hence, the problem requires to identify and remove $p-1$ edges from G.

Theorem 1 *When G is a path, the* min-max MGGPP *can be solved in $O(n \log n)$ time.*

Proof : For a given value γ, we find a partition of P into the minimum number of components such that the gap is less than or equal to γ by scanning P and removing some edges. Starting from one end vertex of P, say v_1, edge (v_{i-1}, v_i) is removed as soon as a vertex v_i is found such that $|w_{v_i} - w_{v_j}| > \gamma$ for at least one vertex v_j in the current subpath. This procedure is then repeated starting from vertex v_i. Exploiting the general binary search approach the overall time complexity is $O(n \log n)$. □

For the other versions, the following theorem provides a polynomial approach.

Theorem 2 *When G is a path, all versions of the* MGGPP *can be solved in $O(n^2 p)$ time.*

Proof : (Sketch) We build an auxiliary directed graph with a source node u_0, p nodes $u_{j,1}, \dots, u_{j,p}$ for each vertex v_j of V, an arc from u_0 to each node $u_{j,1}$ and an arc $(u_{i,r-1}, u_{j,r})$ for each pair of vertices v_i and v_j with $i < j$ and for $r = 2, \dots, p$. The arcs correspond to candidate subpaths and their costs to the corresponding gaps. Therefore, the optimum of the *MGGPP* on G is equal to the minimum cost of a path from u_0 to $u_{n,p}$ on the auxiliary graph. Since the graph is acyclic, this problem can be solved in $O(n^2 p)$. □

2.2 Spiders

A spider is a tree with at most one vertex of degree at least 3.

Theorem 3 *When G is a spider, the* min-max MGGPP *can be solved in $O(n \log^2 n)$ time.*

Proof : We apply the procedure described in Theorem 1 to each of the d paths connecting the leaves of G to the root (the only vertex of degree $d \geq 3$), visiting G bottom-up. Let $P_1, \ldots, P_d$ be the last formed components in the partitions of such paths. We try to merge as many subpaths as possible so that their gap is $\leq \gamma$ by first ordering them w.r.t. their maximum and minimum vertex weights in $O(n \log n)$, and then checking the feasibility of the component under construction with a data structure that scans all the ordered minima and maxima in linear time. Considering the time required by the binary search, the overall time complexity is $O(n \log^2 n)$. □

2.3 Stars

A star is a connected graph with at most one vertex of degree at least 2.

Theorem 4 *When G is a star, the* MGGPP *admits only degenerate solutions, and can be solved in $O(n \log n)$ time.*

Proof : Any solution is a partition of the star where $p - 1$ components are leaves (singletons) and one component contains all the other vertices. An $O(n \log n)$ time algorithm can be obtained as follows: i) sort the leaves by nonincreasing weights; ii) visit the leaves according to such an ordering to detect a non-singleton component with minimum gap. □

2.4 Caterpillars

A caterpillar is a tree formed by a central path with n' vertices and n'' leaves attached to it.

Theorem 5 *When G is a caterpillar, the* MGGPP *can be solved in $O(n^3 p^2 \log n)$ time; the* MGGPPnd *in $O(n^2 p)$ time.*

Proof : The *MGGPPnd* amounts to partitioning the central path $\{v_1, \ldots, v_{n'}\}$. We build an auxiliary graph similar to that used in Theorem 2. An arc represents a feasible subgraph, i.e., a portion of the central path and all the attached leaves. Its cost is the gap of the subgraph. The optimal *min-sum* or *min-max* path from u_0 to $u_{n'p}$ identifies the optimal solution. Building the auxiliary graph, the cost function and detecting the optimal path take $O(n^2 p)$.

In the general case, the leaves can be isolated from the central path, but the auxiliary graph can be modified accordingly, i.e. introducing also arcs between non consecutive layers. Each arc represents a central subgraph including a portion of the central path and possibly some leaves, plus some isolated leaves. The cost of an arc is the gap of the central subgraph. For each arc, we determine the central subgraph with minimum gap adapting the $O(n \log n)$ algorithm used to solve the *MGGPP* on stars. The optimal *min-sum* or *min-max* path from u_0 to $u_{n'p}$ identifies the optimal solution. Detecting the optimal path takes $O(n^2 p^2)$, but computing the arc costs requires an overall $O(n^3 p^2 \log n)$ time. □

2.5 Complete graphs

In this case, the connectivity constraint is trivially satisfied. We sort the vertices by nondecreasing weights and rename them so that $i < j \Rightarrow w_{v_i} \leq w_{v_j}$. Then, we can restrict the search for the optimum to the solutions in which for every pair of subgraphs the vertices of one strictly precede the vertices of the other. In fact, every other feasible solution can be transformed into a non-worsening one of this family: for each pair of vertex subsets V' and V'' violating this condition, merge the two subsets and split the result in two, assigning the first $|V'|$ elements to the first subset and the last $|V''|$ elements to the second. Given the path that visits all vertices in nondecreasing weight order, the optimal solution can be detected as in Theorems 1 and 2. However, the vertex ordering allows more efficient algorithms for some cases.

Theorem 6 *When G is a complete graph, the* min-sum MGGPP *can be solved in $O(n \log n)$ time. The* min-sum MGGPPnd *can be solved in $O\left(\min\left(n\sqrt{n}\log \gamma_V, n^2 \log n, n^2 p\right)\right)$ time, where γ_V is the gap of the whole graph.*

Proof : Thanks to the ordering of the vertex weights, partitioning the auxiliary path into p subpaths of minimum total gap is equivalent to selecting $p-1$ edges such that the sum of the weight differences between their extreme vertices is maximum. This can be done by saving in a *max-heap* the weight differences between adjacent vertices and extracting from it the $p-1$ largest differences. The dominating time is given by the weight ordering.

In the nondegenerate case, it is forbidden to select two consecutive edges. The problem reduces to the search for a maximum weight matching of cardinality $p-1$ on the path. The Enhanced Capacity Scaling algorithm solves the problem in $O(n\sqrt{n}\log \gamma_V)$ time, while a reduction to the minimum cost flow problem solves it in $O(n^2 \log n)$ time. Finally, the algorithm of Theorem 2 solves it in $O(n^2 p)$ time. □

3 Conclusions and perspectives

Table 1 summarizes the special cases discussed in this paper: "NA" marks the non applicable cases (stars admit only degenerate solutions), "?" marks the open cases.

	min-max		*min-sum*	
Topology	*MGGPP*	*MGGPPnd*	*MGGPP*	*MGGPPnd*
Stars	$O(n \log n)$	NA	$O(n \log n)$	NA
Paths	$O(n \log n)$	$O(n^2 p)$	$O(n^2 p)$	$O(n^2 p)$
Spiders	$O(n \log^2 n)$	?	?	?
Caterpillars	$O(n^3 p^2 \log n)$	$O(n^2 p)$	$O(n^3 p^2 \log n)$	$O(n^2 p)$
Complete	$O(n \log n)$	$O(n^2 p)$	$O(n \log n)$	$O(min(n\sqrt{n}\log \gamma_V, n^2 \log n, n^2 p))$

TAB. 1: Summary of the computational complexity results

References

[1] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, eds. *Graph Partitioning and Graph Clustering*, v. 588 of *Contemporary Mathematics.* AMS, 2013.

[2] M. Bruglieri, R. Cordone Partitioning a graph into minimum gap components. *Electronic Notes in Discrete Mathematics*, vol. 55: 33–36, 2016.

[3] M. Bruglieri, R. Cordone, V. Caurio A metaheuristic for the minimum gap Graph Partitioning Problem. *Proceedings of CTW2017*, pp. 23–26, Cologne, Germany, June 6-8 2017.

[4] I. Lari, J. Puerto, F. Ricca, A. Scozzari Partitioning a graph into connected components with fixed centers and optimizing cost-based objective functions or equipartition criteria. *Networks*, vol. 67: 69–81, 2016.

[5] M. Maravalle, B. Simeone, R. Naldini Clustering on trees. *Computational Statistics & Data Analysis*, vol. 24: 217–234, 1997.

[6] A. Mirzaian, E. Arjomandi Selection in X+Y and matrices with sorted rows and columns. *Information Processing Letters*, vol. 20: 13–17, 1985.

[7] Li Xiao, Li Hongpeng, Niu Dongling, Wang Yan, and Liu Gang. Optimization of GNSS-controlled land leveling system and related experiments. *Transactions of the Chinese Society of Agricultural Engineering*, 31(3):48–55, 2015.

Dual bounds for a Maximum Lifespan Tree Problem

Marco Casazza Alberto Ceselli

Dipartimento di Informatica, Università degli Studi di Milano, Italy
{marco.casazza,alberto.ceselli}@unimi.it

Abstract

We investigate a Maximum Lifespan Tree Problem arising in wireless sensor networks. A network is given where each node represents a sensor; at each duty cycle they collect data, and transmit it towards a sink node through multi-hop paths on wireless links. Sensors are powered by a battery, which is partially consumed at each transmission. A tree of links spanning all the sensors must be designed such that the lifespan of the entire network, computed as the minimum lifespan among all nodes, is maximized. We study three mathematical programming formulations and we propose a comparison of the dual bounds obtained exploiting them. In particular, one of the formulations is path-based: to optimize it we design ad-hoc column generation algorithms.

Keywords : *combinatorial optimization, spanning tree, column generation, network design, dual bounds, mobile sensors network.*

1 Introduction

A wireless sensor network is composed by a set of low-energy, battery-powered sensor nodes (sensors in the remainder). Sensor networks find many applications [4]; for instance they are used to collect data from areas having no existing infrastructures, to a single edge node connected to a backbone (sink in the remainder). Sensors connect one another by means of wireless links of limited range: if the data collection area is wide, the underlying communication network is sparse, preventing a direct connection of all sensors to the sink. Typical protocols from the literature assume each sensor to forward data with a shortest path philosophy, always to a single parent sensor [1] [5]. Therefore, for each sensor, a route (that is a sequence of sensors) connecting to the sink must be found, such that each sensor of the route has both its predecessor and its successor within the range of its wireless links, thereby allowing transmissions to be possible. The set of all routes origins a spanning tree rooted in the sink node.

Sensors have clocks, making them to work in duty cycles: at each cycle the sensor receives data by the set of its children sensors, and transmits data to its parent. Since batteries replenishment is usually impossible or very expensive, the conservation of the battery energy is a critical issue. While sensors consumes a negligible amount of energy when in standby, they spend most of the energy during both transmitting and receiving operations. It follows that the selection of the routes is critical, especially for those sensors belonging to several routes. A combinatorial problem arises, that is finding the spanning tree maximizing the lifetime of the network, defined as the number of duty cycles that can be performed without making any sensor run out of battery.

In the literature such a problem is indeed known as *Maximum Lifespan Tree Problem (MLTP)*. It has been studied in [1], where the authors prove the NP-completeness of the problem and provide a routing algorithm with worst case constant factor performance guarantees. In [2] an approximation algorithm is proposed for a version of the problem where data is not aggregated during transmission. In [5] and [3] two exact algorithms are presented to solve both the MLTP with and without the aggregation of the messages. However, both algorithms are strongly based on a technique that decomposes the network in independent and smaller networks when there exist links that, if removed, leave the network disconnected. It follows that their methodology is strongly dependent from the topology of a network.

In this paper we propose three different mathematical programming formulations for MLTP. We then compare the dual bound which can be obtained by each formulation. Two are compact, and allow to use cutting plane procedures of general purpose solvers. The third one is path-based, requiring to rely on column generation: we study the corresponding pricing problem and we provide ad-hoc algorithms for it.

2 Mathematical formulations

We model the MLTP as a problem on a directed graph $G = (N_0, A)$ given, where $N_0 = \{0, 1, \ldots, n\}$ is a set of nodes, representing the n sensors and the sink 0, while $A \subseteq N_0 \times N_0$ is a set of arcs, each representing a link whose range allows communication between the corresponding sensors. We also define as $N = N_0 \setminus \{0\}$ the set of sensor nodes only. For each node $i \in N_0$ we are also given a battery capacity $b_i > 0$, that we assume infinite for the sink ($b_0 = +\infty$). We assume messages of equal size at each duty cycle. Every time a message is transmitted between sensors, the transmitter consumes $e_t > 0$ energy units of its battery, while the receiver consumes $e_r > 0$. No sensor can send or receive messages if its battery is empty, and we define as *lifespan* of a node the number of life cycles before the battery of the corresponding sensor is fully drained.

A solution to MLTP is a spanning tree rooted in 0 that connects all nodes in N_0. A solution is also optimal if it maximizes the lifespan of the tree, computed as the minimum lifespan over all nodes, that is

$$\text{maximize} \min_{i \in N} \left\lfloor \frac{b_i}{(e_t + e_r) \cdot \text{children}(i) + e_t} \right\rfloor \tag{1}$$

where children(i) is a function computing the number of children of node i in the spanning tree. The MLTP can be formulated, adapting [5], as follows:

$$(CM) \begin{cases} \text{maximize} \quad \min_{i \in N} \left\lfloor \dfrac{b_i}{(e_t + e_r) \cdot y_i + e_t} \right\rfloor & & (2) \\ \text{s.t.} \quad \sum_{(j,i) \in A} z_{ji} = \begin{cases} 1 & \text{if } i \neq 0 \\ 0 & \text{otherwise} \end{cases} & \forall i \in N_0 & (3) \\ x_{ij} \geq y_j + 1 - |N| \cdot (1 - z_{ij}) & \forall (i,j) \in A & (4) \\ y_i = \sum_{(i,j) \in A} z_{ij} & \forall i \in N_0 & (5) \\ z_{ij} \in \mathbb{B}, \; x_{ij} \in \mathbb{N}_0 & \forall (i,j) \in A & (6) \\ y_i \in \mathbb{N}_0 & \forall i \in N_0 & (7) \\ l \geq 0 & & (8) \end{cases}$$

each variable z_{ij} is set to 1 if arc (i, j) is selected in the tree, and 0 otherwise, variable x_{ij} is the number of messages travelling arc (i, j) at each life cycle. Each variable y_i is the number of descendants of node i in the tree, that is the number of messages collected by i at each duty cycle. The objective function (2) maximizes the minimum lifespan. Constraints (3) ensure that each node but the root has exactly one incoming arc. When $z_{ij} = 0$, constraints (4) have no effect; when $z_{ij} = 1$, they enforce x_{ij} to take the number of messages going through arc (i, j). Constraints (5) force y_i to be set to the number of descendants of node i. Subtours are avoided by the combination of (4) and (5).

As stressed in [5], such a formulation is nonlinear because the objective functions include (a) a `min` operator (b) a `floor` operator (c) variables at the denominator. However, we observe that

$$\text{maximize} \min_{i \in N} \left\lfloor \frac{b_i}{(e_t + e_r) \cdot y_i + e_t} \right\rfloor = \left\lfloor \left(\text{maximize} \min_{i \in N} \frac{b_i}{(e_t + e_r) \cdot y_i + e_t} \right) \right\rfloor \tag{9}$$

We therefore propose to obtain an equivalent linear formulation by introducing a continuous variable l, and the following set of constraints $l \geq ((e_t + e_r) \cdot y_i + e_t)/b_i$ for all $i \in N$, and by changing the objective function to minimize l.

2.1 Flow formulation

We now propose an improved formulation, based on classic flow problems: let $G' = (N'_0, A')$ be a directed graph that extends G with an additional terminal node t, and where $N'_0 = N_0 \cup \{t\}$ and $A' = A \cup \{(n,t), \forall n \in N\}$. The MLTP can be formulated as follows:

$$
(FM)\begin{cases}
\text{minimize} \quad l & & (10)\\
\text{s.t.} \quad \sum_{(i,j)\in A'} x_{ij} - \sum_{(j,i)\in A'} x_{ji} = \begin{cases} |N| & \text{, if } i = 0 \\ -|N| & \text{, if } i = t \\ 0 & \text{, otherwise} \end{cases} & \forall i \in N'_0 & (11)\\
l \geq ((e_t + e_r) \cdot (\sum_{(j,i)\in A'} x_{ji} - 1) + e_t)/b_i & \forall i \in N & (12)\\
x_{ij} \leq \begin{cases} 1 & \text{if } j = t \\ |N| \cdot z_{ij} & \text{otherwise} \end{cases} & \forall (i,j) \in A' & (13)\\
\sum_{(j,i)\in A'} z_{ji} \leq 1 & \forall i \in N_0 & (14)\\
x_{ij} \geq 0, \ z_{ij} \in \mathbb{B} & \forall (i,j) \in A' & (15)\\
l' \geq 0 & & (16)
\end{cases}
$$

where each variable x_{ij} is the flow traversing arc (i,j), and each variable z_{ij} is set to 1 if any unit of flow traverses arc (i,j), and 0 otherwise. The objective function (10) still minimises the reciprocal of the lifespan. Constraints (11) ensure flow conservation. Constraints (12) force l to take the maximum reciprocal of the nodes lifespan. Constraints (13) impose that no flow can traverse (i,j) unless $z_{ij} = 1$ (except for the terminal). Constraints (14) impose single source conditions.

2.2 Path formulation

Finally, we introduce an extended formulation. Let R be the set of elementary paths connecting the root to any node of the graph, where each path $r \in R$ has a pattern $(\bar{\mathrm{w}}^r, \bar{\mathrm{f}}^r) \in \mathbb{B}^{|N|} \cdot \mathbb{N}^{|N|}$, where $\bar{w}^r_i$ is 1 if node i is visited in path r, and 0 otherwise, and $\bar{f}^r_i$ is the number of nodes visited after node i in path r. The MLTP can be formulated as follows:

$$
(PM)\begin{cases}
\text{minimize} \quad l & & (17)\\
\text{s.t.} \quad \sum_{r\in R} \bar{w}^r_i \cdot \gamma^r \geq 1 & \forall i \in N & (18)\\
((e_t + e_r) \cdot \sum_{r\in R} \bar{f}^r_i \cdot \gamma^r + e_t)/b_i \leq l & \forall i \in N & (19)\\
\gamma^r \in \mathbb{B} & \forall r \in R & (20)
\end{cases}
$$

where each variable γ^r is set to 1 if path r is selected in the solution and 0 otherwise. The objective function (17) minimises the reciprocal of the lifespan. Constraints (18) impose that each node is visited at least once, while constraints (19) force l to take the maximum reciprocal of the lifespans.

Indeed our model has a number of variables that grows exponentially in the number of the nodes. Therefore, we recur to column generation techniques to solve its continuous relaxation: at each iteration of the column generation process we search for a minimum reduced cost path

instances		CM		FM		PM	
nodes	arcs	ratio (%)	time (s)	ratio (%)	time (s)	ratio (%)	time (s)
31	105	320.78	0.02	5.46	0.01	0.79	0.00
36	139	568.59	0.03	4.78	0.02	2.42	0.00
41	182	375.56	0.03	2.42	0.03	0.00	0.01
46	213	586.73	0.04	3.50	0.05	0.00	0.06
51	258	633.67	0.04	1.24	0.05	0.00	0.11
56	326	661.42	0.05	0.59	0.08	0.00	0.25

TAB. 1: Average ratios between the dual bound of a formulation and the best dual bound found by any formulation.

from the root node to any other node of the graph; the corresponding pricing problem is an optimization problem where for each node we are given a profit λ_i when node i is visited and a cost $(e_t + e_r) \cdot \mu_i$ paid for any node visited after i. For solving solve such a problem we devised a dynamic programming algorithm (whose description is omitted).

3 Experimental analysis and conclusions

We implemented our formulations using CPLEX 12.6.3 for both CM and FM, and using an ad-hoc column generation procedure in C++ for PM. CM can be considered as a benchmark, being a slight improvement on those used in previous works like [3]. As a dataset we used instances generated in [5] and [3], with at most 56 nodes. Each instance of the dataset differs for number of nodes $|N_0| \in \{31, 36, 41, 46, 51, 56\}$ and number of arcs. All tests have been run on a PC equipped with Intel(R) Core(TM) i7-6700K CPU and 32 GB or memory.

In Table 1 we report the average results obtained by the cutting planes procedure of CPLEX at the root node of the branching tree for both CM and FM formulations, and the one obtained by the continuous relaxation of PM. For each size of the graph, we report the average number of arcs and, for each formulation, the average ratio between the dual bound provided by a formulation and the best dual bound found by any of them and the computing time to compute such dual bound.

On average PM provides the best dual bounds, while the one provided by CM is always the worst. FM offers a compromise, being as fast as CM, but much more accurate. Combining FM and PM seems promising to target the design of exact algorithms.

References

[1] C. Buragohain, D. Agrawal, and S. Suri. Power aware routing for sensor databases. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1747–1757 vol. 3, March 2005.

[2] J. Liang and T. Li. A maximum lifetime algorithm for data gathering without aggregation in wireless sensor networks. *Applied Mathematics & Information Sciences*, 7(5):1705, 2013.

[3] X. Ma, X. Zhu, and B. Chen. Exact algorithms for maximizing lifetime of wsns using integer linear programming. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, March 2017.

[4] E. Tsiontsiou, B. Addis, Y. Q. Song, and A. Ceselli. Optimal probabilistic energy-aware routing for duty-cycled wireless sensor networks. In *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–7, Nov 2016.

[5] X. Zhu, X. Wu, and G. Chen. An exact algorithm for maximum lifetime data gathering tree without aggregation in wireless sensor networks. *Wirel. Netw.*, 21(1):281–295, January 2015.

A star-based reformulation for the maximum quasi-clique problem

Fabrizio Marinelli[1], Andrea Pizzuti[1], Fabrizio Rossi[2]

[1] Università Politecnica delle Marche, Ancona, Italy,
fabrizio.marinelli@univpm.it, a.pizzuti@pm.univpm.it

[2] Università degli Studi dell'Aquila, L'Aquila, Italy,
fabrizio.rossi@univaq.it

Abstract

Given a simple and undirected graph, the maximum γ-quasi-clique problem consists in identifying the induced subgraph of maximum order and edge density of at least γ. In this paper we propose a new extended formulation for such a problem obtained by decomposing *star* inequalities. Preliminary computational results assess the quality of the continuous relaxation with respect to the tightest formulation available in the literature.

Keywords : *quasi-clique, mixed integer programming, integer reformulation.*

1 Introduction

Given a simple and undirected graph $G = (V, E)$, the well-known maximum clique problem (MCP) consists in finding a maximum cardinality *clique* of G, i.e., a complete induced subgraph of G of maximum order [3]. Since a clique provides a natural and ideal measure of cohesion, a large number of applications where the degree of interaction between entities is of interest, e.g., social network analysis, coding theory, telecommunication and genetics just to mention a few, require, at least in principle, the solution of the MCP. Indeed, the search for a overly structure as a clique often prevents the discovery of worthy *dense* subgraph, and therefore several clique relaxations have been taken into account and related discrete optimization problems studied in the literature [4]. Among them, a pair of complementary (NP-hard) problems are the *maximum quasi-clique problem* (γ-QCP) [5] and the *k-densest subgraph problem* (KDSP), the former searching for the maximum-order induced subgraph of G with edge density of at least γ, the latter the maximum-size induced subgraph of G of order k. To the best of our knowledge, the reference exact methods for γ-QCP and KDSP are the MIP-based approach presented in [6] and the branch-and-bound algorithm with semidefinite bounding procedure by [1], respectively.

In this paper we propose an integer reformulation $[D_\gamma]$ of γ-QCP and a surrogate relaxation $[D^S_\gamma]$ of $[D_\gamma]$. The former improves the quality of the dual bound provided by the tightest formulation available in the literature. The latter, although being a surrogate relaxation, it has almost the same dual bound of $[D_\gamma]$ but contains a number of constraints linear in $V(G)$. Thus, it can be exploited on instances with large and dense graphs.

2 Problem reformulation

Let $H = (Q, E_Q) = G[Q]$ be the subgraph of G induced by the set of nodes $Q \subseteq V$. An optimal solution of γ-QCP is an induced subgraph H of maximum order $|Q^*|$ and at least $|E_{Q^*}| = \gamma\binom{|Q^*|}{2}$ edges. Veremyev et al. [6] propose four MILP formulations for γ-QCP, the tightest of which, reported in the following, consists of $O(|V| + |E|)$ variables and $O(|E|)$ constraints. Let x_i, $i \in V$, and z_e, $e \in E$, be binary variables with $x_i = 1$ iff $i \in Q$, and $z_e = 1$

iff $e \in E_Q$. Let moreover y_k, $k \in K = \{k_L, \ldots, k_U\}$, be the binary variable with $y_k = 1$ if H is of order k. The formulation reads as

$$
\begin{aligned}
[C_\gamma]: \qquad & |Q^*| = \max \sum_{i \in V} x_i && && (1)\\
& z_e \le x_i, \quad z_e \le x_j && \forall e = \{i, j\} \in E && (2)\\
& \sum_{i \in V} x_i \le \sum_{k \in K} k y_k && && (3)\\
& \sum_{k \in K} y_k = 1 && && (4)\\
& \gamma \sum_{k \in K} \frac{k(k-1)}{2} y_k \le \sum_{e \in E} z_e && && (5)\\
& x_i, z_e, y_k \in \{0, 1\} && \forall i \in V, \forall e \in E, \forall k \in K && (6)
\end{aligned}
$$

Edge $e = \{i, j\}$ belongs to the γ-quasi-clique H if (and only if) nodes i and j are both in Q, see constraints (2). The order k of H is defined by constraints (3) and (4), and its density is bounded from below to γ by constraint (5). A lower bound k_L to $|Q^*|$ is given by the order of any clique of G. On the other hand, a basic upper bound k_U to $|Q^*|$ is $\left\lfloor \frac{1}{2} + \frac{1}{2}\sqrt{1 + \frac{8|E|}{\gamma}} \right\rfloor$, see [5]. At the expense of some negligible additional computation, a better upper bound $\bar{k}_U$ can be obtained as follows. Any γ-quasi-clique $H = (Q, E_Q)$ fulfils by definition $|Q|(|Q| - 1) \le 2\frac{|E_Q|}{\gamma}$. Moreover, $|E_Q| = \sum_{i \in Q} \frac{d_i^H}{2} \le \sum_{i \in Q} \frac{\min\{|Q|-1, d_i\}}{2}$, where d_i is the degree of i in G and d_i^H is the degree of i in H. Therefore

$$
|Q|(|Q| - 1) \le \frac{1}{\gamma} \sum_{i \in Q} \min\{|Q| - 1, d_i\} \le \frac{1}{\gamma} \sum_{i=1}^{|Q|} \min\{|Q| - 1, d_i\}
$$

where the last inequality holds if nodes of G are sorted by non-increasing degrees, i.e., $d_i \ge d_j$ for $i < j$. It easy to see that the largest integer Q that satisfies the above inequality is a valid upper bound $\bar{k}_U$ for $|Q^*|$, and that $\bar{k}_U < k_U$ for the $|Q^*| < |V|$.

Model $[C_\gamma]$ can be reformulated by integer decomposition. Let $S(i)$ be the *star* of $i \in V$, i.e., the set of incidence edges of node i. Clearly, the *star constraint*

$$
\sum_{e \in S(i)} z_e \le (k_U - 1) x_i \tag{7}
$$

is a valid inequality of $[C_\gamma]$ for each node i. The set of integer points that satisfy (7) corresponds to the set S_i of all the partial stars $\{S_{i1}, S_{i2}, \ldots\}$ of i with less than k_U edges. Let $S_i^j \subset S_i$ be the partial stars including the edge $\{i, j\} \in E$. Then for any $e = \{i, j\} \in E$, the variables x_i and z_e of $[C_\gamma]$ can be rewritten as

$$
x_i = \sum_{h \in S_i} \lambda_{ih} \qquad z_e = \sum_{h \in S_i^j} \lambda_{ih} \qquad \sum_{h \in S_i} \lambda_{ih} = 1 \qquad \lambda_{ih} \in \{0, 1\},
$$

resulting to the following extended MILP formulation:

$$[D_\gamma]: \qquad |Q^*| = \max \sum_{i\in V}\sum_{h\in S_i} \lambda_{ih} \tag{8}$$

$$\sum_{h\in S_i} \lambda_{ih} \le 1 \qquad \forall i \in V \tag{9}$$

$$\sum_{h\in S_i^j} \lambda_{ih} - \sum_{h\in S_j^i} \lambda_{jh} = 0 \qquad \forall e = \{i,j\} \in E \tag{10}$$

$$\sum_{i\in V}\sum_{h\in S_i} \lambda_{ih} \le \sum_{k\in K} k y_k \tag{11}$$

$$\sum_{k\in K} y_k = 1 \tag{12}$$

$$\sum_{k\in K} \lceil \gamma k(k-1)\rceil y_k - \sum_{i\in V}\sum_{h\in S_i} |S_{ih}|\lambda_{ih} \le 0 \tag{13}$$

$$\lambda_{ih} \in \{0,1\} \qquad \forall i \in V, h \in \{1,\dots,|S_i|\} \tag{14}$$

$$y_k \in \{0,1\} \qquad \forall k \in K \tag{15}$$

Variable λ_{ih} is equal to 1 if the partial star S_{ih} is selected, and 0 otherwise. At most one partial star can be selected for each node, see constraint (9), and the selection of partial stars must be consistent: if a partial star S_{ih} is chosen and S_{ih} contains the edge $\{i,j\}$, then a partial star S_{jp} including the edge $\{j,i\}$ must be selected too, see *balance* constraint (10). To this purpose, the sign of the variable λ_{ih} is positive for edges $\{i,j\} \in S_{ih}$ with $i < j$ and negative otherwise.

The dual bound provided by the continuous relaxation of $[D_\gamma]$ is at least tight as that computed by $[C_\gamma]$ because $[D_\gamma]$ is obtained by integer reformulation of $[C_\gamma]$. However, $[D_\gamma]$ consists of an exponential number in $|E|$ of variables and therefore its solution requires a column generation approach, where columns correspond to attractive partial stars and pricing problems are solved for each node i. In particular, let $\sigma_i, \delta, \psi \in \mathbb{R}^+$ and $\pi_e \in \mathbb{R}$ be the values of dual variables associated to constraints (9),(11),(13) and (10), respectively, and w_e a binary variable equal to one if the edge e belongs to the partial star. The reduced cost of the most profitable partial star of node i is

$$1 - \sigma_i - \delta + \max \sum_{e\in S(i)} (\psi + \Gamma(e)\pi_e) w_e \tag{16}$$

with $\Gamma(e = \{i,j\}) = -1$ if $i < j$ and $\Gamma(e) = 1$ otherwise. The pricing problem looks for the set of at most $k_U - 1$ edges that maximize (16), a set that can be obtained by ranking the edges of $S(i)$ by non decreasing values of $(\psi + \Gamma(e)\pi_e)$.

Formulations $[C_\gamma]$ and $[D_\gamma]$ are not suitable for solving the γ-QCP on dense graphs due their $O(|E|)$ constraints; even the continuous relaxation can be difficult to solve for moderate size instances. However, at the cost of a small loss in the quality of dual bounds, a surrogate relaxation $[D_\gamma^S]$ of $[D_\gamma]$ can be considered by replacing constraints (10) with the following ones:

$$\sum_{h\in S_i} |S_{ih}|\lambda_{ih} - \sum_{j\in S(i)}\sum_{h\in S_j^i} \lambda_{jh} = 0 \qquad \forall i \in V \tag{17}$$

The pricing problem of $[D_\gamma^S]$ can be easily derived from (16) by appropriately replacing the dual variables π_e with those associated to the surrogate constraints (17).

3 Preliminary computational findings

The column generation algorithm was coded in C++ and all the linear programs were solved by IBM® CPLEX® 12.5.0.0 on a Intel® Core2 Duo E8500 3.16 GHz machine with 8Gb RAM.

Experiments were performed on two groups of benchmark instances: 16 sparse graphs from [6] (mean density = 0.008) and the 64 DIMACS instances [2] (mean density = 0.621). All tests are made with $\gamma = 0.9$, $k_L = 2$ and CPU time limit to 4 hours.

We compared formulation $[D^S_\gamma]$, i.e., the surrogate relaxation of $[D_\gamma]$, with $[C_\gamma]$. We observed that the continuous relaxation of $[D^S_\gamma]$ is solved much faster and is definitively more scalable: on sparse graphs the required CPU time is on average 48.49% less than that spent for solving the continuous relaxation of $[C_\gamma]$. Moreover, CPLEX reaches the time limit on 12 out of 64 DIMACS instances and takes 899.46 sec. on average on the remaining 52 graphs when used to solve the continuous relaxation of $[C_\gamma]$, whereas the column generation computes the bounds in 2.23 sec. on average. On the other hand, in both class of instances the dual bound provided by $[D^S_\gamma]$ is roughly the same of that computed by $[C_\gamma]$: on sparse graphs $[D^S_\gamma]$ is slightly worse than $[C_\gamma]$ (1.32% on average) whereas $[D^S_\gamma]$ performs slightly better (0.14% on average) on dense graphs. Finally, though we assessed the tightness of formulation $[D_\gamma]$ on small instances, the current implementation of the column generation lacks of some technicalities, e.g., early termination and stabilization strategies, and therefore running times are not yet competitive for instances with more than 100 nodes.

4 Conclusions and perspective

In this paper a new extended MILP formulation $[D_\gamma]$ for the γ-QCP has been presented and some preliminary test were made to evaluate its performance. The bound provided by $[D_\gamma]$ is as good as that computed by the tightest formulation reported in the literature and experiments show that also the surrogate relaxation $[D^S_\gamma]$ roughly provide the same bounds. However, $[D^S_\gamma]$ seems to be more scalable since the column generation procedure is much faster especially on dense graphs. In perspective, we look at the implementation of a full branch-and-price procedure to solve the γ-QCP to optimality.

References

[1] Krislock, N., Malick, J. and Roupin, F.: *Computational results of a semidefinite branch-and-bound algorithm for k-cluster.* Computers & Operations Research, 66: 153–159, 2016.

[2] Rossi, R.A. and Ahmed, N.K.: *The network Data Repository with Interactive Graph Analytics and Visualization.* Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, `http://networkrepository.com`.

[3] Pardalos, P.M. and Xue, J.: *The maximum clique problem.* Journal of Global Optimization. Optim. 4(3): 301–328, 1994.

[4] Pattilo, J., Youssef, N. and Butenko, S.: *On clique relaxation models in network analysis.* European Journal of Operational Research. 226(1): 9–18, 2013.

[5] Pattilo, J., Veremyev, A., Butenko, S. and Boginski, V. *On the maximum quasi-clique problem.* Discrete Applied Mathematics. 161: 244–257, 2013.

[6] Veremyev, A., Prokopyev, O.A., Butenko, S., and Pasiliao, E.L.: *Exact MIP-based approaches for finding maximum quasi-cliques and dense subgraphs.* Computational Optimization and Applications, 64: 177–214, 2016.

A hybrid heuristic for multi-activity tour scheduling

Stefania Pan[1][2], Mahuna Akplogan[2], Lucas Létocart[1],
Louis-Martin Rousseau[3], Nora Touati[2], Roberto Wolfler Calvo[1]

[1] LIPN, UMR 7030 CNRS, Université Paris 13,
99 avenue Jean-Baptiste Clément 93430, Villetaneuse, France
pan@lipn.fr

[2] Horizontal Software, 9 Rue de l'Isly, 75008 Paris, France.

[3] CIRRELT - Polytechnique Montréal, CP6079 Succ Centre Ville,
Montréal, Canada, H3C3A8.

Abstract

This work investigates the tour scheduling problem with a multi-activity context, a challenging problem that often arises in personnel scheduling. We propose a hybrid heuristic, which combines tabu search and large neighborhood search techniques. We present computational experiments on weekly time horizon problems dealing with up to five work activities. The results show that the proposed approach is able to find good quality solutions.

Keywords : *Tabu search, large neighborhood search, multi-activity tour scheduling.*

1 Introduction

Personnel scheduling problems consist of assigning employees to activities over a given time horizon, taking into account organizational, legal and social constraints. Personnel scheduling arises in different organizations. The specific requirements of different companies result in quite diverse models and solution methods. Recently, Van den Bergh et al. [5] present a comprehensive survey. This work focuses on the multi-activity tour scheduling problem, where each employee has to be assigned to a schedule that covers a time horizon of one week. Furthermore, for every time period, it must be specified if the employee is working on an activity, having a break, or a rest. Under and over coverage are considered and minimized in the objective function. We propose a hybrid heuristic which combines Tabu Search (TS) and Large Neighborhood Search (LNS). The first technique builds an initial solution satisfying workload requirements, and aims at integrating a particular class of legal constraints by keeping the demand satisfied. The second technique completely repairs all schedules making them feasible for all legal constraints, and aims at minimizing under and over coverage. The literature exhibits a wide range of models and solution techniques for solving personnel scheduling problems [5].

2 Problem Definition

The main elements and constraints of the multi-activity tour scheduling problem are presented in Table 1. The problem can be modeled as a mixed-integer linear program. Let E be the set of employees, A the set of work activities, and J the set of slots covering the whole time horizon. We define x_{eja} and x_{ejb} the decision variables taking value 1 if employee e is assigned to activity a, respectively to break b in slot j. Furthermore, let $x_{ej\bar{a}}$ and x_{ejs} be the decision variables taking value 1 if employee e is working, respectively on duty, in slot j. Due to lack of space we do not report the complete formulation, we only remark a particular class of constraints that will be considered later.

TAB. 1: Elements and constraints of the multi-activity tour scheduling

Main elements	Constraints
Slots: time periods of equal length that divide the time horizon.	**Workload**: a number b_{ja} of employees is required for slot j and activity a.
Timeslot: sequence of activities performed consecutively without breaks.	**Legal**: (L1) Duration of activity $a \in [l_a, u_a]$.
Daily shift: one or more timeslots assigned the same day and divided by a break. During these slots, the employee is said to be on duty.	(L2) Consecutive working time $\in [l_w^c, u_w^c]$. (L3) Daily working time $\in [l_w^d, u_w^d]$. (L4) Duration of breaks $b \in [l_b, u_b]$. (L5) Amplitude of daily shift $\in [l_s, u_s]$.
Schedule: sequence of working days and days-off covering all the time horizon.	(L6) Weekly working time $\in [l_w^w, u_w^w]$. (L7) Rest between two daily shifts $\geq l_r$.

$$x_{eja} + (z - \sum_{j'=j+1}^{j+z} x_{ej'a}) + x_{e(j+z+1)a} \geq 1, \qquad \forall k, j, a \in A \cup \{\bar{a}, b, s\}, \forall z \in \{1, ..., l_a - 1\}, \qquad (1)$$

$$\sum_{j'=j}^{j+u_a} x_{ej'a} \leq u_a, \qquad \forall e, j, a \in A \cup \{\bar{a}, b, s\}. \qquad (2)$$

Constraints (1)-(2) impose min and max restrictions on consecutive assignments of activities. These constraints include activities duration (L1) when $a \in A$, consecutive working time (L2) (resp. break duration (L4), amplitude of daily shift (L5)), when we consider $\bar{a}$ (resp. b, s).

3 A heuristic approach

This section describes the hybrid heuristic proposed. It essentially combines tabu search [1] (TS) and large neighborhood search [4] (LNS). Starting from an initial solution satisfying workload requirements, TS aims at integrating constraints (1)-(2) by keeping the demand satisfied. Then, LNS completely repairs all schedules making them feasible, and aims at minimizing the under and over coverage.

Tabu Search. In the first phase, an initial greedy solution is built by assigning employees to activities in order to satisfy workload demand, without checking violations to legal constraints. Then, TS aims at integrating the particular class of constraints defined by (1) and (2), which impose restriction on the consecutive assignment of activities. This heuristic explores the solution space beyond local optimality, and it uses the notions of movement, neighbourhood and tabu list. In the following we detail these features.

Neighborhood. The neighborhood of a solution is defined by the operator `Swap`. Given a subset of consecutive slots $J' = \{j, j+1, \dots\}$ and two employees e_1 and e_2, the solution obtained after applying `Swap` $(e_1, J', a_1) \rightarrow (e_2, J', a_2)$ is equal to the current solution except that employees e_1 and e_2 exchange their activities in all the slots j in J'. The neighborhood $N(p)$ of a solution p consists of all the solutions we can achieve by applying a `Swap` move on a subset J' of slots of the most violated constraint. We remark that `Swap` moves ensure that all the solutions in the neighborhood keep on satisfying workload constraints.

Tabu list. We employ a dynamic tabu list. We fix minimum and maximum lengths. The tabu list increases when the best solution known does not improve after 10 iterations, while it decreases when the best solution known improves. It is updated using a FIFO policy.

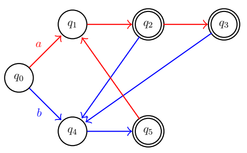
The basic TS is combined with *intensification* and *diversification*. The first deeply explores $N(p)$: when an improving `Swap` is found, moves in adjacent slots are evaluated and eventually applied. The second employs a perturbation operator when the best solution cannot be improved within 100 iterations. Starting from the best solution known, it applies `Swap`s in all slots with duration constraint violation. As a result, the new solution preserves part of the best solution feasibility and differs where duration constraints are violated. Then, the basic tabu search with intensification is restarted. Diversification is performed 2 times.

Large Neighborhood Search. In the second phase, we have a solution that satisfies workload and has a low violation to constraints (L1), (L2), (L4), and (L5). We employ LNS to integrate completely all legal constraints. For each employee, a new schedule is built in order to be as close as possible to the current assigned schedule. The LNS technique iteratively destroys part of the solution and repairs it in the hope of finding a better solution. Similarly to [3], destroying here means choosing an employee and removing his schedule, while repairing means assigning a new schedule to the selected employee for improving the global solution.

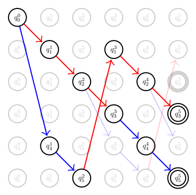
In the following, we explain how automata can model the schedule's rules and can build new feasible schedules. Firstly, activities are combined into timeslots, which are in turn used to build daily shifts. Finally, daily shifts are combined to build feasible schedules. Legal constraints are considered both by means of automata, and by solving resource constrained shortest path problems on the expanded graphs. We refer to [2] for a review on the notions of *automata* and *expanded graph*.

Build feasible timeslots. Timeslots are defined by constraints (L1) and (L2), and they can be modeled by means of an automaton as show in Fig. 1a. Using this automaton, we build the expanded graph in Fig. 1b. All paths from node q_0^0 to a terminal node describe feasible timeslots where the employee works 5 consecutive slots. We build timeslots for all possible starting periods and all possible durations in $[l_w^c, u_w^c]$. The goal is to have a pool of feasible timeslots to be combined into daily shifts.

FIG. 1: Automaton 1a and expanded graph 1b to build timeslots



(a) Automaton accepting all timeslots with activities a and b, with duration respectively in [2, 3] and fixed to 2.



(b) Expanded graph associated with the automaton in Figure 1a.

Build feasible daily shifts. Daily shifts are defined by constraints (L3), (L4) and (L5). Similarly to timeslots, they can be modeled by means of an automaton that essentially combines timeslots with breaks. The only difference here is that we need to introduce two monotone resources (the total working time r_w^d and the amplitude of the daily shift r_s) for each path from node q_0^0 to a terminal node in the expanded graph. Therefore, not all the resource constrained paths are feasible, but only the ones such that $r_w^d \in [l_w^d, u_w^d]$ and $r_s \in [l_s, u_s]$. We build daily shifts for each day, all possible starting slots and amplitude. The goal is to have a pool of feasible daily shifts to be combined into schedules.

Build feasible schedule. The constraints defining schedules are (L6) and (L7). They are modeled by a directed acyclic graph G that combines daily shifts and days off over the time horizon. For every day of the week, graph G has an associated level containing one node for the day off, and one node for each daily shift. Nodes of consecutive levels are connected by an edge if the rest period is satisfied. We introduce one resource r_w^w (weekly working time). Schedules are resource constrained paths from the source node to the sink node.

We define the *cost* of a schedule s associated to employee e as the sum of all activities costs: $c_s^e = \sum_{j \in J} \sum_{a \in A} c_{ja} x_{eja}$. We recall that after the TS phase, we have a solution that fulfill the demand and partially satisfies legal constraints. The goal of the LNS is to build, for each employee, a new feasible schedule that minimizes under-over coverage, by keeping the schedule as close as possible to the already assigned one. To do this, we select an employee e and we set $c_{ja} = -1$ for all (j, a) such that either $x_{eja} = 1$ or is under covered, otherwise $c_{ja} = 1$. They are used to build feasible timeslots of the minimum cost, by associating c_{ja} to the corresponding arc of the timeslots expanded graph (Fig. 1b) and solving a shortest path problem. Then, the costs of the timeslots obtained are given to the arcs of the daily shifts expanded graph, and minimum cost daily shifts are built solving a resource constrained shortest path. Finally, the costs of the daily shifts are associated to the arcs of the directed acyclic graph G, and the minimum cost schedule is built solving a resource constrained shortest path.

4 Computational results

We perform the computational experiments on random instances generated by varying the set of activities A (2 to 5), the number of employees $|E|$ (10 to 60) and the slot time unit (15 and 30 minutes). Workload demand is inspired by input from quick service restaurants, defined so that an optimal solution without under and over coverage exists. Legal constraints are set as follows: (L1) l_a and u_a are respectively between 1h and 2h, and 3h and 4h; (L2) l^c_w and u^c_w are 2h and 6h; (L3) l^d_w and u^d_w are 0h and 10h; (L4) l_b and u_b are 30min and 1h; (L5) l_s and u_s are 0h and 12h; (L6) l^w_w and u^w_w are 0h and 35h; (L7) l_r is 11h. Instances are labeled with the format `E_A`, where `E` and `A` represent the number of employees and activities respectively. The tests were performed on Intel Xeon CPU E3-1220 v5 @3.00GHz 4GB of RAM. The heuristic was implemented using C#. The mixed integer linear program has been solved using CPLEX 12.7.0, which does not manage to solve any instance within 1 hour. The following table reports the name of the instances (*Inst*), which are solved both with slots of *30min* and *15min*. For each instance and for both solving phases *TS* and *LNS*, we show the violation to legal (v_L) and workload (v_W) constraints. More precisely, v_L represents the legal

Inst	30min					15min					Inst	30min					15min				
	TS		LNS		time(s)	TS		LNS		time(s)		TS		LNS		time(s)	TS		LNS		time(s)
	$v_L\%$	$v_W\%$	$v_L\%$	$v_W\%$		$v_L\%$	$v_W\%$	$v_L\%$	$v_W\%$			$v_L\%$	$v_W\%$	$v_L\%$	$v_W\%$		$v_L\%$	$v_W\%$	$v_L\%$	$v_W\%$	
10_2	0.0	0.0	0.0	4.8	33.9	0.0	0.0	0.0	3.5	251.6	40_2	1.4	0.0	0.0	4.4	224.7	1.3	0.0	0.0	4.6	1245.4
10_3	0.0	0.0	0.0	4.6	30.3	0.0	0.0	0.0	3.8	217.2	40_3	0.4	0.0	0.0	5.1	199.8	0.4	0.0	0.0	4.8	1173.6
10_4	0.0	0.0	0.0	5.6	31.3	0.0	0.0	0.0	3.7	222.7	40_4	1.4	0.0	0.0	4.8	213.9	0.6	0.0	0.0	5.1	1089.3
10_5	0.0	0.0	0.0	1.2	29.8	0.0	0.0	0.0	0.8	211.6	40_5	0.7	0.0	0.0	4.0	222.1	0.7	0.0	0.0	4.0	1133.6
20_2	0.8	0.0	0.0	4.4	81.0	0.7	0.0	0.0	4.7	575.9	50_2	1.3	0.0	0.0	4.2	307.6	1.3	0.0	0.0	4.3	1629.2
20_3	0.3	0.0	0.0	4.9	90.3	0.0	0.0	0.0	5.9	501.9	50_3	1.0	0.0	0.0	4.3	238.2	0.9	0.0	0.0	5.3	1482.7
20_4	0.5	0.0	0.0	5.1	70.3	0.0	0.0	0.0	4.2	467.3	50_4	1.6	0.0	0.0	5.6	284.3	1.2	0.0	0.0	5.5	1497.2
20_5	2.3	0.0	0.0	4.8	81.5	1.4	0.0	0.0	4.4	480.6	50_5	3.7	0.0	0.0	4.8	373.7	4.3	0.0	0.0	4.7	1532.1
30_2	1.1	0.0	0.0	4.4	137.9	1.2	0.0	0.0	3.9	885.9	60_2	1.5	0.0	0.0	3.8	350.1	1.4	0.0	0.0	4.0	2016.3
30_3	0.0	0.0	0.0	4.8	120.9	0.1	0.0	0.0	5.1	781.9	60_3	0.9	0.0	0.0	4.5	314.5	1.0	0.0	0.0	4.8	1750.3
30_4	0.3	0.0	0.0	4.6	124.8	1.0	0.0	0.0	5.2	795.7	60_4	1.4	0.0	0.0	5.6	340.4	0.5	0.0	0.0	5.7	1967.8
30_5	3.4	0.0	0.0	5.5	130.8	3.4	0.0	0.0	4.8	797.2	60_5	1.3	0.0	0.0	4.3	343.2	1.3	0.0	0.0	4.8	1777.8

violation at the end of each solving phase compared with the one at the beginning (100 *final violation*/*initial violation*). Furthermore, v_W represents the workload violation defined as the sum of under and over coverage over the total demand ($100 \sum_j \sum_a (u_{ja} + v_{ja})/(\sum_j \sum_a b_{ja})$). Due to the design of the heuristic, violation v_W is always 0 for TS since it starts with a solution satisfying workload, and it uses the `Swap` move. Analogously, violation v_L is always 0 for the LNS since it builds schedules which satisfy all legal constraints. The results show that the proposed hybrid approach is able to find, in reasonable time, a solution satisfying all legal constraints and violating workload demand of 4.5% in average.

References

[1] Fred Glover. Tabu search-part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[2] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, And Computation.* Addison-Wesley, Boston, 3rd edizione edition, 2006.

[3] Claude-Guy Quimper and Louis-Martin Rousseau. A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, 16(3):373–392, April 2009.

[4] Paul Shaw. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. Lecture Notes in Computer Science, pages 417–431. Springer, Berlin, Heidelberg, October 1998.

[5] Jorne Van den Bergh, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385, May 2013.

Parallel machine scheduling with unit time distinct due windows

Oliver Schaudt[1], Stefan Schaudt[2]

[1] RWTH Aachen University, Aachen, Germany
schaudt@mathc.rwth-aachen.de

[2] TU Dortmund, Dortmund, Germany
stefan.schaudt@tu-dortmund.de

Abstract

We consider the problem of non-preemptively scheduling n jobs on m identical parallel machines. Each job has an assigned due window of unit length. The objective is to decide whether a schedule exists in which none of the jobs is either early or tardy.

We present a dynamic program to solve this problem. The running time is bounded by $O(\binom{m+p_{max}}{p_{max}}^2 \cdot (p_{max} + m \cdot n) \cdot T)$, whereas p_{max} denotes the maximal processing time over all jobs and T the time horizon. Our first computational results indicate that the algorithm is significantly faster than solving the respective integer program with a standard solver.

Keywords : *Just-in-time scheduling, distinct due windows.*

1 Introduction

We consider the problem of non-preemptively scheduling n jobs on m identical parallel machines, in which each job has a assigned due window of unit length. The objective is to decide whether a schedule exists such that non of the jobs is either early or tardy. Recall that a job is *early* if its completion time is less than the lower due window bound and *tardy* if it completes after the upper bound. We present an exact algorithm to solve this problem.

A much more general problem is this: Is it possible to build a feasible schedule given n jobs and m identical parallel machines, each job having a release date and a due date? This problem is known to be strongly NP-complete. If the number of distinct release and due dates is bounded by a constant, a pseudo-polynomial algorithm exists. The problems remains NP-complete (in the ordinary sense) even if release and due dates can take only two distinct values.

Another well-studied problem is the scheduling of unit time jobs based on release and due dates for the case of a single machine or identical parallel machines. For the single machine case, Steiner and Yeomans [3] presented a linear time solution algorithm. In the case of m identical parallel machines, Simons and Warmuth [2] created an algorithm that runs in $O(n^2 \cdot m)$ time.

1.1 Motivation

Delivering parcels or food by using autonomous driving robots emerged in recent years and the motivation of our problem stems from this application. In contrast to a common delivery vehicle a robot might have only a capacity of one parcel. A typical robot delivery might look as follows.[1]

Assume that there is a depot with at least one delivery robot and a customer who wants to order. First, the customer places an order and decides to get the goods delivered by a robot.

[1]StarShip Technologies: https://www.starship.xyz

For this, she chooses an arrival window of the robot. Depending on this time window and the location of the customer, a starting time of the robot can be calculated. Shortly before the starting time is reached, the robot will be loaded at the depot. The customer gets a notification when the robot is arriving. After the customer has taken out his order, the robot returns to its depot.

The advantage is the free choice of the delivery window for the customer and the autonomy of the robots. Assume that an operator receives requests from customers overnight. He wants to find out whether there is a feasible schedule through the use of m robots, in which none of the customers are served outside of their time window.

The transformation is not explained here because of the page restriction.

2 Preliminaries and definitions

Suppose we have given a set $\mathcal{J}$ of n jobs and a set of m identical parallel machines. Each job j has an integer processing time, $p(j) \in \{1, \dots, p_{max}\}$, an integer lower due window bound, $d_l(j)$, and an integer upper due window bound, $d_u(j)$. It holds that for each jobs the difference between these bounds is one, $d_u(j) - d_l(j) = 1$. The time horizon is denoted by T, with $T = \max_{j \in \mathcal{J}} d_u(j)$. Furthermore, $\mathcal{J}$ is partitioned into several job sets, denoted by $\mathcal{J}_t$ for $t \in \{0, \dots, T-1\}$, whereas $\mathcal{J}_t$ contains all jobs with a lower due window bound of t. If job j is starts at time t on machine m, it will be finished at time $t + p(j)$. Meanwhile, machine m is blocked for the interval $[t, t + p(j))$. The definition of a feasible schedule is similar to the definition provided by Simons [1]. A feasible schedule is a mapping
$g : \{1, \dots, n\} \to \{0, \dots, T-1\} \times \{1, \dots, m\}$, whereas $g(j) = (s(j), h(j)), 1 \leq j \leq n$, such that:

- $d_l(j) \leq s(j) + p(j) \leq d_u(j)$ (every job has to complete in its due window),
- If $h(j) = h(k)$ with $j \neq k$ and $s(j) \leq s(k)$, then $s(j) + p(j) \leq s(k)$ (each machine can execute at most one job at a time).

We define a partial schedule as a schedule in which not all jobs have been assigned yet. Finally, preemption is not allowed, meaning that a job can not be interrupted.

Tuples

In the following, we consider the due windows separately in ascending order. Assume that we already have a partial schedule up to time t. Furthermore, the machines are ordered in non increasing order according to their maximal completion time $c_{max,i}$. To decide whether it is possible to schedule set $\mathcal{J}_t$, we are interested in the space that remains between time t and the maximum completion time on each machine. This space is represented by a tuple of integers with dimension p_{max}. The first entry corresponds to the number of jobs which completes at time t. Additionally, the i'th entry represents the number of jobs with a completion time between $t - i + 1$ and t. For each due window the number of tuples is bounded by $\binom{m+p_{max}}{p_{max}}$.

Tuple vector

For simplicity, we define a vector $\vec{v}$, which contains all possible tuples for m machines and a maximal processing time of p_{max}. The first tuple of is $(0, \dots, 0)$. The tuple in position i of $\vec{v}$ can be obtained by taking tuple at place *i-1* and increasing the highest entry of the tuple, which is not already m, by one. If this entry is not the last entry, all entries with a larger index are equal to the increased entry. Hence, the last tuple of $\vec{v}$ is $(m, \dots, m)$.

Partial order

Beside vector $\vec{v}$, we can define a partial order to compare two tuples. Given two tuples, x and y, with $x = (x_1, \dots, x_{pmax})$ and $y = (y_1, \dots, y_{pmax})$. It holds that x is preferred to y, short $x \leq y$, if $x_i \leq y_i \; \forall i \in \{1, \dots, p_{max}\}$.

Machine tuples

Given a tuple $(x_0, x_1, \dots, x_n)$ for due window $[t-1, t]$, the corresponding machine tuple is calculated by $(x_1, x_2 - x_1, \dots, x_n - x_{n-1})$. This is denoted by $(\tilde{x_1}, \dots, \tilde{x_n})$, whereas $\tilde{x_i}$ corresponds to the number of machines with a c_{max} of $t - i$.

3 Dynamic program

The dynamic program considers the due windows separately in ascending order, starting with due window $[0, 1]$. In each iteration of the dynamic program it tries to extend the partial schedule by one time unit. The frontier of the partial schedule is represented by a tuple, up to time t. Assuming that tuples up to time t have been obtained, the first tuple is considered. In the subroutine, jobs with a lower due window bound that is smaller than t are assumed to be fix and will not be reassigned or shifted. Based on this tuple and the given set $\mathcal{J}_t$, the program tries to obtain all tuples from vector $\vec{v}$. These tuples represent the new frontier up to time $t + 1$. A detailed description of this procedure is given in subsection 3.1. Whenever it is possible to obtain one of the tuples of vector $\vec{v}$ it will be marked, in case it is not marked yet. This will then become one of the tuples for the next iteration. After all tuples of $\vec{v}$ have been examined, the dynamic program continues by considering the next tuple.

Finally, if all tuples for time t have been considered, the next iteration starts for due window $[t + 1, t + 2]$. In this iteration the program examines all marked tuples for time $t + 1$.

To start the dynamic program at least one marked tuple is needed. This tuple represents the schedule up to time 0 and is denoted by $(m, \dots, m)$.

There are two options for how a run can terminate. Either we have marked a tuple for time T or there exists a point in time, in which none of the tuples is marked. In the first case, we have found a feasible schedule which will be returned. Otherwise, there will be no schedule which contains all jobs.

In the following we explain the subroutine to extend a partial schedule.

3.1 Subroutine

Assume we have given a job set $\mathcal{J}_t$ and two tuples x and y. Tuple x represents the frontier of the partial schedule up to time t and y represents the new frontier, which we are trying to obtain. The machines are sorted according to their c_{max} value in non increasing order. In case the job set equals the empty set, $\mathcal{J}_t = \emptyset$, it is only possible to obtain tuple x, whereas all entries shifted by one to the right and a zero is added in first place, $(0, x_1, \dots, x_{n-1})$.

The extension works as follows, we transform x and y into machine tuples, denoted by $\tilde{x} = (\tilde{x_1}, \dots, \tilde{x_n})$ and respectively $\tilde{y} = (\tilde{y_1}, \dots, \tilde{y_n})$. We start by creating a new machine tuple $\tilde{z}$. The entry $\tilde{z}_i$ represents the number of machines that are valid for jobs with a processing time of at most $i-1$. It will be calculated by comparing x and y, $\tilde{z}_i = \tilde{x}_{i-1} - \tilde{y}_i \forall i \in 3, \dots, n-1$, $\tilde{z}_2 = \tilde{x}_1$, $\tilde{z}_1 = 0$ and $\tilde{z}_n = \tilde{x}_n + \tilde{x}_{n-1} - \tilde{y}_n$. If one of the entries of $\tilde{z}$ is negative, the procedure terminates because based on tuple x it is impossible to obtain the new frontier.

Otherwise, the program tries to place the jobs of set $\mathcal{J}_t$ on the machines represented by tuple $\tilde{z}$. This is done in non increasing order concerning the processing time and decreasing order concerning the index of $\tilde{z}$. In the case a valid machine is found by the program, it tries to place the job in a way that it is completion time equals the lower due window bound. A special case are jobs with a processing time of one. The procedure considers to conditions, first, place the job in a way that it is ending at the left bound of the due window and second, the chosen machine has the highest c_{max} value under all considered machines. If this attempt fails, it inserts the job in a way that it is completing at the right end of its due window bound.

After each placement of a job on a machine, we reduce the corresponding entry of $\tilde{z}$ by one. If the job is placed completing at the left end of its due window, we increase the second entry of $\tilde{z}$ by one. In the case it's impossible to place a job, the procedure terminates because based on tuple x it is impossible to obtain the new frontier.

After we have placed all jobs, we transform the new frontier into a tuple, denoted by s. Finally, we have to do a feasibility check of s. If the first two entries of tuple s are less or equal compared to y, $s_i < y_i$ for $i \in \{1, 2\}$, tuple s will be returned. Otherwise, the procedure terminates. However, it can be that the returned tuple s is not tuple y. However, both tuples are comparable and s will be preferred according to the partial order.

3.2 Running time

The number of iterations of the dynamic program is equal to the time horizon T. Moreover, the number of tuples is bounded by $\binom{m+p_{max}}{p_{max}}$ for each point in time. In each iteration the dynamic program calls the subroutine at most $\binom{m+p_{max}}{p_{max}}^2$ times. This subroutine takes at most p_{max} units for blocking machines and additional $m \cdot n$ units for assigning jobs to machines. In conclusion the resulting running time is $O(\binom{m+p_{max}}{p_{max}}^2 \cdot (p_{max} + m \cdot n) \cdot T)$.

3.3 First computational results

We compared the running time of the dynamic program to the running time of the integer linear program formulation of this problem solved by *Gurobi* 7.0.2 . Both are implemented in Java. Our test sets vary on the number of machines, the number of jobs, the maximal processing time of a job and the time horizon T. The number of jobs is equal to $10, 20, \dots, 100$ and the maximal processing time and the number of machines takes values of $2, 3, \dots, 8$. Based on these 3 input parameters the time horizon was calculated. For each combination of parameters we have calculated a time horizon, in a way that over 10000 runs the number feasible schedules was as close as possible to 5000.

In each test set the processing time, p_j, of each job is uniformly and discretely distributed between 1 and p_{max}. Furthermore, the lower due window bound d_l, is uniformly and discretely distributed between 0 and $T-1$. In the case that $d_l - p_j$ is smaller than one, we set $p_j = d_l$.

With this generation procedure we generated in total 490 test sets. On average, the dynamic program is 48 times faster than the integer linear problem formulation of this problem.

4 Outlook

Our approach can be used to deal with non-overlapping due windows of arbitrary size. This model covers applications where the customer chooses a one hour slot for the delivery of her parcel while the scheduler is able to plan with 15 minute slot resolution.

References

[1] B. Simons. *Multiprocessor Scheduling of Unit-Time Jobs with Arbitrary Release Times and Deadlines.* SIAM Journal on Computing, 12(2):294-299, 1983.

[2] B. Simons and M. K. Warmuth. *A fast algorithm for multiprocessor scheduling of unit-length jobs.* SIAM Journal on Computing, 18(4):690-710, 1989.

[3] G. Steiner and S. Yeomans. *A note on "Scheduling unit-time tasks with integer release times and deadlines".* Information Processing Letters, 47(3):165-166, 1993.

Single machine scheduling with bounded job rearrangements

Arianna Alfieri[1], Gaia Nicosia[2], Andrea Pacifici[3], Ulrich Pferschy[4]

[1] Politecnico di Torino
arianna.alfieri@polito.it
[2] Università degli studi "Roma Tre"
nicosia@ing.uniroma3.it
[3] Università degli studi di Roma "Tor Vergata"
andrea.pacifici@uniroma2.it
[4] University of Graz
ulrich.pferschy@uni-graz.at

Abstract

In operations scheduling, changes in the scenario predicted beforehand (due to, e.g., disruptions, breakdowns, realizations different from expected data values) make it often necessary to revise a plan in order to meet the original objectives under the new circumstances and with the changed data. In this case, cost and/or stability issues require the consideration of solutions which do not differ too much, i.e., that can be obtained by a small number of modifications, from the original ones. In this paper we focus on a single-machine scheduling setting where we need to "adjust" a given, predefined solution, by re-sequencing jobs, but there are bounds on the number and type of allowed job shifts. We propose mathematical programming models and possible solution approaches.

Keywords : *single-machine scheduling, integer programming, dynamic programming*

1 Introduction

In several application settings, scheduling decisions are often taken in multiple successive phases (e.g., corresponding to different stages of a production facility) enabling real-time "corrections" of the solution computed in the previous phase. For instance, rearrangement of a solution schedule may be needed in order to account for possible different criteria at different stages of a supply chain. This is quite a common scenario, in which the decision makers at various levels of the supply chain have their own objectives and constraints that would lead to different schedules (see e.g., [1]). As a consequence, as the orders proceed from one stage to the next—since it is unlikely that the same schedule fits all the stages of the supply chain—re-sequencing operations may be needed at intermediate buffers to consider new requirements.

Rearrangements could be also necessary in other settings: Consider a system with a single objective function and subject to events that lead to a perturbation of the originally given or estimated data, e.g., changes in the expected job durations or availabilities as in [2]. In this case, the given, previously optimal, solution sequence is not performing so well anymore and re-scheduling must be carried out.

In general, modifications of an already devised sequence (i.e., re-scheduling due to new requirements or to disruptions or data changes) could be very expensive both in terms of costs and operational complications associated to unstable solutions, as it happens in the well known nervousness phenomenon [6]. As a consequence, when changing a solution schedule, cost and stability issues become of primary importance. On these grounds, we aim at balancing the cost for re-positioning the jobs and the improvement in the objective. In the following, we specifically refer to the latter problem as *scheduling with rearrangements*.

2 Problem definition

Consider a deterministic single-machine environment where n jobs with given processing times have to be scheduled according to a regular objective function $C(\sigma)$ (e.g., minimization of the total completion time, or minimization of the maximum tardiness, etc.), which depends on the job sequence σ.

Without loss of generality, it is assumed that an initial sequence $\sigma_0 = \langle 1, 2, \ldots, n \rangle$, of the n jobs (representing an optimal or satisfactory solution for the original data) is given and that job j is the job placed in the j-th position of σ_0. Suppose now that, due to changed conditions (e.g., altered processing times, different objective at the current supply chain stage, etc.), σ_0 is no more adequate in terms of the ongoing performance indicator $C(\cdot)$. In order to get back to a satisfactory solution, it is possible to rearrange jobs, so that a new sequence $\sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle$ is obtained which achieves a better performance. The problem we address in this work is to determine a new job sequence σ such that:

(*i*) $C(\sigma)$ is minimum (or, in any case, $C(\sigma) \leq C(\sigma_0)$) and

(*ii*) the value of a "distance function" between the new sequence σ and the initial sequence σ_0 is bounded.

Several different metrics are used in the literature to define distances between *rankings*. Hereafter, the ranking of a job denotes its position in the sequence (e.g., the ranking of job j in σ_0 is j) and hence a permutation or sequence is completely described by the rankings of its jobs. The classical metrics are Kendall's tau [3] and Spearman's footrule [5]. More recently, several other metrics have been proposed; in [4] a number of them are surveyed and new ones are proposed.

The metric we adopt in this paper is the number of *moves* necessary to reach σ starting from σ_0, where a move is the removal of a job from the current sequence and its insertion in a *successive* position. By "successive position" we mean a position with larger index number, i.e. further "to the right" of the sequence. This restricted notion of moves is motivated by the real-world example of a machine which receives its input by a conveyor belt. Since the conveyor keeps moving forward, it is not possible to insert a removed jobs in an earlier position in the sequence, i.e., in front of the queue, see Figure 1 for an illustration. In this figure, the new sequence σ is obtained in three moves. As the conveyor belt proceeds to the left, we can remove jobs 1 and 2 to place them after 4 and 6, respectively. Doing so, 3 becomes the first job of the new sequence. Note that, in this setting, we would not be allowed to move jobs 3 and 4 before job 1.

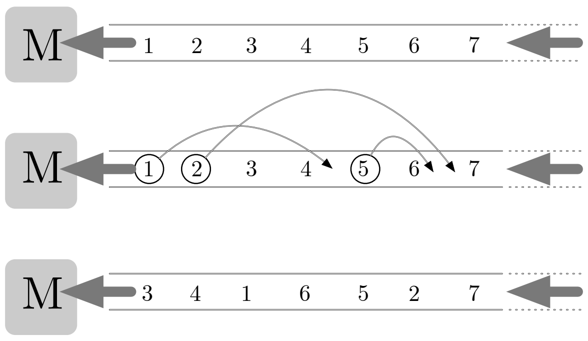


FIG. 1: As the conveyor belt proceeds leftward, three moves are performed.

Observe that even under our restricted definition of moves, every permutation of the jobs can be obtained by a suitable sequence of feasible moves. Note also that in our metric we do not care about the distance (i.e., number of positions) a job is moved, but we focus on the number of moving operations. This is motivated by the practical situation where a job is associated with an item (work piece, part of a machinery) and moving such an item entails, for instance, a forklift to pick up the item and drive it to the new position. In this case, driving distance is mostly negligible, while the pickup and placing operations are costly.

The following Lemma helps in determining the number of moves necessary to reach a given sequence σ, starting from the initial sequence, say σ_0. (Recall that, in σ_0, the name of a job indicates its position/ranking in the sequence.)

Lemma 1 *Transforming a starting sequence σ_0 into a new, given sequence σ by moves to successive positions, a job j destined for a position i in σ must be moved from its original position j in σ_0, if and only if, there is a job $k > j$ placed before the i-th position in σ.*

For instance, in Figure 1, $\sigma = \langle 3, 4, 1, 6, 5, 2, 7\rangle$, therefore 1, 5, and 2, have been re-positioned, since they are preceded by higher ranking jobs, while 3, 4, 6, and 7 were not.

Let $\mathfrak{S}_n$ indicate the set of all sequences (i.e. permutations) of the n jobs. We are now in the position to give a formal definition of our problem.

> Single-machine Scheduling with Rearrangements (SSRP)
> *Given:* a set $J = \{1, 2, \dots, n\}$ of jobs with nonnegative processing times p_i, $i = 1, \dots, n$, an initial sequence of jobs $\sigma_0 = \langle 1, 2, \dots, n\rangle$, a regular objective function $C : \mathfrak{S}_n \longrightarrow \mathbb{R}_+$ depending on the job sequence, and an integer k;
> *Find:* a sequence $\sigma \in \mathfrak{S}_n$ that can be reached in at most k moves from σ_0 such that $C(\sigma)$ is minimum.

The job in position i in the sequence $\sigma \in \mathfrak{S}_n$ is denoted by $\gamma_\sigma(i) \in \{1, 2, \dots, n\}$ (e.g.., $\gamma_{\sigma_0}(i) = i$), while $\pi_\sigma(j)$ denotes the position of job j. If the context makes it clear which sequence we are talking about, we omit the subscript σ and write, e.g., $\pi(\gamma(i)) = i$ or $\gamma(\pi(j)) = j$.

3 Solution approaches and preliminary results

We propose integer linear programming models for three cases of our problem, corresponding to three different objective functions. In addition, we present efficient procedures to solve restricted versions of SSRP. These can be of use, as building blocks, in devising heuristic solution algorithms for the general case.

3.1 Mathematical programming model

In our study we consider the minimization of the following objective functions $C(\sigma)$. Hereafter we only sketch the idea behind the corresponding MIP models for our Single-machine Scheduling with Rearrangements problem and omit the obvious details of the complete formulations.

In all mathematical programs we use the assignment variables $x \in \{0, 1\}^{n\times n}$, where $x_j(i) = 1$ indicates that job j is placed in position i in σ. The ILP models consider the following criteria:

- *Total completion time.* In this case the objective is

$$C(\sigma) = \sum_{i=1}^{n} \sum_{h=1}^{i} \sum_{j=1}^{n} x_j(h) p_j. \tag{1}$$

- *Maximum lateness.* Here $C(\sigma) = L_{\max}$ represents the maximum lateness:

$$C(\sigma) = L_{\max} \geq \sum_{h=1}^{i} \sum_{j=1}^{n} p_j x_j(h) - \sum_{j=1}^{n} d_j x_j(i) \qquad \forall\, i = 1, \dots, n. \tag{2}$$

- *Number of tardy jobs*

$$C(\sigma) = \sum_{i=1}^{n} U_i \tag{3}$$

  Here binary variables $U_i \geq \frac{1}{M}\left(\sum_{h=1}^{i} \sum_{j=1}^{n} p_j x_j(h) - \sum_{j=1}^{n} d_j x_j(i)\right)$, $i = 1, 2, \dots, n$ indicate that job *in position* i is tardy when $U_i = 1$.

We observe that in order to model the number of moves, i.e. to "count" the number of jobs that are to be repositioned, recalling Lemma 1, we may express the quantities $\gamma(i)$ and $\pi(j)$ as functions of the x variables.

We performed a number of preliminary computational experiments on randomly generated instance using the commercial ILP solver Gurobi. These tests show that Gurobi is able to solve quite efficiently instances with up to 40 jobs and $k = 3$. As soon as n or k grows, the time required to solve the problem becomes too large. So, we also implemented a few simple greedy rules to detect the k jobs to be repositioned and on which positions to schedule them. Such greedy rules are extremely fast (a few milliseconds for quite large instances), but poorly effective when the number of jobs is small. Their performance improves when the number of jobs becomes larger.

A few experiments were also performed to test how the value of k influences the objective function values. In most cases, it turns out that it is possible to find the optimal solution as soon as k reaches $\approx \frac{n}{3}$.

3.2 Restricted problems

We consider a restricted variant of SSRP in which the set of k jobs that we are allowed to move is given as an input. For this restriction we are able to provide efficient solution algorithms. More precisely, our contributions are:

- a dynamic programming algorithm for the problem in which we want to minimize the number of late jobs;
- a modification to Lawler's algorithm for solving the problem when the objective is the minimization of the maximum lateness.

References

[1] A. Agnetis, N.G. Hall, and D. Pacciarelli. *Supply chain scheduling: Sequence coordination.* Discrete Applied Mathematics, 154(15):2044–2063, 2006.

[2] N.G. Hall and C.N. Potts. *Rescheduling for Job Unavailability.* Operations Research, 58 (3):746-755, 2010.

[3] M.G. Kendall. *A New Measure of Rank Correlation.* Biometrika, 30(1–2):81–93, 1938.

[4] R. Kumar and S. Vassilvitskii. *Generalized Distances Between Rankings.* Proceedings of the 19th International Conference on World Wide Web:571–580, 2010.

[5] C. Spearman. *The Proof and Measurement of Association between Two Things.* The American Journal of Psychology, 15(1):72–101, 1904.

[6] X. Wang and S.M. Disney. *The bullwhip effect: Progress, trends and directions.* European Journal of Operational Research, 250(3):691–701, 2016.

A Generalized Turán Problem and its Applications

Lior Gishboliner[1], Asaf Shapira[2]

[1] School of Mathematics, Tel Aviv University, Tel Aviv 69978, Israel.
liorgis1@post.tau.ac.il

[2] School of Mathematics, Tel Aviv University, Tel Aviv 69978, Israel
asafico@tau.ac.il [*]

Abstract

The investigation of conditions guaranteeing the appearance of cycles of certain lengths is one of the most well-studied topics in graph theory. In this paper we consider a problem of this type which asks, for fixed integers ℓ and k, how many copies of the k-cycle guarantee the appearance of an ℓ-cycle? Extending previous results of Bollobás–Győri–Li and Alon–Shikhelman, we fully resolve this problem by giving tight (or nearly tight) bounds for all values of ℓ and k.

We also present a somewhat surprising application of the above mentioned estimates to the study of graph property testing. Prior to this work, all bounds for the query complexity of testing graph properties were either polynomial or there was a tower-type gap between the best known upper and lower bounds. We fill this gap by showing that for every super-polynomial function $f(\varepsilon)$, there is a monotone graph property $\mathcal{P}$, such that the query complexity of the optimal one-sided-error ε-tester for $\mathcal{P}$ is precisely given by $f(\varepsilon)$. We thus obtain the first examples of tight super-polynomial bounds for the one-sided-error query complexity of graph properties. A special case of this result resolves a problem of Alon and the second author, while another special case partially resolves a problem of Goldreich.

1 The Generalized Turán Problem for Cycles

Turán's Theorem, one of the cornerstone results in graph theory, determines the maximum number of edges in an n-vertex graph that does not contain a K_t (the complete graph on t vertices). Turán's problem is the following more general question: for a fixed graph H and an integer n, what is the maximum number of edges in an n-vertex H-free graph? This quantity is denoted by $\mathrm{ex}(n, H)$. Estimating $\mathrm{ex}(n, H)$ for various graphs H is one of the most well-studied problems in graph theory.

Alon and Shikhelman [4] have recently initiated the systematic study of the following natural generalization of $\mathrm{ex}(n, H)$; for fixed graphs H and T, estimate $\mathrm{ex}(n, T, H)$, which is the maximum number of copies[1] of T in an n-vertex graph that contains no copy of H. Note that $\mathrm{ex}(n, H) = \mathrm{ex}(n, K_2, H)$. For the sake of brevity we refer the reader to [4] for more background and motivation, as well as examples of some well-studied problems in extremal combinatorics which can be cast in the setting of studying $\mathrm{ex}(n, T, H)$ for various pairs H and T.

Let C_k denote the cycle of length k. The problem of estimating $\mathrm{ex}(n, C_k, C_\ell)$ has recently received a lot of attention. Bollobás and Győri [5] proved that $\mathrm{ex}(n, C_3, C_5) = \Theta(n^{3/2})$. Győri and Li [11] extended this result by showing that $\Omega(\mathrm{ex}(n, \{C_4, C_6, \ldots, C_{2\ell}\})) \leq \mathrm{ex}(n, C_3, C_{2\ell+1}) \leq O_\ell(\mathrm{ex}(n, C_{2\ell}))$, where $\mathrm{ex}(n, \{C_4, C_6, \ldots, C_{2\ell}\})$ is the maximal number of edges in an n-vertex graph with no copy of C_{2t} for any $2 \leq t \leq \ell$. This result was also obtained by Füredi and

[*]Supported in part by ISF Grant 1028/16 and ERC Starting Grant 633509.

[1]When counting copies of T in G we always mean unlabeled copies.

Özkahya [7], who in addition proved similar bounds for $\mathrm{ex}(n, C_3, C_{2\ell})$. Our main result, stated as Theorem 1, significantly extends the above results by giving asymptotically tight bounds for $\mathrm{ex}(n, C_k, C_\ell)$ for all fixed k, ℓ.

Theorem 1 *For distinct k, ℓ we have*

$$\mathrm{ex}(n, C_k, C_\ell) = \begin{cases} \Theta_k(n^{k/2}) & k \geq 5, \ell = 4, \\ \Theta_k(\ell^{\lceil k/2 \rceil} n^{\lfloor k/2 \rfloor}) & \ell \geq 6 \text{ even, } k \geq 4, \\ \Theta_k(\ell^{\lceil k/2 \rceil} n^{\lfloor k/2 \rfloor}) & k, \ell \text{ odd, } 5 \leq k < \ell. \end{cases}$$

The implied constants in the theorem depend on k. Note that Theorem 1 covers all cases except for those already resolved in [11, 7], and the trivial cases where either k is even and ℓ is odd, or k and ℓ are both odd and $k > \ell$. In these cases a blow-up of C_k is C_ℓ-free, and hence $\mathrm{ex}(n, C_k, C_\ell) = \Theta(n^k)$.

2 Applications to Graph Property Testing

We would now like to describe somewhat surprising applications of Theorem 1 related to the area of graph property testing. Let $\mathcal{P}$ be a monotone graph property, that is, a property closed under removal of vertices and edges. We say that a graph G is ε-far from $\mathcal{P}$ if one must remove at least εn^2 edges to make G satisfy $\mathcal{P}$. In the setting we will consider here, which was first introduced in [10], one assumes that a (randomized) algorithm can sample a set of vertices S from an input graph G, and based on the graph induced by this set should be able to distinguish with probability at least 2/3 between the case that G satisfies a property $\mathcal{P}$ and the case that G is ε-far from satisfying $\mathcal{P}$. Such an algorithm is called an ε-tester for $\mathcal{P}$. We denote by $q = q_{\mathcal{P}}(\varepsilon, n)$ the smallest integer for which there is an ε-tester for $\mathcal{P}$ that works by randomly selecting a set of vertices S of size q from n-vertex graphs. The surprising fact is that for many natural properties $\mathcal{P}$ there is a function $q_{\mathcal{P}}(\varepsilon)$ satisfying $q_{\mathcal{P}}(\varepsilon, n) \leq q_{\mathcal{P}}(\varepsilon)$ for every n, that is, there is an ε-tester for $\mathcal{P}$ which inspects an induced subgraph of size that depends only on ε and not on $|V(G)|$.

Alon and the second author [3] proved that every monotone property $\mathcal{P}$ is testable with a number of queries that depends only on ε. In fact, the tester is extremely simple: it accepts if and only if the sample satisfies $\mathcal{P}$. So what the authors of [3] actually proved is that for every $\varepsilon > 0$ there is a (least) integer $w_{\mathcal{P}}(\varepsilon)$ such that if a graph G is ε-far from $\mathcal{P}$ then a random subset $S \subseteq V(G)$ of $w_{\mathcal{P}}(\varepsilon)$ vertices is such that $G[S]$ (the subgraph of G induced by S) does not satisfy $\mathcal{P}$ with probability at least 2/3. This result generalizes the famous graph removal lemma [12], that can be stated as saying that if a graph G is ε-far from being H-free then a randomly chosen set of vertices S of size $w_H(\varepsilon)$ is such that $G[S]$ contains a copy of H with probability at least 2/3.

Note that while testers might have 2-sided error[2], the above tester for a monotone property has 1-sided error, that is, it accepts graphs satisfying $\mathcal{P}$ with probability 1 (and rejects those that are ε-far from $\mathcal{P}$ with probability at least 2/3). It is easy to see that a 1-sided tester of a monotone property $\mathcal{P}$ cannot reject an input if $G[S]$ satisfies $\mathcal{P}$. Hence $w_{\mathcal{P}}(\varepsilon)$ actually equals the query complexity of the optimal 1-sided tester for $\mathcal{P}$.

The aforementioned result of [3] is proved using Szemerédi's regularity lemma, and hence the bounds it supplies for $w_{\mathcal{P}}(\varepsilon)$ are (at least) of tower-type. Unfortunately, there are properties for which no (significantly) better bounds are known. The prime example is triangle freeness, for which, despite much effort, the best known bounds are $(1/\varepsilon)^{c \log 1/\varepsilon} \leq w_{\mathcal{P}}(\varepsilon) \leq \mathrm{tower}(O(\log 1/\varepsilon))$, where[3] the lower bound was obtained in [12], and the upper bound was obtained only recently by Fox [6].

[2]That is, the tester is allowed to err in both direction, i.e. reject graphs satisfying $\mathcal{P}$ with small probability as well as accept graphs that are ε-far from $\mathcal{P}$ with small probability.

[3]$\mathrm{tower}(x)$ is the tower function, that is, a tower of exponents of height x.

Another notable result is that of [10], who showed that k-colorability is testable with $\mathrm{poly}(1/\varepsilon)$ queries, thus improving upon an earlier tower-type bound of Rödl and Duke.

Prior to this work, all known bounds for the query complexity of monotone properties were of the above two types, that is, they were either of the form $w_{\mathcal{P}}(\varepsilon) = \mathrm{poly}(1/\varepsilon)$ or there was (at least) a tower-type gap between the best known lower and upper bounds. In light of this situation, Goldreich [8] asked to exhibit a property for which the optimal query complexity is $2^{1/\varepsilon}$. Here we apply Theorem 1 to partially answer this question, by exhibiting a property for which the optimal *1-sided* error query complexity is $2^{\Theta(1/\varepsilon)}$. In fact, we establish the following general result:

Theorem 2 *There is an absolute constant c such that for every decreasing function $f\colon (0,1) \to \mathbb{N}$ satisfying $f(x) \geq 1/x$, there exists a monotone graph property $\mathcal{P}$ satisfying $f(\varepsilon) \leq w_{\mathcal{P}}(\varepsilon) \leq \varepsilon^{-14} f(\varepsilon/c)$.*

Theorem 2 can be considered a *hierarchy theorem* for query complexity of 1-sided ε-testers, somewhat reminiscent of the famous time/space hierarchy theorems in computational complexity theory.

We now turn to the second application of Theorem 1. In what follows, we use $w_k(\varepsilon)$ instead of $w_{\mathcal{P}}(\varepsilon)$, where $\mathcal{P}$ is the k-colorability property. Recall that the general bounds for $w_{\mathcal{P}}(\varepsilon)$ obtained by the aforementioned result of [3] are of tower-type (or even worse). A natural problem, first raised already in [3] and later also by Goldreich [9] and Alon and Fox [2] is to characterize the properties for which $w_{\mathcal{P}}(\varepsilon) = \mathrm{poly}(1/\varepsilon)$. Since this problem seemed (and still seems) to be out of reach, [3] asked if one can at least solve the following (very) special case. Given a set of integers L, let us say that a graph is L-free if it is C_ℓ-free for every $\ell \in L$. The problem of [3] then asks to characterize the sets L for which[4] $w_L(\varepsilon) = \mathrm{poly}(1/\varepsilon)$. The result of [10] stating that $w_2(\varepsilon) = \mathrm{poly}(1/\varepsilon)$ is then equivalent to the statement that if L consists of *all* odd integers then $w_L(\varepsilon) = \mathrm{poly}(1/\varepsilon)$. Another related result is due to Alon [1] who proved that $w_L(\varepsilon) = \mathrm{poly}(1/\varepsilon)$ if L contains at least one even integer, and that $w_L(\varepsilon)$ is super-polynomial whenever L is a *finite* set of odd integers. Our second application of Theorem 1 solves all the cases not handled by previous results.

Theorem 3 *Let $L = \{\ell_1, \ell_2, \ldots\}$ be an infinite increasing sequence of odd integers. Then we have $w_L(\varepsilon) = \mathrm{poly}(1/\varepsilon)$ if and only if $\limsup_{j \longrightarrow \infty} \frac{\log \ell_{j+1}}{\log \ell_j} < \infty$.*

By the above theorem, as long as ℓ_j does not grow faster than 2^{2^j}, one can get a polynomial bound on $w_L(\varepsilon)$, while for any (significantly) faster-growing ℓ_j one cannot get such a bound.

We now turn to the last application of Theorem 1. It is natural to ask whether $q_{\mathcal{P}}(\varepsilon)$ can be significantly smaller that $w_{\mathcal{P}}(\varepsilon)$, that is, if 2-sided testers have any advantage over 1-sided ones. The simple answer is of course yes; for example, if $\mathcal{P}$ is the property of having edge density at least 1/4 (i.e. having at least $n^2/4$ edges) then it is easy to see that $q_{\mathcal{P}}(\varepsilon) \leq \mathrm{poly}(1/\varepsilon)$ as one can just estimate the edge density of the input. On the other hand, it is also easy to see that $\mathcal{P}$ is not testable with 1-sided error using a number of queries that is independent of n. It is thus more natural to restrict ourselves to graph properties that *can* be tested with 1-sided error, and ask: to what extent are 2-sided testers more powerful than 1-sided testers for monotone properties?

It is (perhaps) natural to guess that at least for monotone properties $\mathcal{P}$, 2-sided testers do not have any advantage over 1-sided testers, the explanation being that the only way one can be convinced that a graph is far from satisfying a monotone property $\mathcal{P}$ is by finding a witness to this fact in the form of a subgraph not satisfying $\mathcal{P}$. As Theorem 4 below shows, this intuition turns out to be false in a very strong sense. This theorem implies that 2-sided testers can be *arbitrarily* more efficient than 1-sided testers, even for monotone graph properties. Prior to this work, it was not even known that 2-sided testers can be super-polynomially stronger than 1-sided testers.

[4]We slightly abuse the notation here by using L in the notation w_L also for the *property* of being L-free.

Theorem 4 *For every decreasing function $f\colon (0,1) \to \mathbb{N}$ satisfying $f(x) \geq 1/x$, there is a monotone graph property $\mathcal{P}$ so that*

- *$\mathcal{P}$ has 1-sided error query complexity $w_{\mathcal{P}}(\varepsilon) \geq f(\varepsilon)$.*
- *$\mathcal{P}$ has 2-sided error query complexity $q_{\mathcal{P}}(n, \varepsilon) = \mathrm{poly}(1/\varepsilon)$ for every $n \geq n_0(\varepsilon)$.*

References

[1] N. Alon, Testing subgraphs in large graphs, Random Structures and Algorithms 21 (2002), 359-370.

[2] N. Alon and J. Fox, Easily testable graph properties, Combin. Probab. Comput. 24 (2015), 646-657.

[3] N. Alon and A. Shapira, Every monotone graph property is testable, SIAM Journal on Computing 38 (2008), 505-522.

[4] N. Alon and C. Shikhelman, Many T copies in H-free graphs, Journal of Combinatorial Theory, Series B, 121 (2016), 146-172.

[5] B. Bollobás and E. Győri, Pentagons vs. triangles, Discrete Math. 308 (2008), 4332-4336.

[6] J. Fox, A new proof of the graph removal lemma, Ann. of Math. 174 (2011), 561-579.

[7] Z. Füredi and L. Özkahya, On 3-uniform hypergraphs without a cycle of a given length. Discrete Applied Mathematics, 216, pp.582–588, 2017.

[8] O. Goldreich, Contemplations on testing graph properties, ECCC Technical Report 2005. Also, Studies in Complexity and Cryptography 6650 (2011), 547-554.

[9] O. Goldreich, **Introduction to Property Testing**, Forthcoming book, 2017.

[10] O. Goldreich, S. Goldwasser and D. Ron, Property testing and its connection to learning and approximation, J. ACM 45 (1998), 653-750.

[11] E. Győri and H. Li, The maximum number of triangles in C_{2k+1}-free graphs, Combinatorics, Probability and Computing 21 (2012), 187-191.

[12] I.Z. Ruzsa and E. Szemerédi, Triple systems with no six points carrying three triangles, in Combinatorics (Keszthely, 1976), Coll. Math. Soc. J. Bolyai 18, Volume II, 939–945.

On the spectra of general random mixed graphs

Dan Hu[1,3], Hajo Broersma[1], Jiangyou Hou[2], Shenggui Zhang[3]

[1] Faculty of EEMCS, University of Twente, Enschede, The Netherlands
hudan@mail.nwpu.edu.cn, h.j.broersma@utwente.nl

[2] School of Mathematics, Northwest University, Xi'an, China
jiangyonghou@nwu.edu.cn

[3] Department of Applied Mathematics, Northwestern Polytechnical University, Xi'an, China
sgzhang@nwpu.edu.cn

Abstract

A mixed graph is a graph that can be obtained from a simple undirected graph by replacing some of the edges by arcs in precisely one of the two possible directions. The Hermitian adjacency matrix of a mixed graph G of order n is the $n \times n$ matrix $H(G) = (h_{ij})$, where $h_{ij} = -h_{ji} = \mathrm{i}$ (with $\mathrm{i} = \sqrt{-1}$) if there exists an arc from v_i to v_j (but no arc from v_j to v_i), $h_{ij} = h_{ji} = 1$ if there exists an edge (and no arcs) between v_i and v_j, and $h_{ij} = 0$ otherwise (if v_i and v_j are neither joined by an edge nor by an arc). We study the spectra of the Hermitian adjacency matrix and the normalized Hermitian Laplacian matrix of general random mixed graphs, i.e., in which all arcs are chosen independently with different probabilities (and an edge is regarded as two oppositely oriented arcs joining the same pair of vertices). For our first main result, we derive a new probability inequality and apply it to obtain an upper bound on the eigenvalues of the Hermitian adjacency matrix. Our second main result shows that the eigenvalues of the normalized Hermitian Laplacian matrix can be approximated by the eigenvalues of a closely related weighted expectation matrix, with error bounds depending on the minimum expected degree of the underlying undirected graph.

Keywords : *general random mixed graphs; random Hermitian adjacency matrix; random normalized Hermitian Laplacian matrix; spectra.*

1 Introduction

Spectra of the adjacency matrix and the normalized Laplacian matrix of graphs have many applications in graph theory. For example, the spectrum of the adjacency matrix of a graph is related to its connectivity and the number of occurrences of specific subgraphs, but also to its chromatic number and its independence number. The spectrum of the normalized Laplacian matrix is related to diffusion on graphs, random walks on graphs and the Cheeger constant. For more details on these notions, and for more applications of spectra of the adjacency matrix and the normalized Laplacian matrix, we refer the interested reader to two monographs [1, 7].

Also for random graphs, spectra of their adjacency matrix and their normalized Laplacian matrix are well-studied (See, e.g., [2, 3, 5, 6, 8, 11, 12]). Füredi and Komlós [12] studied the asymptotic behaviour of the spectra of Erdős-Rényi random graphs, giving the separation of first and remaining eigenvalues of the adjacency matrices of random graphs. These results were extended to sparse random graphs [8, 14]. Friedman et al. [9, 10, 11] studied the second largest eigenvalue (in absolute value) of random d-regular graphs. Chung, Lu, and Vu [3] studied spectra of the adjacency matrix of random power law graphs, and spectra of the normalized Laplacian matrix of random graphs with given expected degrees. Their results on random graphs with given expected degree sequences were supplemented by Coja-Oghlan et al. [5, 6] for sparse random graphs. Lu and Peng [16, 17] studied spectra of the adjacency matrix and the normalized Laplacian matrix of edge-independent random graphs, as well as spectra of the normalized Laplacian matrix of random hypergraphs. Oliveira [18] considered the problem of approximating the spectra of the adjacency matrix and the normalized Laplacian matrix of random graphs. His results were improved by Chung and Radcliffe [4].

In this talk we focus on the spectra of the Hermitian adjacency matrices and the normalized Hermitian Laplacian matrices of general random mixed graphs.

2 Preliminaries

A graph is called a *mixed graph* if it contains both directed and undirected edges. We use $G = (V(G), E(G), A(G))$ to denote a mixed graph with a set $V(G)$ of vertices, a set $E(G)$ of (undirected) edges, and a set $A(G)$ of arcs (directed edges). We define the *underlying graph* of G, denoted by $\Gamma(G)$, as the graph with vertex set $V(\Gamma(G)) = V(G)$, and edge set

$$E(\Gamma(G)) = \{v_iv_j \mid v_iv_j \in E(G) \text{ or } (v_i, v_j) \in A(G) \text{ or } (v_j, v_i) \in A(G)\}.$$

We adopt the terminology and notation of Liu and Li in [15], and define the *Hermitian adjacency matrix* of a mixed graph G of order n to be the $n \times n$ matrix $H(G) = (h_{ij})_{n\times n}$, where

$$h_{ij} = \begin{cases} 1, & \text{if } v_iv_j \in E(G); \\ \mathrm{i}, & \text{if } (v_i, v_j) \in A(G) \text{ and } (v_j, v_i) \notin A(G); \\ -\mathrm{i}, & \text{if } (v_i, v_j) \notin A(G) \text{ and } (v_j, v_i) \in A(G); \\ 0, & \text{otherwise.} \end{cases}$$

Here, $\mathrm{i} = \sqrt{-1}$. This matrix, that is indeed Hermitian, as one easily sees, was also introduced independently by Guo and Mohar in [13]. We denote by $\lambda_i(H(G))$ the i-th largest eigenvalue of $H(G)$ (multiplicities counted). We use $\{\lambda_1(H(G)), \lambda_2(H(G)), \dots, \lambda_n(H(G))\}$ to denote the *spectrum* of $H(G)$ in nonincreasing order. The set of these eigenvalues is called the *Hermitian adjacency spectrum* (or briefly *H-spectrum*) of G. Let $V(G) = \{v_1, v_2, \dots, v_n\}$, and let $D(G) = \mathrm{diag}(d_1, d_2, \dots, d_n)$ be a diagonal matrix, in which d_i is the degree of the vertex v_i in the underlying graph $\Gamma(G)$. Then the matrix $L(G) = D(G) - H(G)$ is called the *Hermitian Laplacian matrix* of G, and the matrix $\mathcal{L}(G) = I - D(G)^{-\frac{1}{2}}H(G)D(G)^{-\frac{1}{2}}$ is called the *normalized Hermitian Laplacian matrix* of G. Here I is the $n \times n$ identity matrix. We denote by $\lambda_i(\mathcal{L}(G))$ the i-th largest eigenvalue of $\mathcal{L}(G)$ (multiplicities counted). We use $\{\lambda_1(\mathcal{L}(G)), \lambda_2(\mathcal{L}(G)), \dots, \lambda_n(\mathcal{L}(G))\}$ to denote the spectrum of $\mathcal{L}(G)$ in nonincreasing order. The set of these eigenvalues is called the *normalized Hermitian Laplacian spectrum* of G.

If we regard (replace) each edge $v_iv_j \in E(G)$ in $G = (V(G), E(G), A(G))$ as (by) two oppositely directed arcs (v_i, v_j) and (v_j, v_i), then G is a directed graph. Throughout this talk, we regard mixed graphs as directed graphs, in the above sense. Next, we give the definition of a *general random mixed graph* $\widehat{G}_n(p_{ij})$. Let K_n be a complete graph on n vertices. The *complete directed graph* DK_n is the graph obtained from K_n by replacing each edge of K_n by two oppositely directed arcs. Let p_{ij} be a function of n such that $0 < p_{ij} < 1$ $(i \neq j)$. We always assume that $p_{ii} = 0$ for all indices i. The random mixed graph model $\widehat{\mathcal{G}}_n(p_{ij})$ consists of all random mixed graphs $\widehat{G}_n(p_{ij})$ in which each arc (v_i, v_j) with $i \neq j$ is chosen randomly and independently, with probability p_{ij} from the set of arcs of DK_n, where we let the vertex set be $\{v_1, v_2, \dots, v_n\}$. Here the probabilities p_{ij} for different arcs are not assumed to be equal, that is, $\widehat{G}_n(p_{ij})$ is an arc-independent random mixed graph of order n. Then the *Hermitian adjacency matrix* of $\widehat{G}_n(p_{ij})$, denoted by $H(\widehat{G}_n(p_{ij})) = (h_{ij})$ (or H_n, for brevity), satisfies that:

- H_n is a random Hermitian matrix, with $h_{ii} = 0$ for $1 \leq i \leq n$;
- the upper-triangular entries h_{ij}, $1 \leq i < j \leq n$ are independent random variables, which take value 1 with probability $p_{ij}p_{ji}$, i with probability $p_{ij}(1 - p_{ji})$, $-\mathrm{i}$ with probability $(1 - p_{ij})p_{ji}$, and 0 with probability $(1 - p_{ij})(1 - p_{ji})$.

If A is a random $n \times n$ matrix, we write $\mathbb{E}(A)$ to denote the coordinate-wise expectation of A, so $\mathbb{E}(A)_{ij} = \mathbb{E}(A_{ij})$. Similarly, If A is a random $n \times n$ Hermitian matrix, then $\mathrm{Var}(A) = \mathbb{E}((A - \mathbb{E}(A))^2)$. We use $\mathbb{E}H_n$ as shorthand for $\mathbb{E}(H_n)$, and note that it is obvious that $(\mathbb{E}H_n)_{ij} = \mathbb{E}(h_{ij}) = p_{ij}p_{ji} + \mathrm{i}(p_{ij} - p_{ji})$.

In [4], Chung and Radcliffe give a short proof for the following expression that applies to random graphs.

Theorem 1 ([4]) *Let $X_1, X_2, \ldots, X_m$ be independent random $n \times n$ Hermitian matrices. Moreover, assume that $\|X_i - \mathbb{E}(X_i)\| \leq c$ for all i. Let $X = \sum_{i=1}^m X_i$. Then for any $a > 0$,*

$$\Pr(\|X - \mathbb{E}(X)\| \geq a) \leq 2n \ \exp\Big(- \frac{a^2}{2\| \sum_{i=1}^m \mathrm{Var}(X_i)\| + 2ac/3}\Big).$$

2.1 Our results

We recently proved the following strengthened version of Theorem 1.

Theorem 2 *Let $X_1, X_2, \ldots, X_m$ be independent random $n \times n$ Hermitian matrices. Moreover, assume that $\|X_i\| \leq c$ for all i. Let $X = \sum_{i=1}^m X_i$. Then*

$$\Pr(\lambda_{\max}(X) \geq a) \leq n \ \exp\left(- \frac{(a - \|\mathbb{E}(X)\|)^2}{2\|\sum_{i=1}^m \mathbb{E}(X_i^2)\| + \frac{2c}{3}(a - \|\mathbb{E}(X)\|)}\right) \ \textit{for } a > \|\mathbb{E}(X)\|.$$

In particular,

$$\Pr(\|X\| \geq a) \leq 2n \ \exp\left(- \frac{(a - \|\mathbb{E}(X)\|)^2}{2\|\sum_{i=1}^m \mathbb{E}(X_i^2)\| + \frac{2c}{3}(a - \|\mathbb{E}(X)\|)}\right) \ \textit{for } a > \|\mathbb{E}(X)\|.$$

For the proofs of our main results below, we express the random Hermitian adjacency matrix and random normalized Hermitian Laplacian matrix by sums of independent random Hermitian matrices, and make use of the above inequalities. Our two main results can be stated as follows.

For the first theorem, we use $\Delta(\Gamma(\widehat{G}_n(p_{ij})))$ to denote the maximum expected degree of the underlying graph of $\widehat{G}_n(p_{ij})$. Hence, by straightforward calculations, $\Delta(\Gamma(\widehat{G}_n(p_{ij}))) = \max_{1 \leq i \leq n} \sum_{j=1}^n (p_{ij} + p_{ji} - p_{ij}p_{ji})$.

Theorem 3 *Let $\widehat{G}_n(p_{ij})$ and $H_n = (h_{ij})$ be defined as above, and let $\Delta = \Delta(\Gamma(\widehat{G}_n(p_{ij})))$. Let $\epsilon > 0$ be an arbitrarily small constant, chosen such that for n sufficiently large, $\Delta > \frac{4}{9}\ln(2n/\epsilon)$. Then with probability at least $1 - \epsilon$, for n sufficiently large, the eigenvalues of H_n satisfy*

$$|\lambda_i(H_n)| \leq \max_{1 \leq i \leq n} \sum_{j=1}^n \sqrt{p_{ij}^2 p_{ji}^2 + (p_{ij} - p_{ji})^2} + 2\sqrt{\Delta \ln(2n/\epsilon)}$$

for all $1 \leq i \leq n$.

In the next theorem, we assume that $V(\widehat{G}_n(p_{ij})) = \{v_1, v_2, \ldots, v_n\}$, and we let $D_n = \mathrm{diag}(d_1, d_2, \ldots, d_n)$ denote the diagonal matrix in which d_i is the degree of the vertex v_i in the underlying graph of $\widehat{G}_n(p_{ij})$. We let $\mathbb{E}D_n$ denote the coordinate-wise expectation of D_n. Recall that $\mathcal{L}_n = I_n - D_n^{-1/2} H_n D_n^{-1/2}$ denotes the normalized Hermitian Laplacian matrix of $\widehat{G}_n(p_{ij})$, where I_n denotes the $n \times n$ identity matrix. We let $\delta(\Gamma(\widehat{G}_n(p_{ij})))$ denote the minimum expected degree of the underlying graph of $\widehat{G}_n(p_{ij})$. Hence, $\delta(\Gamma(\widehat{G}_n(p_{ij}))) = \min_{1 \leq i \leq n} \sum_{j=1}^n (p_{ij} + p_{ji} - p_{ij}p_{ji})$.

Theorem 4 *Let $\widehat{G}_n(p_{ij})$, H_n, D_n and $\mathcal{L}_n$ be defined as above, and let $\delta = \delta(\Gamma(\widehat{G}_n(p_{ij})))$. Let $\epsilon > 0$ be an arbitrarily small constant. Then there exists a constant $k = k(\epsilon)$ such that if $\delta > k \ln n$, then with probability at least $1 - \epsilon$, the eigenvalues of $\mathcal{L}_n$ and $\widetilde{\mathcal{L}_n}$ satisfy*

$$|\lambda_i(\mathcal{L}_n) - \lambda_i(\widetilde{\mathcal{L}_n})| \leq 7\sqrt{\frac{\ln(4n/\epsilon)}{\delta}}$$

for all $1 \leq i \leq n$, where $\widetilde{\mathcal{L}_n} = I_n - (\mathbb{E}D_n)^{-1/2}(\mathbb{E}H_n)(\mathbb{E}D_n)^{-1/2}$.

During the talk, we will discuss the main ideas behind the proofs of these results, as well as the relevance of the results for the field.

References

[1] F. Chung, *Spectral graph theory,* AMS publications, 1997.

[2] F. Chung, L. Lu, V.H. Vu, Eigenvalues of random power law graphs, *Ann. Combin.,***7** (2003), 21–33.

[3] F. Chung, L. Lu, V.H. Vu, Spectra of random graphs with given expected degrees, *Proc. Nat. Acad. Sci. USA,* **100(11)** (2003), 6313–6318.

[4] F. Chung, M. Radcliffe, On the Spectra of General Random Graphs, *Electron. J. Combin.,* **18** (2011) P215, 14 pp.

[5] A. Coja-Oghlan, On the Laplacian eigenvalues of $G(n,p)$, *Combin. Probab. Comput.,* **16** (2007), 923–946.

[6] A. Coja-Oghlan, A. Lanka, The spectral gap of random graphs with given expected degrees, *Electron. J. Combin.,* **16** (2009), R138.

[7] D.M. Cvetković, M. Doob, H. Sachs, *Spectra of Graphs, Theory and Applications,* Academic Press, 1980.

[8] U. Feige, E. Ofek, Spectral techniques applied to sparse random graphs, *Random Struct. Alg.*, **27(2)** (2005), 251–275.

[9] J. Friedman, *A Proof of Alon's Second Eigenvalue Conjecture and Related Problem,* Memoirs of the American Mathematical Society 2008, 100 pp.

[10] J. Friedman, On the second eigenvalue and random walks in random d-regular graphs, *Combinatorica,* **11(4)** (1991), 331–362.

[11] J. Friedman, J. Kahn, E. Szemerédi, On the second eigenvalue in random regular graphs, in *Proc. 21st ACM Symp. Theory of Computing*, 1989, 587–598.

[12] Z. Füredi, J. Komlós, The eigenvalues of random symmetric matrices, *Combinatorica,* **1(3)** 1981, 233–241.

[13] K. Guo, B. Mohar, Hermitian adjacency matrix of digraphs and mixed graphs, *J. Graph Theory,* **85** (2017), no. 1, 217–248.

[14] M. Krivelevich, B. Sudakov, The largest eigenvalue of sparse random graphs, *Combin. Probab. Comput.,* **12** (2003), 61–72.

[15] J. Liu, X. Li, Hermitian-adjacency matrices and Hermitian energies of mixed graphs, *Linear. Algebra. Appl.,* **466** (2015) 182–207.

[16] L. Lu, X. Peng, Loose Laplacian spectra of random hypergraphs. *Random Struct. Alg.*, **41** (2012), no. 4, 521–545.

[17] L. Lu, X. Peng, Spectra of edge-independent random graphs, *Electron. J. Combin.,***20** (2013), Paper 27, 18 pp.

[18] R. Oliveira, Concentration of the adjacency matrix and of the Laplacian in random graphs with independent edges, http://arxiv.org/abs/0911.0600.

Probabilistic Analysis of Optimization Problems on Generalized Random Shortest Path Metrics

Stefan Klootwijk, Bodo Manthey, Sander K. Visser

University of Twente, Department of Applied Mathematics, Enschede, The Netherlands

Abstract

A graph $G = (V, E)$ satisfies the α, β-cut-property if the fraction of edges present in each cut of the graph lies between α and β. The Erdős-Rényi random graph $G(n, p)$ satisfies this property w.h.p. for $\alpha = (1 - \varepsilon)p$ and $\beta = (1 + \varepsilon)p$ whenever p is sufficiently large and ε is a suitably chosen constant.

We study the behavior of random shortest path metrics applied to graphs G that satisfy the α, β-cut-property. These random metrics are defined as follows: Let $w(e)$ be independently drawn random edge weights for all $e \in E$, and define $d(u, v)$ to be the shortest path distance between u and v in G with respect to the weights w.

Using the ideas of Bringmann et al. (*Algorithmica, 2015*), who studied random shortest path metrics on the complete graph, i.e., the graph that satisfies the $1, 1$-cut-property, we derive some properties of the metric and obtain a clustering of the vertices. Using this, we conduct a probabilistic analysis of some simple heuristics on these random shortest path metrics.

Keywords : *Random shortest paths, Random metrics, Approximation algorithms, Erdős-Rényi random graph*

1 Introduction

Large-scale optimization problems, such as the traveling salesman problem, show up in many applications and domains all around us. Often it is not possible to solve those problems exactly, since that would take too much time. In practice often ad-hoc heuristics are being used that provide solutions that come quite close to the optimal solutions. In many cases those, often simple, heuristics show a remarkable performance, even though the theoretical results about those heuristics are way more pessimistic.

In order to explain this difference, the method of probabilistic analysis has been widely used over the last decades. In this method, the performance of the heuristics is analysed with respect to a random instance. However, it is not trivial to come up with a good model for such random instances. So far, in almost all cases either Euclidean space has been used to generate instances, or independent, identically distributed edge lengths were used.

However, both approaches fall short of explaining the average-case performance of heuristics on general metric instances. In order to overcome this, Bringmann et al. [2] used the following model for generating random metric spaces, which had been proposed by Karp and Steele [6]. Given an undirected complete graph, start by drawing random edge weights for each edge independently and then define the distance between any two vertices as the total weight of the shortest path between them, measured with respect to the random weights.

Bringmann et al. called this model *random shortest path metrics*. This model is also known as *first-passage percolation*, introduced by Hammersley and Welsh as a model for fluid flow through a (random) porous medium [4, 5].

Our goal is to further broaden the knowledge of metric spaces that can be used to generate random instances, with special interest to the probabilistic analysis of algorithms on those random metric spaces. By making extensive use of the ideas of Bringmann et al. [2] we will extend their results to random shortest path metrics on a class of graphs that naturally arises when considering the Erdős-Rényi random graph $G(n,p)$. To be more precise, we consider arbitrary (fixed) graphs that satisfy what we called the α,β-cut-property.

Definition 1 *Let $0 < \alpha \leq \beta \leq 1$. A finite simple graph $G = (V, E)$ satisfies the α,β-cut-property if for all $\varnothing \subset U \subset V$ it holds that $\alpha\mu_U \leq |\delta(U)| \leq \beta\mu_U$, where $\mu_U := |U|(|V| - |U|)$ is the maximum number of possible edges in the cut defined by U.*

Note that the complete graph satisfies the α,β-cut-property for $\beta = 1$ and any value of $\alpha \leq 1$, so in particular for $\alpha = \beta = 1$. Moreover, for sufficiently large p and suitably chosen constant ε, it follows that the Erdős-Rényi random graph $G(n,p)$ w.h.p. satisfies the α,β-cut-property for $\alpha = (1-\varepsilon)p$ and $\beta = (1+\varepsilon)p$. So, the Erdős-Rényi random graph $G(n,p)$ satisfies the α,β-cut-property w.h.p. for α,β with $\beta/\alpha = O(1)$.

2 Model and Clustering

Let $G = (V, E)$ be a graph on n vertices that satisfies the α,β-cut-property. Let $w : E \to \mathbb{R}_{\geq 0}$ denote the random weights of the edges, independently drawn from the standard exponential distribution. And let $d : V \times V \to \mathbb{R}_{\geq 0}$ denote the shortest path distances with respect to G and w. Note that d defines a metric on V. Also observe that having $\alpha > 0$ ensures that G is connected; this allows us to define d as we did, without having to be careful in case G is not connected and some shortest paths do not exist.

Using of the ideas of Bringmann et al. [2] we find a clustering of the vertices in generalized random shortest path metrics. This clustering is a very helpful tool for the probabilistic analysis of the simple heuristics in the next section.

Lemma 1 *Consider a random shortest path metric on a graph G that satisfies the α,β-cut-property and let $\Delta \geq 0$. The expected number of clusters needed to partition the instance into clusters, each of diameter at most 6Δ, is $O(1 + n/\exp(\beta\Delta n/5))$.*

3 Results

Greedy matching. The greedy algorithm for the minimum-weight perfect matching problem[1] iteratively adds the edge between the two closest unmatched vertices to the matching. Reingold and Tarjan showed that the worst-case approximation ratio for greedy matching on metric instances is $\Theta(n^{\log_2(3/2)})$ [7]. We show that the greedy matching outputs a matching with expected costs at most $O(\beta/\alpha)$. On the other hand, the optimum matching has a total length of $\Omega(1)$. So, we obtain an expected approximation ratio of $O(\beta/\alpha)$.

Theorem 1 *The greedy algorithm for minimum-length perfect matching has an expected approximation ratio of $O(\beta/\alpha)$ on generalized random shortest path metrics.*

The main idea of the proof is to divide the algorithm into several phases defined by the length of the edges that are added to the matching. For the first phases we can bound the maximum number of edges hat is being added in that phase by making use of the clustering property, and we can show that the total expected contribution of the last phases is negligible, in order to obtain the result.

Nearest-neighbor algorithm for TSP. The nearest-neighbor algorithm for the traveling salesman problem starts building a tour from an arbitrary vertex, and then iteratively adds the edge that connects the last added vertex to its closest unvisited vertex (and finally closes the tour by adding the edge that connects the last vertex to the first vertex). Ausiello et

[1]Within our terminology, minimum-distance perfect matching might be a better name for this problem.

al. showed that the worst-case approximation ratio for nearest-neighbor on metric instances is $O(\log(n))$ [1]. We show that nearest-neighbor outputs a tour of length at most $O(\beta/\alpha)$ in expectation. On the other hand, the optimum tour has a total length of $\Omega(1)$. So, we obtain an expected approximation ratio of $O(\beta/\alpha)$.

Theorem 2 *The nearest-neighbor algorithm for the traveling salesman problem has an expected approximation ratio of $O(\beta/\alpha)$ on generalized random shortest path metrics.*

The main idea of the proof is to partition the used edges into several groups defined by the length of the edges. For the groups with the shortest edges we can bound the maximum number of edges in those groups by making use of the clustering property, and we can show that the total expected contribution of the groups with the longest edges is negligible, in order to obtain the result.

Insertion heuristics for TSP. An insertion heuristic for the traveling salesman problem starts building a tour from a small initial tour on a few vertices and then iteratively adds the other vertices to this tour according to some rule. Rosenkrantz et al. showed that the worst-case approximation ratio for any insertion heuristic on metric instances is $O(\log(n))$ [8]. We show that all insertion heuristics output a tour of length at most $O(\beta/\alpha)$ in expectation. On the other hand, the optimum tour still has a total length of $\Omega(1)$. So, we obtain an expected approximation of $O(\beta/\alpha)$.

Theorem 3 *Any insertion heuristic heuristic for the traveling salesman problem has an expected approximation ratio of $O(\beta/\alpha)$ on generalized random shortest path metrics.*

The main idea of the proof is to show that the initial tour has small expected costs and then to partition the partition the added vertices into several groups defined by the additional length added to the tour when the vertex was added to it. We bound the number of vertices in each group, in order to obtain the result.

Running time of 2-opt for TSP. The 2-opt heuristic for the traveling salesman problem starts with an initial tour and then improves this tour by so-called 2-exchanges until no further improvement is possible. A 2-exchange consists of selecting two edges $\{u, v\}$ and $\{x, y\}$ from the tour, removing those edges from the tour and replacing them by the edges $\{u, x\}$ and $\{v, y\}$ to obtain a shorter tour. Even on metric instances, the number of iterations that 2-opt needs to terminate can be $\Omega(2^n)$, as was shown by Englert et al. [3]. We show that the expected number of iterations of 2-opt is polynomially bounded for random shortest path metrics on graphs with the α, β-cut-property.

Theorem 4 *The expected number of iteration that 2-opt needs to find a local optimum is bounded by $O(n^{10} \ln^3(n))$ on generalized random shortest path metrics.*

The main idea of the proof is to use tail bounds for the maximum length of the initial tour and for the minimum improvement made by any 2-exchange. The number of iterations of 2-opt is bounded by the ratio of these values.

***k*-Median.** The k-median problem is to find a subset $U \subseteq V$ with $|U| = k$ such that $\sum_{v \in V} \min_{u \in U} d(u, v)$ is minimized. We consider the (trivial) heuristic that selects the k vertices of U randomly, independent of the metric space, e.g., $U = \{v_1, \ldots, v_k\}$. We call the solution that this heuristic outputs the *trivial solution* to the k-median problem. In general this trivial solution can be arbitrarily bad, even if we only consider metric instances. However, we show that this trivial heuristic yields an $O(\beta/\alpha)$ approximation in expectation whenever k is not too large.

Theorem 5 *If $k \leq (1 - \varepsilon)n$ for some constant $\varepsilon > 0$, then the trivial solution to the k-median problem has an expected approximation ratio of $O(\beta/\alpha)$ on generalized random shortest path metrics. Moreover, for sufficiently small k this expected approximation ratio is given by $(\beta/\alpha)(1 + O(\ln \ln(n/k)/ \ln(n/k)))$.*

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag Berlin Heidelberg, 1999. ISBN 978-3-642-63581-6. doi: 10.1007/978-3-642-58412-1.

[2] K. Bringmann, C. Engels, B. Manthey, and B.V.R. Rao. Random shortest paths: Non-euclidean instances for metric optimization problems. *Algorithmica*, 73(1):42–62, 2015. doi: 10.1007/s00453-014-9901-9.

[3] M. Englert, H. Röglin, and B. Vöcking. Worst case and probabilistic analysis of the 2-opt algorithm for the TSP. *Algorithmica*, 68(1):190–264, 2014. doi: 10.1007/s00453-013-9801-4.

[4] J.M. Hammersley and D.J.A. Welsh. First-passage percolation, subadditive processes, stochastic networks, and generalized renewal theory. In J. Neyman and L.M. Le Cam, editors, *Bernoulli 1713 Bayes 1763 Laplace 1813, Anniversary Volume, Proceedings of an International Research Seminar Statistical Laboratory, University of California, Berkeley 1963*, pages 61–110. Springer Berlin Heidelberg, 1965. doi: 10.1007/978-3-642-49750-6_7.

[5] C.D. Howard. Models of first-passage percolation. In H. Kesten, editor, *Probability on Discrete Structures*, pages 125–173. Springer Berlin Heidelberg, 2004. doi: 10.1007/978-3-662-09444-0_3.

[6] R.M. Karp and J.M. Steele. Probabilistic analysis of heuristics. In *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, chapter 6, pages 181–205. John Wiley & Sons Ltd., 1985. ISBN 978-0-471-90413-7.

[7] E.M. Reingold and R.E. Tarjan. On a greedy heuristic for complete matching. *SIAM Journal on Computing*, 10(4):676–681, 1981. doi: 10.1137/0210050.

[8] D.J. Rosenkrantz, R.E. Stearns, and P.M. Lewis II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977. doi: 10.1137/0206041.

Edge Domination in subclasses of bipartite graphs

B.S.Panda, Shaily Verma

Indian Institute of Technology, Delhi, INDIA

bspanda@maths.iitd.ac.in, shailyverma048@gmail.com

Abstract

An *edge dominating set* of a graph $G = (V, E)$ is a set of edges D such that every edge not in D is adjacent to some edge in D. The minimum cardinality of any edge dominating set of G is called *edge domination number* of G and is denoted by $\gamma'(G)$. Given a graph G and an integer $k, 1 \leq k \leq n$, the Edge Dominating Set Problem is to decide whether $\gamma'(G) \leq k$. It is known that the problem is NP-complete for bipartite graphs. In this paper, we show that this problem remains NP-complete even for perfect elimination bipartite graphs and star-convex bipartite graphs. We have given a linear time algorithm to find the minimum edge domination set of a chain graph.

Keywords : *Edge domination, Perfect Elimination bipartite graphs, star-convex bipartite graphs, chain graphs, NP-completeness, Polynomial time algorithms.*

1 Introduction

A set of edges D of a graph $G = (V, E)$ is said to be an *edge dominating set* if every edge in $E - D$ is adjacent to some edge in D. The *edge domination number* of G, $\gamma'(G)$ is the cardinality of a minimum edge dominating set of G. The set of edges D is said to be an *independent edge dominating set* if D is an edge dominating set and no two edges in D are adjacent. The concept of edge domination and independent edge domination was introduced by Mitchell and Hedetniemi [1] in 1977. In [1], they have given a linear time algorithms to find minimum edge dominating set as well as minimum independent edge dominating set for trees. Later, it has been shown that the size of minimum edge dominating set of a graph G is equal to the size of independent edge dominating set. The decision problem associated with edge domination is as follows:

Edge Dominating Set Problem

Instance: A graph $G = (V, E)$ and a positive integer k

Question: Does G have an edge dominating set with size at most k.

Yannakakis and Gavril [3] in 1980 has showed that the problem remains NP-complete even for planar graphs or bipartite graphs with maximum degree 3. A. Srinivasan, et.al, [2] gave a $O(nm + n^2)$-time algorithm to find a minimum edge dominating set of a bipartite permutation graph in 1995.

2 Preliminaries

All the graphs considered in this paper are simple and undirected. For a graph $G = (V, E)$, the sets $N(v) = \{u \in V(G) | uv \in E\}$ denotes the *open neighborhood* and *closed neighborhood* of a vertex v, respectively. The *degree* of a vertex v in graph G is $|N(v)|$ and is denoted by $d_G(v)$. If $d_G(v) = 1$, then v is called a *pendant vertex.* An edge incident on some pendant vertex is called a *pendant edge.* A graph $G = (V, E)$ is said to be *bipartite* if $V(G)$ can be partitioned into two disjoint sets X and Y such that every edge of G joins a vertex in X to another vertex

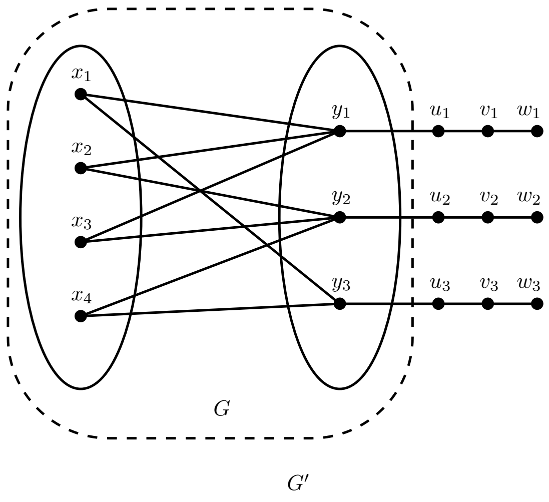


FIG. 1: An example of construction of the graph G'

in Y. Such a partition (X, Y) of V is called a *bipartition.* A bipartite graph with bipartition (X, Y) of V is denoted by $G = (X, Y, E)$.

Let $G = (X, Y, E)$ be a bipartite graph. An edge $e = xy$ is called *bisimplicial edge* if $G[N(x) \cup N(y)]$ is a complete bipartite graph. Let $\sigma = (x_1y_1, x_2y_2, \ldots, x_ky_k)$ be a sequence of pairwise nonadjacent edges of G. Denote $S_j = \{x_1, x_2, \ldots, x_j\} \cup \{y_1, y_2, \ldots, y_j\}$ and let $S_0 = \emptyset$. Then σ is said to be a *perfect edge elimination ordering* for G if each edge $x_{j+1}y_{j+1}$ is bisimplicial in $G = [(X \cup Y) \setminus S_j]$ for $0 \leq j \leq k-1$ and $G = [(X \cup Y) \setminus S_k]$ has no edge. A graph for which there exists a perfect edge elimination ordering is a *perfect elimination bipartite graph.* A bipartite graph $G = (X, Y, E)$ is a *chain graph* if the neighborhoods of the vertices of X form a chain, that is, the vertices of X can be linearly ordered, say $x_1, x_2, \ldots, x_{n_1}$ such that $N(x_1) \subseteq N(x_2) \subseteq \cdots \subseteq N(x_{n_1})$. A bipartite graph $G = (X, Y, E)$ is called a *tree-convex bipartite graph*, if a tree $T = (X, E_X)$ can be defined such that for every vertex y in Y, the neighborhood of y induces a subtree of T. If T is a star, then G is called *star-convex bipartite graph.*

3 NP-Completeness

It is shown that EDGE DOMINATING SET PROBLEM is NP-complete for bipartite graphs [3]. In this section, we prove that the problem remains NP-complete even for perfect elimination bipartite graphs and star-convex bipartite graphs, which are proper subclasses of bipartite graphs.

Theorem 1 EDGE DOMINATING SET PROBLEM *for the perfect elimination bipartite graphs is NP-complete.*

Proof : The reduction has been shown from the EDGE DOMINATING SET PROBLEM in bipartite graphs, which has known to be NP-complete [3]. Let $G = (X \cup Y, E)$ be a bipartite graph with $|X| = m$ and $|Y| = n$. Construct the graph G' from graph G as follows:

For each vertex, $y_i \in Y$, take a copy of P_3 say, $u_iv_iw_i$ and add the edge y_iu_i, for all i, $1 \leq i \leq n$.

The constructed graph G' is a perfect elimination bipartite graph, as $\alpha = (v_1w_1, y_1u_1, v_2w_2, y_2u_2, \ldots, v_nw_n, y_nu_n)$ is a perfect edge elimination ordering of G'. An example of the construction of G' has been shown in Figure 1.

Claim: G has an edge dominating size of at most k if and only if G' has an edge dominating set of size at most $n + k$.

□

Next, we will show that the problem remains NP-complete for star-convex bipartite graphs.

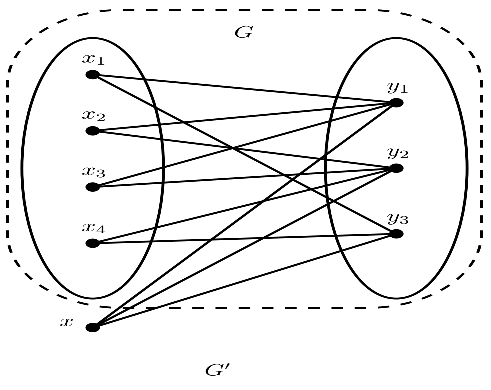


FIG. 2: An example of the construction of the graph G'

Theorem 2 EDGE DOMINATING SET PROBLEM *for the star-convex bipartite graphs is NP-complete.*

Proof : The reduction is from the EDGE DOMINATING SET PROBLEM in bipartite graphs, which has known to be NP-complete [3]. Let $G = (X \cup Y, E)$ be a bipartite graph with $|X| = m$ and $|Y| = n$. Construct the graph G' from graph G as follows:

Take a new vertex x in the partite set X and make it adjacent with every vertex in partite set Y.

Now, we can define a star $S = (X \cup \{x\}, E')$ on the partite set X together with vertex x, such that the vertex x is the centre of the star S. Since every vertex y, $y \in Y$ is adjacent to x, $N(y)$ induces a star, which is a subgraph of S. Thus, the constructed graph G' is a star convex bipartite graph, and the construction of G' has been shown in Figure 2. Next, we have to prove the following claim:

Claim: G has an edge dominating size of at most k if and only if G' has an edge dominating set of size at most $k + 1$.

$\square$

As the star convex bipartite graph is a proper subclass of tree convex bipartite graphs, the next corollary follows immediately:

Corollary 1 EDGE DOMINATING SET PROBLEM *remains NP-complete for the tree convex bipartite graphs.*

4 A linear-time algorithm for chain graphs

Recall that, A. Srinivasan, et.al, [2] gave an algorithm to find a minimum edge dominating set of a bipartite permutation graph in 1995 with complexity $O(nm + n^2)$. In this section we improve the complexity for chain graph, which is a subclass of bipartite permutation graph. We propose a linear time algorithm to find edge dominating set of a chain graph.

Lemma 1 *Let $G = (X \cup Y, E)$ be a chain graph and $\sigma(X) = (x_1, x_2, \ldots, x_{n_1})$ and $\sigma(Y) = (y_1, y_2, \ldots, y_{n_2})$ be the chain ordering of X and Y. Then there exists a minimum edge dominating set of G such that $y_1x_{n_1} \in E(G)$.*

Proof : Let D be a minimum edge dominating set of chain graph G with $|D| = k$. By the definition of chain graph, $y_1x_{n_1} \in E(G)$. By the definition of edge dominating set, at least one end point of each edge should be in $V(D)$. Now if $y_1x_{n_1} \in D$, then we are done. Suppose that $y_1x_{n_1} \notin D$. Then there are three cases:

Case I: $y_1 \in V(D)$, $x_{n_1} \in V(D)$.

Since $y_1, x_{n_1} \in V(D)$ and $y_1x_{n_1} \notin D$, there exist some edges in set D which are incident on y_1 and x_{n_1}, say $y_1x_j \in D$ and $y_ix_{n_2} \in D$, for some $x_j \in X$ and $y_i \in Y$. Now if $y_ix_j \in E(G)$, then

$D \setminus \{y_1x_j, y_ix_{n_1}\} \cup \{y_ix_j, y_1x_{n_1}\}$ is also an edge dominating set. Suppose that $y_ix_j \notin E(G)$. Let $D' = D \setminus \{y_1x_j, y_ix_{n_1}\} \cup \{y_1x_{n_1}\}$. Now if $E(G - V(D')) = \emptyset$, then D' would be an edge dominating set of G with smaller size, which is a contradiction. Therefore $E(G - V(D')) \neq \emptyset$. Suppose that there exists a pair of edges $e_1, e_2 \in E(G - V(D'))$ such that e_1 is incident on x_j but not on vertex y_i and the edge e_2 is incident on y_i but not on x_j. Let $e_1 = yx_j$ and $e_2 = y_ix$, where $x \in X$ and $y \in Y$. Then by the chain property of X, $yx \in E(G)$. If $yx \in E(G - V(D'))$, then the edge yx is not dominated by the edge dominating set D also, which is a contradiction. So $yx \in D$, but in that case edges e_1 and e_2 also dominated by edge $xy \in D$, which is again a contradiction. Therefore, all the edges $e \in E(G - V(D'))$, are either incident on x_j only or on y_i only. So, any one edge $e \in E(G - V(D'))$ together with the set D', dominate all the edges of the graph G. Hence, $D' \cup \{e\}$ is an edge dominating set of G of size k with required property.

Case II: $y_1 \in V(D)$ but $x_{n_1} \notin V(D)$.

Since $y_1 \in V(D)$ and $y_1x_{n_1} \notin D$, there exists an edge in set D which is incident on y_1, say $y_1x_j \in D$, for some $x_j \in X$. Since $x_{n_1} \notin V(D)$, every neighbor of x_{n_1} must belong to $V(D)$. But $N(x_{n_1}) = Y$. Therefore, the set D' such that $D' = D \setminus \{y_1x_i\} \cup \{y_1x_{n_1}\}$ is also an edge dominating set with size k as every edge incident on x_j will dominate by some edge in D' as $Y \subset V(D)$. Hence D' is a minimum edge dominating set with required property.

Case III: $x_{n_1} \in V(D)$ but $y_1 \notin V(D)$.

Since $x_{n_1} \in V(D)$ and $y_1x_{n_1} \notin D$, there exists an edge in set D which is incident on x_{n_1}, say $y_ix_{n_1} \in D$, where $y_i \in Y$. Since $y_1 \notin V(D)$, every neighbor of y_1 must belong to $V(D)$. Now $N(y_1) = X$. Therefore $D' = D \setminus \{y_ix_{n_1}\} \cup \{y_1x_{n_1}\}$ is also an edge dominating set with size k as every edge incident on y_i will dominate by some edge incident on $x \in V(D)$. Hence D' is a minimum edge dominating set with required property.

□

Theorem 3 *Let $G = (X \cup Y, E)$ be a chain graph and $\sigma(X) = (x_1, x_2, \ldots, x_{n_1})$ and $\sigma(Y) = (y_1, y_2, \ldots, y_{n_2})$ be the chain ordering of X and Y. Then, $D = \{y_ix_{n_1+1-i} \in E(G) \mid 1 \leq i \leq \min(\frac{n_1}{2}, \frac{n_2}{2})\}$ is a minimum edge dominating set of G.*

Proof : By Lemma 1, there exists a minimum edge dominating set D of G such that $y_1x_{n_1} \in D$. Now $G_1 = G - \{y_1, x_{n_1}\}$ is again a chain graph. If $y_2x_{n_1-1} \notin E(G)$, $D = \{y_1x_{n_1}\}$ is a minimum edge dominating set of G. Otherwise, again by Lemma 1, there exists a minimum edge dominating set D_1 of graph G_1 such that $y_2x_{n_1-1} \in D_1$ and so $y_2x_{n_1-1} \in D$. Let i be the minimum index such that $y_ix_{n_1-i} \notin E(G)$, where $2 \leq i \leq \min(\frac{n_1}{2}, \frac{n_2}{2})$. In that case, $D = \{y_1x_{n_1}, y_2x_{n_1-1}, \ldots, y_{i-1}x_{n_1-(i-1)}\}$ will be an edge dominating set of G of minimum size. □

Next corollary immediately follows by Theorem 3.

Corollary 2 *The minimum edge dominating set of a chain graph $G = (X, Y, E)$ can be computed in $O(n + m)$ time.*

References

[1] S. Mitchell, S. T. Hedetniemi. *Edge domination in trees.* Congressus Numerantium. 19: 489–509, 1977.

[2] A. Srinivasan, K. Madhukar, P. Nagavamsi, C. Pandu Rangan and Maw-Shang Chang. *Edge domination on bipartite permutation graphs and cotriangulated graphs.* Information Processing Letters. 56: 165–171, 1995.

[3] M. Yannakakis, F. Gavril. *Edge Dominating sets in graphs.* SIAM J. Appl. Math. 38(3): 364–372, 1980.

Listing Conflicting Triples in Optimal Time

Mathias Weller

CNRS, LIGM, Université Paris Est, Marne-la-Vallée, France

Abstract

Different phylogenetic studies might tell different stories about the evolutionary history of a given set of species. This leads to phylogenetic trees that "disagree" on triples of taxa ("conflict triples"). An efficient enumeration of all conflicts exhibited by a pair of phylogenetic trees on n taxa can help inferring phylogenetic trees more efficiently. As it is possible that a significant part of the $\binom{n}{3}$ triples are in conflict, the trivial $\theta(n^3)$-time algorithm that checks for each triple whether it constitutes a conflict, was considered optimal. In this work, we show how to enumerate all d conflict triples between a pair of phylogenetic trees in $O(n+d)$ time. Since any deterministic algorithm has to spend $\Omega(n)$ time reading the input and $\Omega(d)$ time writing the output, no deterministic algorithm can solve this task faster than we do (up to constant factors).

Keywords : *phylogenetic trees, parameterized algorithms, enumeration, total linear time*

1 Introduction

Evolutionary ("phylogenetic") trees are fundamental when studying the development of species on earth. However, depending on the source, they might imply different evolutionary histories of the same taxa. Such contradictions manifest themselves as "conflict triples" (or "conflicts" for short), that is, three taxa a, b, c such that one phylogenetic tree P implies that a common ancestor of a and b split off the common lineage of a, b and c before splitting into a and b while another tree Q implies that a common ancestor of b and c split off the common lineage before splitting into b and c. More formally, $\text{LCA}_P(ab) \neq \text{LCA}_P(abc)$ and $\text{LCA}_Q(bc) \neq \text{LCA}_Q(ab) = \text{LCA}_Q(abc)$. Conflict triples are essential ingredients to building, reconciling, and comparing phylogenetic trees [3, 4, 7, 8]. While *counting* the number of conflicts has received some attention in the past [2], not much work has been done on *enumerating* them. Such development might have been discouraged by the fact that a significant portion of the $\binom{n}{3}$ triples of taxa might be in conflict, in which case the trivial algorithm that tests each triple of taxa for being a conflict would be optimal. This work emerged from the question whether we can do better if only few triples are actually in conflict. Indeed, we show how to enumerate all d conflict triples of a pair (P, Q) of phylogenetic trees on n taxa in $O(n + d)$ time ("total linear time"). Since all algorithms solving this problem need to read the input (size $\Theta(n)$) and write the output (size $\Theta(d)$), this is asymptotically "best possible".

2 Preliminaries

A *(phylogenetic) tree* T is a rooted, binary outbranching whose leaves $\mathcal{L}(T)$ are bijectively labeled by a set X (of taxa) and we refer to its root by $r(T)$. Since the labeling is bijective, we use leaves and labels interchangeably. If some vertex v of T is a strict ancestor of a vertex u in T, we write $u <_T v$ and we abbreviate $\forall_{v \in Z}\, v <_T u$ to $Z <_T u$. We also abbreviate sets of leaves (or labels) by the concatenation of their names, that is, abc refers to $\{a, b, c\}$. The *least common ancestor* of two leaves (or labels) a and b in T is the minimum among all u with $ab <_T u$ and we write $\text{LCA}_T(ab) = u$. The subtree of T rooted at u is denoted

```
Procedure ListSubtreeConflicts
  Input: Tree $T$, leaf subset $Z \subseteq \mathcal{L}(T)$ in post-order.
  Output: Triples $abc$ with $a, b \in Z$, and $c \in \mathcal{L}(T) \setminus Z$, and $ab \not|_T c$
1 if $Z \neq \varnothing$ then
2     foreach $c \in \mathcal{L}(T) \setminus Z$ do
3         $T' \leftarrow T|_{Z \cup \{c\}}$;
4         $y \leftarrow$ parent of $c$;
5         while $y \neq r(T')$ do
6             $x \leftarrow$ sibling of $y$ in $T'$;
7             foreach $a \in \mathcal{L}(T'_y) \setminus \{c\}$ and $b \in \mathcal{L}(T'_x)$ do list $abc$;
8             $y \leftarrow$ parent of $y$ in $T'$;
```

by T_u and the *restriction* $T|_L$ of T to a set $L \subseteq \mathcal{L}(T)$ is the unique tree on the vertex set $\{\mathrm{LCA}_T(x, y) | x, y \in L\}$ such that $<_T$ is an extension of $<_{T|_L}$. In this work a *triple* abc in T is a set of three labels $abc \subseteq X$. We say that abc *touches* $\mathrm{LCA}_T(abc)$ and omit the mention of T if it is clear from context. We say a triple abc is ab*-biased* in T if $\mathrm{LCA}_T(ab) \neq \mathrm{LCA}_T(abc)$ and we write $ab|_T c$ to indicate this fact. A triple abc is called a *conflict* of a pair (P, Q) of trees if, for some $xy \subseteq abc$, we have that abc is xy-biased in exactly one of P and Q. Recall that abc and cab refers to the same conflict, so when claiming that abc is not listed twice, this also means that no two permutations of abc are listed. For two vertices $u \in V(P)$ and $v \in V(Q)$, we define $u \sqcap v := \mathcal{L}(P_u) \cap \mathcal{L}(Q_v)$ and $u \wr v := \mathcal{L}(P_u) \setminus \mathcal{L}(Q_v)$. Note that $\sqcap$ is symmetrical while $\wr$ is not. Also note that, for children u_P, v_P of $r(P)$ and children u_Q, v_Q of $r(Q)$, we have $u_P \wr u_Q = u_P \sqcap v_Q = v_Q \sqcap u_P = v_Q \wr v_P$.

In the following, we call a tree T *LCA-enabled* if the LCA of any two vertices in T can be found in constant time. Note that we can LCA-enable any tree in linear time [1, 6]. In the algorithm, we will want to compute the subtree T' of a tree T that is induced by a set Z of leaves.

Observation 1 **([5, Section 8])** *Let T be an LCA-enabled tree and let $Z \subseteq \mathcal{L}(T)$ be in post-order. Then, $T|_Z$ can be computed in $O(|Z|)$ time.*

Let P and Q be trees and let $u \in V(P)$ and $v \in V(Q)$. To verify $\mathcal{L}(P_u) = \mathcal{L}(Q_v)$ in constant time, we use an $O(|P| + |Q|)$-time preprocessing: first, construct a mapping $m : V(P) \to V(Q)$ such that $m(x) := \mathrm{LCA}_Q(\mathcal{L}(P_x))$ using the recursion $m(x) = \mathrm{LCA}_Q(m(x'), m(x''))$ where x' and x'' are the children of x in P. Second, store the number of leaves below each vertex in P or Q. Then, $\mathcal{L}(P_u) = \mathcal{L}(Q_v)$ can be tested as $|\mathcal{L}(P_u)| = |\mathcal{L}(Q_{\mathrm{LCA}_Q(\mathcal{L}(P_u))})|$.

Observation 2 *With an $O(|P| + |Q|)$-time preprocessing, we can decide $\mathcal{L}(P_u) = \mathcal{L}(Q_v)$ in constant time for each u and v.*

3 The Algorithm

In this section, we assume that we are given a pair (P, Q) of phylogenetic trees on the label-set X. Our algorithm starts off by arbitrarily pairing the children of $r(P)$ with the children of $r(Q)$ and we let $\{(u_P, u_Q), (v_P, v_Q)\}$ be this pairing. Observe that any conflict triple abc that does not touch $r(T)$ is completely contained in T_{u_T} or T_{v_T} for any $T \in \{P, Q\}$. Thus, all conflicts that touch neither $r(P)$ nor $r(Q)$ are in one of $u_P \sqcap u_Q$, $u_P \sqcap v_Q$, $v_P \sqcap u_Q$, or $v_P \sqcap v_Q$ and these sets are disjoint. Thus, our algorithm will first list all conflict triples abc that touch $r(P)$ or $r(Q)$ and then recurse into the subtrees of P and Q induced by these sets.

Root-touching conflicts. In the following, we consider a conflict abc touching $r(P)$ and, without loss of generality, suppose that abc is ab-biased and let $x_P \in \{u_P, v_P\}$ such that $ab <_P x_P$. Further, let $x_Q \in \{u_Q, v_Q\}$.

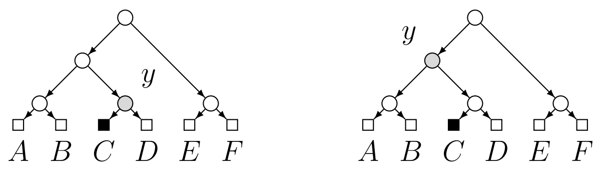


FIG. 1: An example illustrating the tree T' in two steps of `ListSubtreeConflicts` (gray = vertex y, black = leaf c with label C). Left: first step (y is the parent of c), listing DAC and DBC. Right: second step, listing all abC, with $a \in \{A, B, D\}$ and $b \in \{E, F\}$.

First, suppose that abc also touches $r(Q)$. Then, exactly one of ab is in Q_{x_Q} since, otherwise, abc is either not a conflict or not touching $r(Q)$. Thus, we can suppose without loss of generality that $a \in x_P \sqcap x_Q$, $b \in x_P \wr x_Q$, and $c \notin \mathcal{L}(P_{x_P})$. Hence, we can list all conflicts touching $r(P)$ and $r(Q)$ by listing all elements of $(x_P \sqcap x_Q) \times (x_P \wr x_Q) \times (X \setminus \mathcal{L}(x_P))$ for all pairs $(x_P, x_Q) \in \{(u_P, v_P), (u_Q, v_Q)\}$.

Second, suppose that abc touches $r(P)$ but not $r(Q)$ and let $y_P \in \{u_P, v_P\} - x_P$. Then, abc is either in Q_{u_Q} or in Q_{v_Q}. In the first case, $a, b \in u_Q \sqcap x_P$ and $c \in u_Q \wr x_P$ and, in the second case, $a, b \in v_Q \sqcap x_P$ and $c \in v_Q \wr x_P$. Equivalently, in the second case, $a, b \in v_Q \wr y_P$ and $c \in v_Q \sqcap y_P$ and, in the first case, $a, b \in u_Q \wr y_P$ and $c \in u_Q \sqcap y_P$. Thus, the elements of

$$\bigcup_{(x_P,x_Q)\in\{(u_P,u_Q),(v_P,v_Q)\}} \Big((x_Q \sqcap x_P)^2 \times (x_Q \wr x_P)\Big) \cup \Big((x_Q \wr x_P)^2 \times (x_Q \sqcap x_P)\Big)$$

are independent of the initial matching between the children of $r(P)$ and $r(Q)$ and we list all triples a, b, c contained in this set for which $ab\nmid_Q c$. To stay within our running-time bounds, however, we have to make sure that we can compile this list in output-linear time. To this end, we use `ListSubtreeConflicts` to list all the triples abc with $a, b \in x_Q \sqcap x_P$ and $c \in x_Q \wr x_P$ (or vice versa), and $ab\nmid_Q c$ in constant time per listed triple (see Figure 1 for an illustration). The idea is (i) to focus on the subtree Q' of Q that is rooted at $\mathrm{LCA}_Q(x_Q \sqcap x_P)$, (ii) to pick any leaf $c \in x_Q \wr x_P$ and, (iii) for each y on the unique path from c to $r(Q')$, listing all triples abc for which $a, c <_Q y$ and $b \not<_Q y$, thereby ensuring $\mathrm{LCA}_Q(ac) \neq \mathrm{LCA}_Q(abc)$.

Lemma 1 `ListSubtreeConflicts` *is correct, that is, it outputs a triple abc if and only if* $a, b \in Z$, $c \notin Z$, *and* $ab\nmid_T c$. *Further, the procedure takes* $O(d)$ *time (where d is the total number of listed triples) and no triple is listed twice.*

Bounding Set-Computations. With Lemma 1, we can finally list all d_r conflict triples abc with $\mathrm{LCA}_P(abc) = r(P)$ or $\mathrm{LCA}_Q(abc) = r(Q)$ in $O(d_r)$ time. Thus, our algorithm completes the following tasks in the mentioned times.
Task (a): list all conflict triples touching $r(P)$ or $r(Q)$: $O(d_r)$ time;
Task (b): compute common and uncommon leaves: $O(|X|)$ time;
Task (c): compute the subtrees induced by these leaf-sets: $O(|X|)$ time;
Task (d): preprocess these subtrees for the recursive calls: $O(|X|)$ time;
Task (e): make recursive calls
The algorithm in its current form has a worst-case running time of $O(|X|^2)$. In the following, we show how to avoid the costly computations of (b), (c), and (d) if they are not necessary and bound their running-time in $O(d_r)$ if they are. To this end, note that our algorithm outputs

$$|u_P \sqcap u_Q| \cdot |u_P \wr u_Q| \cdot (|v_P \sqcap v_Q| + |v_P \wr v_Q|) \le d_r$$

unique conflicts touching both roots. Thus, if $u_P \sqcap u_Q \neq \varnothing$ and $u_P \wr u_Q \neq \varnothing$, then

$$\begin{aligned} |X| &= |u_P \sqcap u_Q| + |u_P \wr u_Q| + |v_P \sqcap v_Q| + |v_P \wr v_Q| \\ &\le |u_P \sqcap u_Q| \cdot |u_P \wr u_Q| \cdot (|v_P \sqcap v_Q| + |v_P \wr v_Q|) + 2 \le d_r + 2 \end{aligned}$$

and we can thus bound the time spent for (b), (c), and (d) in $O(d_r)$. By symmetry, the same holds if $v_P \sqcap v_Q \neq \varnothing$ and $v_P \wr v_Q \neq \varnothing$. It remains to explore the cases that one of $u_P \sqcap u_Q$ and $u_P \wr u_Q$ and one of $v_P \sqcap v_Q$ and $v_P \wr v_Q$ is empty.

First, $u_P \wr u_Q = v_P \sqcap v_Q = \varnothing$. Then all leaves of P_{u_P} are leaves of Q_{u_Q} and all leaves of P_{v_P} are not leaves of Q_{v_Q}. Thus, Q_{v_Q} does not have any leaves, contradicting the fact that P and Q are binary trees. Symmetrically, $u_P \sqcap u_Q = v_P \wr v_Q = \varnothing$ cannot happen.

Second, $u_P \wr u_Q = v_P \wr v_Q = \varnothing$. Then, $\mathcal{L}(P_{u_P}) = \mathcal{L}(Q_{u_Q})$ and $\mathcal{L}(P_{v_P}) = \mathcal{L}(Q_{v_Q})$. This situation can be detected in constant time, given a linear-time preprocessing of P and Q that links a node x_P of P to a node x_Q of Q if and only if P_{x_P} and Q_{x_Q} have the same leaf-set (see Observation 2). In this case, there are no root-conflicts and none of the costly steps (b)–(d) are necessary.

Third, $u_P \sqcap u_Q = v_P \sqcap v_Q = \varnothing$. Then, changing the root-child pairing to (u_P, v_Q) and (v_P, u_Q) gives the previous case. The same preprocessing allows us to detect and deal with this.

Theorem 1 *Given phylogenetic trees P and Q on the same set of n taxa, we can enumerate all d conflict triples in $O(n+d)$ time and no triplet is output more than once.*

4 Conclusion

We have shown how to list all d conflicts between two phylogenetic trees in $O(n+d)$ time where n is the number of taxa and d is the number of listed conflicts. This improves the previously used, trivial $\Theta(n^3)$-time algorithm that tests for each leaf-triple *abc* for being a conflict. The presented algorithm is fastest-possible (up to constant factors), since all algorithms solving the problem must at least read the input and write the output. A simple next step is to extend the algorithm to non-binary outbranchings. More challengingly, we want to reconsider other polynomial-time enumeration problems parameterized by the length of the output list in hope to produce more "fastest-possible" algorithms. We also plan to analyze real-world phylogenetic trees to see whether the parameter is sufficiently smaller than n^3 to make it worth implementing in practice.

Acknowledgments Big thanks go to the *Institut de Biologie Computationelle* for funding my research, as well as Krister Swenson and Celine Scornavacca for fruitful discussions.

References

[1] M. A. Bender and M. Farach-Colton. The LCA problem revisited. In *LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000, Proceedings*, volume 1776 of *LNCS*, pages 88–94. Springer, 2000.

[2] G. S. Brodal, R. Fagerberg, T. Mailund, C. N. Pedersen, and A. Sand. Efficient algorithms for computing the triplet and quartet distance between trees of arbitrary degree. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1814–1832. Society for Industrial and Applied Mathematics, 2013.

[3] J. Byrka, S. Guillemot, and J. Jansson. New results on optimizing rooted triplets consistency. *Discrete Applied Mathematics*, 158(11):1136 – 1147, 2010. ISSN 0166-218X. doi: 10.1016/j.dam.2010.03.004.

[4] C. Chauve, M. Jones, M. Lafond, C. Scornavacca, and M. Weller. Constructing a consensus phylogeny from a leaf-removal distance. under review, 2017.

[5] R. Cole, M. Farach-Colton, R. Hariharan, T. Przytycka, and M. Thorup. An $o(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. *SIAM Journal on Computing*, 30(5):1385–1404, 2000.

[6] D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2):338–355, 1984.

[7] J. Jansson, N. B. Nguyen, and W.-K. Sung. Algorithms for combining rooted triplets into a galled phylogenetic network. *SIAM Journal on Computing*, 35(5):1098–1121, 2006. doi: 10.1137/S0097539704446529.

[8] V. Ranwez, A. Criscuolo, and E. J. Douzery. Supertriplets: A triplet-based supertree approach to phylogenomics. *Bioinformatics*, (26):i115–i123, 2010.

How to exploit structural properties of dynamic networks to detect nodes with high temporal closeness

Marwan Ghanem[1], Clémence Magnien[1], Fabien Tarissan[2]

[1] Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France
`firstname.lastname@lip6.fr`

[2] Université Paris-Saclay, CNRS, ENS Paris-Saclay, ISP UMR 7220, France
`fabien.tarissan@ens-paris-saclay.fr`

Abstract

The ability to detect important nodes in temporal networks has been investigated lately. This has been a challenge on both the theoretical aspects as well as computational ones. In this study we propose and evaluate different strategies to detect nodes that have high temporal closeness.

Keywords : *Temporal closeness, Sampling, Dynamic network*

1 Introduction and definitions

Evaluating the importance of nodes in complex networks has been an interesting question for a long time. Several measures of importance have been introduced, such as degree, closeness or betweenness centrality. As complex networks have grown in size, approximation methods have been introduced. One of the first method to approximate centrality was introduced in [6]. They consider k source nodes selected randomly, from which they compute the shortest-paths with all other nodes of the network. Since then, several methods have been proposed to help selecting the source nodes [4] or the target nodes [8] in order to reduce the computation required to estimate the closeness and betweenness centrality Those studies all consider a single and static network. However, most of real application involve networks whose structure evolves with time. This led the community to propose adaptation of centrality metrics to assess the importance of nodes through time [7, 11]. This temporal dimension makes the computation more demanding, making methods for approximating centrality metrics even more essential. In this study, we study how structural properties of dynamic networks can be exploited to detect nodes that have a high *temporal closeness centrality* [7].
More precisely, let $G = (V, E)$ be a dynamic network composed of a set V of nodes and a set E of temporal links of the form (u, v, t) where $u, v \in V$ and t is a timestamp. A temporal path from u to v starting at time t_s is given by a sequence of links $(u, v_0, t_0), (v_0, v_1, t_1), \ldots, (v_{k-1}, v, t_k)$ such that $t_0 > t_s$ and, for all $i, i = 0..k-1$, $t_i < t_{i+1}$.

Such a path is a *shortest path* if it has the least duration $(t_k - t_s)$ among all paths from u to v starting at time t_s. The *(temporal) distance* from u to v at time t_s is then the duration of such a shortest path (denoted $d_{t_s}(u,v)$)[1]. Following the classical definition of the closeness of a node in a static network, the *temporal closeness* of a node u at time t is defined by:

$$C_t(u) = \sum_{v \neq u} \frac{1}{d_t(u,v)}$$

It measures the importance of node u at time t in the dynamic network. In order to assess what nodes are important at time t, one can rank the nodes according to their temporal closeness at time t and consider for instance the top 25% rankings. This enables in turn to compute for each node u its total duration spent in the top 25% rankings (denoted by $Dur_{top}(u)$). It measures the *global* importance of node u in G. The purpose of the present study is to propose strategies to detect which nodes are globally important without relying on the exact computation of the temporal closeness of all nodes at all time instant.

2 Strategies and results

In order to detect globally important nodes, we propose to first compute global properties of the nodes that can easily be extracted either from the aggregated graph $G_A = (V, E_A)$ (with $E_A = \{(u,v) | \exists t, (u,v,t) \in E\}$) or from an analysis of the temporal activity. For every node u, we compute its closeness centrality $CC(u)$, its degree centrality $DC(u)$ and its number of links $NL(u)$ – all computed on G_A – as well as its duration of activity $DU(u)$[2] and its average inter-contact duration time $LD(u)$[3]. Then we propose:
Parameter based strategy (P_1/P_2): we consider the rankings given by mixing the importance measured by P_1 and P_2 defined by: $R(u) = \alpha \times \texttt{rank}(P_1(u)) + (1-\alpha) \times \texttt{rank}(P_2(u))$ with $\alpha \in [0:1]$[4] and where $\texttt{rank}(P)$ is the rank provided by property P.
Parameterless strategy (PS): we only take into account the number of links and the duration of activity: $R(u) = \texttt{rank}(NL(u) \times DU(u))$.
In order to assess the relevance of each strategy (and for any α), we compute the number of nodes correctly detected as important[5] in the top k nodes (for $k \in [1..n]$) and denote this vector as the *hit rate* vector. The hit rate vector of a perfect strategy would then be equal to $[1, 2, .., n]$. From these vectors we can compute the distance between any strategy and the perfect strategy and normalize it by the worse case strategy. Formally, we define the score of a strategy by: $score(S) = 1 - \frac{distance(perfect\_strategy, S)}{distance(perfect\_strategy, worse\_case)}$
Figure 1 shows the scores for all the strategies (with different values for α) when applied on nine datasets whose characteristics are provided in Table 1. We observe that in most cases NL/DU,DU/LD and PS score higher than other combinations as well as any pure static centralities. They are much closer to a perfect strategy or ground truth than any

[1] $d_{t_s}(u,v) = \infty$ if there is no path between u and v.
[2] the difference between that last and the first activity.
[3] the average time between two consecutive links involving u.
[4] note that $\alpha = 1$ implies that only P_1 is considered.
[5] we consider the exact computation of the temporal closeness as the ground truth.

Datasets	Type	#Nodes	#Edges	Duration	Ref
Enron	Email	151	47 088	3 years	[10]
Radoslaw	Email	168	82 876	9 months	[9]
DNC	Email	1891	39 264	2.6 years	[1]
HashTags	Social Network	3 048	100 429	22 days	-
Facebook	Social Network	8 977	66 153	1 year	[12]
Article Tags	Social Network	2902	571 877	10 years	-
Reality Mining	Movement	96	1 M	9 month	[5]
Taxi Rome	Movement	158	241 736	1 day	[3]
Primary	Movement	242	125 773	1.5 days	[2]

TAB. 1: Dataset, Type, Number of nodes, Number of links, Duration

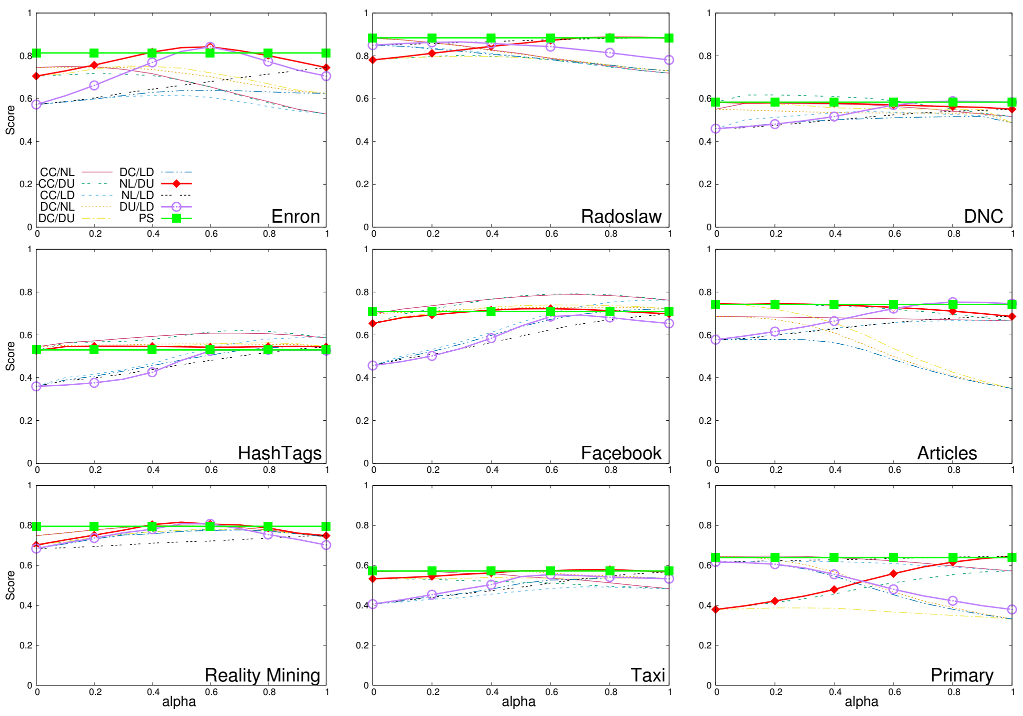


FIG. 1: Score for each strategy on the nine datasets

other strategies. In addition, we can observe that datasets of same nature lead to similar α value for the best strategies.

3 Conclusions

In this study we proposed different strategies that rely on global properties of nodes to detect nodes with high temporal closeness centrality. In most cases, three strategies present the best results. They all take into account temporal properties of the nodes.

This work is a first step to adapt recent technics [6, 4, 8] to approximate the importance of nodes in dynamics networks.

Acknowledgements

This work is funded in part by the European Commission H2020 FETPROACT 2016-2017 program under grant 732942 (ODYCCEUS), by the ANR (French National Agency of Research) under grants ANR-15-CE38-0001 (AlgoDiv) and ANR-13-CORD-0017-01 (CODDDE), by the French program "PIA - Usages, services et contenus innovants" under grant O18062-44430 (REQUEST), and by the Ile-de-France program FUI21 under grant 16010629 (iTRAC).

References

[1] Dnc co-recipient network dataset – KONECT, September 2016.

[2] A.L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435(7039):207–211, 2005.

[3] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from `https://crawdad.org/roma/taxi/20140717`, July 2014.

[4] Ulrik Brandes and Christian Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, 17(7):2303–2318, 2007.

[5] Nathan Eagle and Alex (Sandy) Pentland. CRAWDAD dataset mit/reality (v. 2005-07-01). Downloaded from `https://crawdad.org/mit/reality/20050701`, July 2005.

[6] David Eppstein and Joseph Wang. Fast approximation of centrality. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 228–229, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[7] Clémence Magnien and Fabien Tarissan. Time evolution of the importance of nodes in dynamic networks. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ASONAM '15, pages 1200–1207, New York, NY, USA, 2015. ACM.

[8] Arun S. Maiya and Tanya Y. Berger-Wolf. Online sampling of high centrality individuals in social networks. In *Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part I*, PAKDD'10, pages 91–98. Springer-Verlag, 2010.

[9] Radosław Michalski, Sebastian Palus, and Przemysław Kazienko. Matching organizational structure and social network extracted from email communication. In *Lecture Notes in Business Information Processing*, volume 87, pages 197–206. Springer Berlin Heidelberg, 2011.

[10] Jitesh Shetty and Jafar Adibi. Discovering important nodes through graph entropy the case of Enron email database. In *Proceedings of the 3rd international workshop on Link discovery - LinkKDD '05*, pages 74–81, New York, New York, USA, August 2005. ACM Press.

[11] Taro Takaguchi, Yosuke Yano, and Yuichi Yoshida. Coverage centralities for temporal networks. *The European Physical Journal B*, 89(2):35, 2016.

[12] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. On the evolution of user interaction in Facebook. In *Proc. Workshop on Online Social Networks*, pages 37–42, 2009.

Temporal matching in link stream: kernel and approximation

Julien Baste[1]   Binh-Minh Bui-Xuan[1]

Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France
`[julien.baste,buixuan]@lip6.fr`

Abstract

A link stream is a sequence of pairs of the form $(t, \{u, v\})$, where $t \in \mathbb{N}$ represents a time instant and $u \neq v$. Given an integer γ, the γ-edge between vertices u and v, starting at time t, is the set of temporally consecutive edges defined as $\{(t', \{u, v\}) \mid t' \in [\![t, t + \gamma - 1]\!]\}$. We introduce the notion of temporal matching of a link stream to be a set of pairwise non overlapping γ-edges belonging to the link stream. Unexpectedly, the problem of computing a temporal matching of maximum size turns out to be NP-hard. We provide a kernelization algorithm parameterized by the solution size for the problem. As a byproduct we also depict a 2-approximation algorithm.

Keywords : *graph, parameterized algorithm, link stream.*

1 Introduction

The question of mining data stemming from human activities not only comes from an old and well studied topic of social science. In the recent years, mining human data is also moved by the sheer amount of applications in web analytics, graph mining statistics, criminology graph visualization and so on. An important, yet not quite well understood, feature of graph data collected by such tools comes from the time dimension: edges here are timestamped edges. They come ordered by the time interval where they are effectively active. We call this kind of data a link stream, in the sense of [3, 4].

A link stream L is a sequence of pairs of the form $(t, \{u, v\})$ where $\{u, v\}$ is an edge, in the sense of classical loopless undirected simple graphs, and $t \in \mathbb{N}$ is an integer representing a discretized time instant. If every pair $(t, \{u, v\})$ in L satisfies $t = t_0$ for some fixed t_0, then we say that link stream L is constant. A constant link stream is equivalent to the formalism of a graph in the classical sense. Given an integer γ, a time instant t, and two distinct vertices u and v, we define the γ-edge between u and v starting at time t as the set $\{(t', \{u, v\}) \mid t' \in [\![t, t + \gamma - 1]\!]\}$. Two γ-edges are compatible when they are not overlapping (cf. formal definition in Section 2). Finally, a γ-matching of link stream L is a set of compatible γ-edges where each γ-edge contains exclusively edges from L. We consider the problem of computing a maximum γ-matching of an input link stream, that we call γ-MATCHING. When $\gamma = 1$, this problem can be solved by a slight extension of the notorious polynomial time algorithm given in [2].

Unfortunately, we found that γ-MATCHING on arbitrary input is NP-hard, as soon as $\gamma > 1$. We address the question of pre-processing, in polynomial time, an input instance of γ-MATCHING, in order to reduce it to an equivalent instance of smaller size, in the sense of kernelization algorithms introduced by [1]. We show that γ-MATCHING when parameterized by the solution size admits a quadratic kernel. On the way to do so, we also depict a procedure which turns out to define a 2-approximation algorithm for γ-MATCHING. Our paper is organised as follows. We first introduce the notion of temporal matching (Section 2), before presenting our algorithmic tools (Section 3) in order to obtain our main result, the kernelization algorithm (Section 4). We close the paper with concluding remarks and directions for further research (Section 5).

2 Temporal matching

We denote by $\mathbb{N}$ the set of nonnegative integer. Given two integers p and q, we denote by $[\![p, q]\!]$ the set $\{r \in \mathbb{N} \mid p \leq r \leq q\}$. A *link stream* L is a triple (T, V, E) such that $T \subseteq \mathbb{N}$, V is a set, and $E \subseteq T \times \binom{V}{2}$. The elements of V are called *vertices* and the elements of E are called *(timed) edges*. A *temporal vertex* of L is a pair (t, v) such that $t \in T$ and $v \in V$.

Given an integer γ, a γ-edge between two vertices u and v at time t, denoted $\Gamma_\gamma(t, u, v)$, is the set $\{(t', \{u, v\}) \mid t' \in [\![t, t + \gamma - 1]\!]\}$. We say that a γ-edge Λ is incident to temporal vertex (t, v) if there exists a vertex $u \in V$ such that $(t, \{u, v\}) \in \Lambda$. We say that two γ-edges are compatible if there is no temporal vertex (t, v) that is incident to both of them. A γ-matching $\mathcal{M}$ of a link stream L is a set of pairwise compatible γ-edges. We say that a γ-edge Λ is incident with a vertex $v \in V$ if there exist a vertex $u \in V$ and an integer $t \in T$ such that $\Lambda = \Gamma_\gamma(t, u, v)$. We say that an edge $e \in E$ is in a γ-matching $\mathcal{M}$ if there exists $\Lambda \in \mathcal{M}$ such that $e \in \Lambda$.

This paper focus on the following problem.

γ-MATCHING
Input: A link stream L and an integer k.
Output: A γ-matching of L of size k or a correct answer that such a set does not exist.

Property 1 *γ-MATCHING is NP-hard.*

Sketch of the proof. We reduce from 3-SAT, that is well known to be NP-hard. Let φ be a formula with n variables $x_1, \ldots, x_n$ and m clauses $C_0, \ldots, C_{m-1}$ such that each clauses is of size at most 3. Without loss of generality, we assume that a clause does not contain twice the same variable. We call X the set containing the n variables and $\mathcal{C}$ the set containing the m clauses.

We define the linkstream $L = (T, V, E)$ in the following way: $T = [\![0, (m + 1)\gamma - 1]\!]$, $V = \{x^-, x^=, x^+ \mid x \in X\} \cup \{x_t^{++}, x_t^{--} \mid x \in X, t \in [\![0, m - 1]\!]\} \cup \{c\}$, and $E = E_{var} \cup E_{cla}$ where:

$$\begin{aligned}
E_{var} &= \{(t, \{x^=, x^+\}), (t, \{x^=, x^-\}) \mid t \in [\![0, (m + 1)\gamma - 1]\!], x \in X\} \\
&\cup \quad \{(t, \{x^+, x_i^{++}\}), (t, \{x^-, x_i^{--}\}) \mid t \in [\![1, m\gamma]\!], x \in X, i = \left\lfloor \frac{t-1}{\gamma} \right\rfloor\} \\
E_{cla} &= \{(t, \{c, x_i^{++}\}) \mid t \in [\![i\gamma + 1, (i + 1)\gamma]\!], i \in [\![0, m - 1]\!], x \in X, x \text{ appears positively in } C_i\} \\
&\cup \quad \{(t, \{c, x_i^{--}\}) \mid t \in [\![i\gamma + 1, (i + 1)\gamma]\!], i \in [\![0, m - 1]\!], x \in X, x \text{ appears negatively in } C_i\}.
\end{aligned}$$

We depict in Figure 1 the linkstream build for $\gamma = 3$ and $\varphi = (\overline{w} \vee x \vee \overline{y}) \wedge (\overline{w} \vee \overline{x} \vee z)$. In order to finish the proof of the property, it is sufficient to show the equivalence between above gadget and the original instance of 3-SAT, that is:

Claim 1 *There is an assignment of the variables that satisfies φ if and only if L contains a γ-matching of size $(2m + 1)n + m$.*

Due to space limit, we omit the formal proof of Claim 1. Intuitively, the edge between $(0, \{x^=, x^+\})$ and $(0, \{x^=, x^-\})$ that is in the requested γ-matching determines if the variable x is set to true or false. Moreover, the size of the requested γ-matching ensures that if the edge $(0, \{x^=, x^+\})$ (resp. $(0, \{x^=, x^-\})$) is in the γ-matching, then every edge $(t, \{x^=, x^+\})$ (resp. $(t, \{x^=, x^-\})$), $t \in [\![0, (m + 1)\gamma - 1]\!]$, and every edge $(t, \{x^-, x_i^{--}\})$ (resp. $(t, \{x^+, x_i^{++}\})$), $t \in [\![1, m\gamma]\!]\}$, $i = \left\lfloor \frac{t-1}{\gamma} \right\rfloor$, are in the γ-matching as well. Finally, during the time interval $[\![i\gamma + 1, (i + 1)\gamma]\!]$, we check that the clause C_i is satisfied.

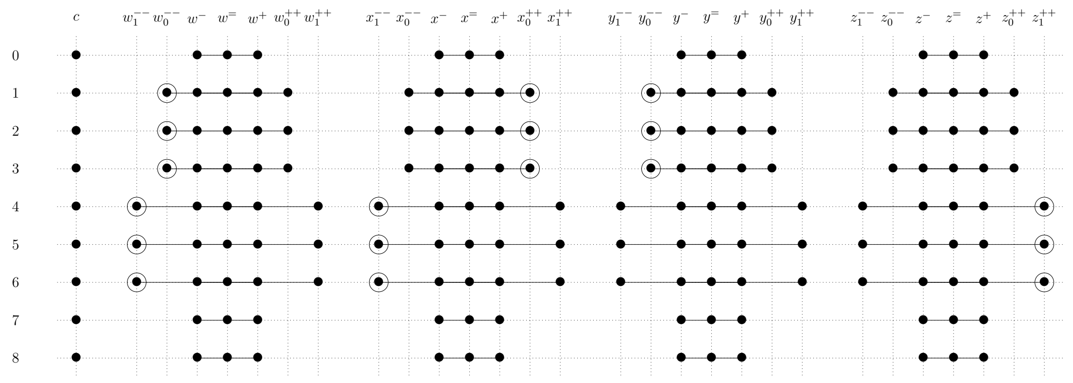

c
w_1^{--} w_0^{--} w^- $w^=$ w^+ w_0^{++} w_1^{++}
x_1^{--} x_0^{--} x^- $x^=$ x^+ x_0^{++} x_1^{++}
y_1^{--} y_0^{--} y^- $y^=$ y^+ y_0^{++} y_1^{++}
z_1^{--} z_0^{--} z^- $z^=$ z^+ z_0^{++} z_1^{++}
0
1
2
3
4
5
6
7
8

FIG. 1: The constructed linkstream L when $\varphi = (\overline{w} \vee x \vee \overline{y}) \wedge (\overline{w} \vee \overline{x} \vee z)$ and $\gamma = 3$. Here $T = [\![0, 8]\!]$ and an edge $(t, \{u, v\})$ of L is depicted by a bold line, following the horizontal line corresponding to time t, going from the vertical line corresponding to u to the vertical line corresponding to v. For readability, the edges incident with c are not drown. Instead, we have circled the temporal vertices that are neighbors of c.

3 Approximation algorithm

In this section, we adopt the greedy approach in order to provide a 2−approximation algorithm for γ-MATCHING. Let $L = (T, V, E)$ be a link stream. Let $\mathcal{P}$ be the set of every γ-edges of L. Let $\preceq$ be an arbitrary total ordering on the elements of $\mathcal{P}$ such that for any two elements of $\mathcal{P}$, $\Lambda_1 = \Gamma_\gamma(t_1, u_1, v_1)$ and $\Lambda_2 = \Gamma_\gamma(t_2, u_2, v_2)$ such that $t_1 < t_2$, we have $\Lambda_1 \preceq \Lambda_2$.

We denote by $\mathcal{A}$ the following greedy algorithm. The algorithm starts with $\mathcal{M} = \emptyset$, $\mathcal{Q} = \mathcal{P}$, and a function $\rho : V \times T \to \{0, 1\}$ such that for each $(t, v) \in T \times V$, $\rho(t, v) = 0$. The purpose of ρ is to keep track of the temporal vertices that are incident to a γ-edge of $\mathcal{M}$. As long as $\mathcal{Q}$ is not empty, the algorithm selects Λ, the γ-edge of $\mathcal{Q}$ that is minimum for $\preceq$, and removes it from $\mathcal{Q}$. Let K be the set of the 2γ temporal vertices that are incident to Λ. If, for each $(t, v) \in K$, $\rho(t, v) = 0$, then the algorithm adds Λ to $\mathcal{M}$ and for each $(t, v) \in K$, it sets $\rho(t, v)$ to 1, otherwise it does nothing at this step. When $\mathcal{Q} = \emptyset$, the algorithm returns $\mathcal{M}$.

As $\mathcal{P}$ can be determined in a sorted way in time $\mathcal{O}(m)$, this algorithm runs in time $\mathcal{O}(n\tau + m)$, where $\tau = |T|$, $n = |V|$, $m = |E|$, and where γ is a constant hidden in the $\mathcal{O}$.

Given a γ-matching $\mathcal{M}$, we define the *bottom temporal vertices* of $\mathcal{M}$, denoted $\texttt{bot}(\mathcal{M})$, as the set $\{(t + \gamma - 1, u), (t + \gamma - 1, v) \mid \Gamma_\gamma(t, u, v) \in \mathcal{M}\}$. Lemma 1 shows the crucial role of the bottom temporal vertices of the matching returned by $\mathcal{A}$. We omit the proof of Lemma 1 because of the space restriction.

Lemma 1 *Let γ be a positive integer, let L be a link stream, and let $\mathcal{M}$ be a γ-matching returned by $\mathcal{A}$ when applied to L. If $\mathcal{M}'$ is a γ-matching of L, then every γ-edge of $\mathcal{M}'$ is incident to, at least, one temporal vertex of* $\texttt{bot}(\mathcal{M})$.

Lemma 1 plays a cornerstone role in the proof of subsequent Theorem 2. As a byproduct, we also obtain the following result.

Theorem 1 *$\mathcal{A}$ is a 2-approximation of the γ-MATCHING problem.*

4 Kernelization algorithm

We now show a kernelization algorithm for γ-MATCHING by a direct pruning process based on Lemma 1. The main idea is as follows. First, we compute the set S of all bottom temporal vertices of a γ-matching produced by previously defined algorithm $\mathcal{A}$. Then, we prune the original instance by only keeping edges that belong to a γ-edge incident to a temporal vertex of S. More precisely, we prove the following result.

Theorem 2 *There exists a polynomial-time algorithm that for each instance (L, k), either correctly determines if L contains a γ-matching of size k, or returns an equivalence instance (L', k) such that the number of edges of L' is $2(k-1)(2k-1)\gamma^2$.*

Proof : Let $L = (T, V, E)$ be a link stream and k be an integer. We first run the algorithm $\mathcal{A}$ on L. Let $\mathcal{M}$ be the γ-matching outputed by the algorithm and let $\ell = |\mathcal{M}|$. If $\ell \geq k$, then we already have a solution and then we return a true instance. If $\ell < \frac{k}{2}$, then, by Theorem 1, we know that the instance does not contains a γ-matching of size k, and then we return a false instance. We now assume that $\frac{k}{2} \leq \ell < k$.

Lemma 1 justifies that we are now focusing on the temporal vertices of $\texttt{bot}(\mathcal{M})$ in order to find the requested kernel. We construct a set $\mathcal{P}$ of γ-edges and we show that any edge e, that is not in a γ-edge of $\mathcal{P}$, is useless when looking for a γ-matching of size k. For each $(t, u) \in \texttt{bot}(\mathcal{M})$, and for each t' such that $\max(0, t-\gamma+1) \leq t' \leq t$, we consider the set $\mathcal{S}(t', u)$ of every γ-edge, existing in L, with the form $\Gamma_\gamma(t', u, v)$ with $v \in V$. If the set $\mathcal{S}(t', u)$ is of size at most $2k-1$, we add every element of $\mathcal{S}(t', u)$ to $\mathcal{P}$. Otherwise, we select $2k-1$ elements of $\mathcal{S}(t', u)$ that we add them to $\mathcal{P}$. In both cases, we denote by $\mathcal{S}'(t', u)$ the set of elements of $\mathcal{S}(t', u)$ that we have added to $\mathcal{P}$. This finish the construction of $\mathcal{P}$. As $|\texttt{bot}(\mathcal{M})| = 2\ell$ and for each element of $\texttt{bot}(\mathcal{M})$ we have added at most $(2k-1)\gamma$ γ-edges to $\mathcal{P}$, we have that $|\mathcal{P}| \leq 2\ell(2k-1)\gamma \leq 2(k-1)(2k-1)\gamma$.

We now prove that if L contains a γ-matching $\mathcal{M}'$ of size k, then it also contains a γ-matching $\mathcal{M}''$ of size k such that $\mathcal{M}'' \subseteq \mathcal{P}$. Let $\mathcal{M}'$ be a γ-matching of L of size k such that $p = |\mathcal{M}' \setminus \mathcal{P}|$ is minimum. We have to prove that $p = 0$. Assume that $p \geq 1$. Let Λ be a γ-edge in $\mathcal{M}' \setminus \mathcal{P}$. Let (t, u) be a temporal vertex of $\texttt{bot}(\mathcal{M})$ that is incident to Λ. We know by Lemma 1 that this temporal vertex exists. Assume that $\Lambda = \Gamma_\gamma(t', u, v)$ for some $v \in V$ and some t' such that $\max(0, t-\gamma+1) \leq t' \leq t$. As $\Lambda \notin \mathcal{P}$, we have that $\Lambda \in \mathcal{S}(t', u) \setminus \mathcal{S}'(t', u)$, and so $|\mathcal{S}'(t', u)| = 2k-1$. Let $N_{\mathcal{S}'}(t', u)$ be the set of vertices w of $V \setminus \{u\}$ such that a γ-edge of $\mathcal{S}'(t', u)$ is incident to w. As $\mathcal{M}' \setminus \{\Lambda\}$ is of size $k-1$, the γ-edges that it contains can be incident to at most $2k-2$ vertices. This means that there exists $w \in N_{\mathcal{S}'}(t', u)$ such that no γ-edge of $\mathcal{M}' \setminus \{\Lambda\}$ is incident to w. Thus $(\mathcal{M}' \setminus \{\Lambda\}) \cup \{\Gamma_\gamma(t', u, w)\}$ is a γ-matching of size k. As $\Lambda \notin \mathcal{P}$ and $\Gamma_\gamma(t', u, v) \in \mathcal{P}$, this contradicts the fact that p is minimum.

We now can define the link stream $L' = (T, V, E')$ such that $E' = \{e \in E \mid \exists \Lambda \in \mathcal{P} : e \in \Lambda\}$. As $|\mathcal{P}| \leq 2(k-1)(2k-1)\gamma$ and every element of $\mathcal{P}$ is a γ-edge, we have that $|E'| \leq 2(k-1)(2k-1)\gamma^2$. The theorem follows. □

5 Conclusion and perspectives

We introduce the notion of a temporal matching in a link stream. Unexpectedly, the problem of computing a temporal matching, called γ-MATCHING, turns out to be NP-hard. We then show a kernelization algorithm for γ-MATCHING parameterized by the size of the solution. Our process produces quadratic kernels. On the way to obtaining the kernelization algorithm, we also provide a 2-approximation algorithm for γ-MATCHING. We believe that the same techniques extend to a large class of hitting set problems in the link streams.

References

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

[2] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[3] M. Latapy, T. Viard, and C. Magnien. Stream graphs and link streams for the modeling of interactions over time. 2017. `https://arxiv.org/abs/1710.04073`.

[4] T. Viard, M. Latapy, and C. Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.

A Modular Overlapping Community Detection Algorithm: Investigating the “From Local to Global” Approach

Maximilien Danisch[1], Noé Gaumont[2], Jean-Loup Guillaume[3]

[1] Sorbonne Université, CNRS, Laboratoire d’Informatique de Paris 6, LIP6, Paris F-75005, France
[2] Complex Systems Institute of Paris Ile-de-France (ISC-PIF), Paris, France, CAMS, CNRS - EHESS, Paris, France
[3] Laboratoire Informatique, Image et Interaction (L3i), Université de La Rochelle, La Rochelle, France

Abstract

We propose an overlapping community detection algorithm following a “from local to global approach”: our algorithm finds local communities one by one by repetitively optimizing a quality function that measures the quality of a community. Then, as some extracted local communities can be very similar to each-other, a cleaning procedure is applied to obtain the global overlapping community structure. Our algorithm depends on three modules: (i) a quality function, (ii) an optimization heuristic and (iii) a cleaning procedure. Various such modules can be independently plugged in. We show that, using default modules, our algorithm improves over a state-of-the-art method on some real-world graphs with ground truth communities. In the future we would like to study which combination of modules performs best in practice and make our code parallel.

1 Introduction

The Web graph (web pages connected by hyperlinks), Facebook (profiles connected by friendships), Internet (computers connected by Internet connections) and a human brain (neurons connected by synapses) are only a few examples of graphs extracted from the real world.

Designing practical algorithms to find relevant groups of nodes in such graphs has applications ranging from web search to drug design. However, designing such *community detection* algorithms is an extremely challenging task, indeed most real-world graphs are huge making any quadratic time algorithm not practical. In addition, community detection is an ill-defined problem, indeed there is no clear definition of what is a community, i.e. a relevant set of nodes.

In this paper we propose a generic and modular algorithm, called MOCDA for Modular Overlapping Community Detection Algorithm, that allows to compute a set of overlapping communities. This algorithm is based on a "local to global" approach (seed-centric approach) [4] where local communities are expanded around seeds by the repeated addition of nodes. A function is used to assess the quality of each community and an optimisation heuristic is used to optimize it. As two local communities could differ by only a small number of nodes, all the local communities are cleaned to remove the noise caused by similar communities and provide the global overlapping structure of the network. All steps are modular: the quality function, the optimization itself and the cleaning procedures can be modified to fit the needs of the user.

The rest of the paper is organized as follows: in Section 2 we present the algorithm and the different modules, in Section 3 we benchmark our algorithm against existing state-of-the-art method and we conclude in Section 4.

2 Algorithm

We detail here our modular algorithm for detecting overlapping communities that relies on three modules: (i) the definition of a function that evaluates the quality of a community, (ii) an optimization heuristic and (iii) a cleaning procedure. An efficient C implementation is available at `https://github.com/maxdan94/mocda` in which options are available to chose different strategies for the three modules.

2.1 Quality functions

A quality function is a function that evaluates if a given set of nodes is a good community or not. As there are several definitions of a *good* community, there are several ways to evaluate a set of nodes, such as the clustering coefficient or the conductance. We narrow the set of possible quality functions in MOCDA to those relying on local features to evaluate a given set of nodes or very simple global fatures. Indeed, a quality function actually does not need a complete knowledge of the graph to evaluate to what extent a set of nodes is a good community.

Formally, we consider quality functions that can be expressed as a function $f(\phi)$ where ϕ is a set of features among the following:

- n: number of nodes in the graph.
- m: number of links in the graph.
- t: number of triangles in the graph.
- s: number of nodes in the community.
- l_2: number of links with both end nodes in the community.
- l_1: number of links with exactly one node in the community.
- t_3: number of triangles with three nodes in the community.
- t_2: number of triangles with exactly two nodes in the community.
- t_1: number of triangles with exactly one node in the community.

Many quality functions of the literature can be written under that form such as conductance $\phi = \frac{l_1}{\min(2 \cdot l_2 + l_1, 2 \cdot m - 2 \cdot l_2 - l_1)}$ or cohesion $C = \frac{t_3}{\binom{s}{3}} \times \frac{t_3}{t_3 + t_2}$ [3].

The reason we bound our algorithm to these features is practical: we will optimize the input function in a greedy way by adding or removing nodes one at a time starting from a small set of nodes, we thus need to be able to evaluate the increase or decrease of the function extremely quickly and these parameters allows to do it. In particular, for the features related to triangles, we rely on the compact-forward algorithm detailed in [5] which allows to list all triangles in very-large real world graphs.

2.2 Optimization heuristics

We focus only on greedy and stochastic approaches. Given a node of interest u and a quality function f, a possible optimization heuristic consists in the following three steps that can be changed independently in our program.

- **Initialization:** start from a community containing only one node, or two linked nodes, or a node and all its neighbors.
- **Optimization:** at each iteration, add a randomly chosen node, neighbor of the community C, that increases the quality f. It is also possible to add the node that leads to the highest increase and/or to authorize the removal of a node.
- **Stop:** stop when the quality function can no longer be increased, i.e., when a local maximum has been reached. It is also possible to add the least quality-decreasing node with the hope that it will improve even more afterwards and return the set of nodes of highest quality obtained.

Therefore, given a quality function and a seed community, the optimization grows this seed to a full community. Since the optimization step is stochastic, two execution starting from the same seed community can give different results.

2.3 Cleaning procedures

Since the optimization is to be repeated several times using different seed communities and as two different optimizations can lead to similar final communities, it is important to clean these obtained sets of nodes. There are several ways to do that, which all require to compute the similarity between two sets of nodes. For this we use the F_1 similarity: given two sets a and b the similarity is given by $F_1(a,b) = 2\frac{|a \cap b|}{|a|+|b|}$. We say that two sets are similar if their similarity is higher than a given threshold. We use the following cleaning procedures depending on the order the communities are examined: (i) process the obtained sets of nodes on-the-fly or (ii) process the obtained sets of nodes in decreasing order of quality. In both cases a new set of nodes is kept if and only if it is not similar to a previously processed set of nodes.

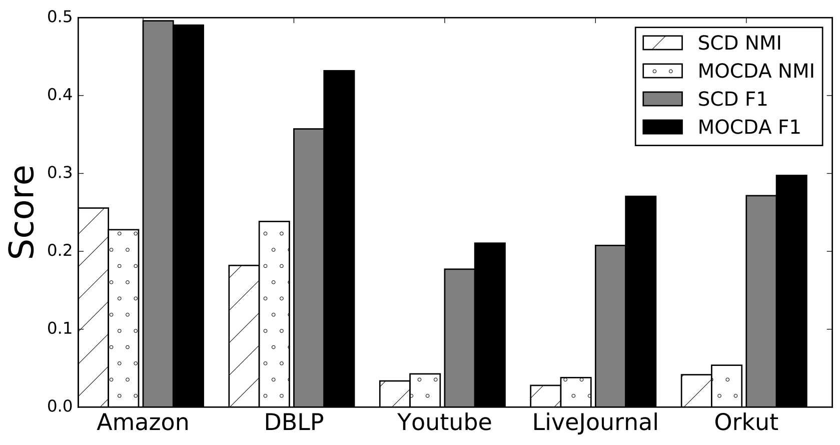


FIG. 1: $\mathcal{F}_1^{avr}$ and NMI scores using ground truth

Note that other cleaning procedures could also be investigated such as doing the intersection or union of found communities in some way, rather than simply removing sets.

Once the communities are cleaned, the algorithm only outputs the communities that were found to be similar to another one k or more times. This redundancy test states that a community found only once may be less relevant than a community found a large number of times.

Implementation details. Several solutions had to be found in order to obtain a program efficient in terms of time and memory. We were able to check whether a quality function of the type $f(n, m, t, s, l_1, l_2)$ (resp. $f(n, m, t, s, l_1, l_2, t_1, t_2, t_3)$) increases or decreases under the addition or deletion of a node u in constant time and proceed to the actual addition or deletion in $O(d_u)$ time (resp. $O(\sum_{v \in N(u)} d_v)$) in case f indeed increases while using linear memory. For the cleaning procedure, we were able to compute the maximum F_1-score between the new community and an already found one in time $O(s \cdot t)$ in the worst case, where s is the size of the found community and t is the maximum number of communities a node belongs to. Note that s and t are, in practice, very small compared to the size of the graph leading to a nearly linear cleaning procedure in terms of the number of communities.

3 Experimental evaluation

A wide range of methods have been designed to detect communities in graphs and we refer to the following reviews of existing methods [1, 9, 4, 2]. We compared our algorithm using default options to several algorithms including the Louvain method, BIGCLAM and OSLOM but we report the results only for SCD [8] as it performed better and faster than the other methods. SCD has been shown to lead to a non-overlapping and non-exhaustive community structure that is more similar to the real overlapping community structure of some real-world networks than the overlapping community structure unfolded by some state-of-the-art algorithms.

We tried 16 quality functions such as conductance, average degree, edit distance to an isolated clique and cohesion among others. The quality function leading to the best results on all networks was $\frac{l_2}{n^{1.5}}$. We also tried several optimization heuristics and cleaning procedures and the best trade-off between time and accuracy is obtained by a stochastic optimization allowing removal of nodes (we do $2 \cdot n$ optimizations, each time starting from a randomly chosen node) and on-the-fly cleaning (simply outputting all unique communities found at least twice).

To test our algorithm we applied our method on networks with a known community structure [6]. In order to compare the ground truth and the structure found by MOCDA and SCD, we used the Normalized Mutual Information (NMI) [7] and the $\mathcal{F}_1^{avr}$ [8]. Figure 1 shows the NMI and the $\mathcal{F}_1^{avr}$ [8]. The larger are these metrics the more similar are the two community structures. Table 1 shows the running time on a laptop with a CPU Intel i7-6500U at 2.50GHz and 16Go of RAM. As we can see, even though our algorithm is slower than SCD, it outperforms it on four of the five datsests. Note that SCD has been shown to be a very competitive method, our algorithm relying on these three simple modules is thus a very promising tool.

Algorithm	Amazon	DBLP	Youtube	LiveJ.	Orkut
SCD	4.5 sec	4 sec	23 sec	8.5 min	41 min
MOCDA	4.9 sec	10.2 sec	19.1 min	83.1 min	37.9 h

TAB. 1: Running time comparison

4 Conclusion

We proposed a generic and modular algorithm to extract overlapping communities in large networks using a local-to-global approach. This algorithm, using default options, gives better results in most real-world graphs than a state-of-the-art algorithm, even though it is slower.

For future work, we first plan to parallelize the algorithm since the optimization procedure can be performed independently from several seed communities. Furthermore, we want to add more complex features, such as cliques of size more than 3 or others motifs that can be computed efficiently on large networks. We also plan to study in depth the impact of these parameters on the obtained results.

Learning which combinations of modules (quality function, optimization heuristic and cleaning procedure) perform best in an automatic way is also an interesting research perspective.

More important still, the modular design of MOCDA enables anybody to create and test its own quality function which will meet its needs.

References

[1] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[2] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 2016.

[3] Adrien Friggeri, Guillaume Chelius, and Eric Fleury. Triangles to capture social cohesion. In *SocialCom*, pages 258–265. IEEE, 2011.

[4] Rushed Kanawati. Seed-centric approaches for community detection in complex networks. In *International Conference on Social Computing and Social Media*, pages 197–208. Springer, 2014.

[5] Matthieu Latapy. Main-memory triangle computations for very large (sparse (power-law)) graphs. *Theoretical Computer Science*, 407(1):458–473, 2008.

[6] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[7] Aaron F McDaid, Derek Greene, and Neil Hurley. Normalized mutual information to evaluate overlapping community finding algorithms. *arXiv preprint*, 2011.

[8] Arnau Prat-Pérez, David Dominguez-Sal, and Josep-LLuis Larriba-Pey. High quality, scalable and parallel community detection for large real graphs. In *WWW*, pages 225–236. ACM, 2014.

[9] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *Acm computing surveys (csur)*, 45(4):43, 2013.

Total k-rainbow Domatic Number

Pavitra Kumbargoudra[1], S. S. Shirkol[2]

[1]The Oxford College of Engineering, Bangalore, India
pavitra2504@gmail.com
[2]SDM College of Engineering and Technology, Dharwad, India
shailajashirakol@gmail.com

Abstract

The Total k-rainbow domination number is defined by considering k types of guards or set of k colors. A location which does not have any type of guard (color) assigned, should have all type of guards (colors) in its immediate surrounding location to protect it. For a positive integer k, a function $f : V(G) \to P(\{1, 2, \dots, k\})$ is said to be a total k-rainbow dominating function if $\forall v \in V(G), \cup_{u \in N[v]} f(u) = \{1, 2, \dots, k\}$, where $f(u)$ is nonempty subset of $\{1, 2, \dots, k\}$ and $N[v]$ is a closed neighborhood of v. A set $\{f_1, f_2, \dots, f_d\}$ of total k-rainbow dominating function of a graph G with the property that $\sum_{i=1}^{d} |f_i(v)| \leq k$ for each $v \in V(G)$, is called a total k-rainbow dominating family (functions) on G. The maximum number of functions in a total k-rainbow dominating family (TkRD family) on G is called the total k-rainbow domatic number of G, is denoted by $d_{trk}(G)$. In this paper we initiate the study of total k-rainbow domatic number in graphs and we obtain $d_{trk}(K_n) = min\{n, k\}$, $d_{trk}(C_n) \leq 3$. We also proved some bounds for $d_{trk}(G)$.

Keywords : *k-rainbow domination number, Total k-rainbow domination number, k-rainbow domatic number, Total k-rainbow domatic number.*

1 Introduction

Let G be a simple graph with vertex set $V = V(G)$ and the edge set $E = E(G)$. The number of vertices in the graph G is known as order of G and is denoted by $n = n(G)$. For any vertex $v \in V(G)$, the open neighborhood

$N(v)$ is the set $\{u \in V(G)|uv \in E(G)\}$ and the closed neighborhood of v is the set $N[v] = N(v) \cup \{v\}$. The degree of a vertex $v(G)$, $d(v)$, is the number of edges incident on the vertex v . The minimum and maximum degree of a graph G are denoted by $\delta = \delta(G)$ and $\Delta = \Delta(G)$ respectively. Tree T is a connected acyclic graph. We write K_n for complete graph of order n, C_n for a cycle of length n, P_n for path of length n and W_n for wheel graph of order n. We follow [1] and [2] for notation and graph theory terminology.

M. A. Henning [2] introduced the concept of k-rainbow domination number by considering mathematical model of assigning guards to each location from k different type of guards. According to him a location which is not having any type of guards needs to have all type of guards in its neighboring location.

Definition 1.1. Let G be a graph and f be a function that assigns to each vertex a set of guards chosen from the set $\{1, 2, \ldots, k\}$ i.e. $f : V(G) \to P(\{1, 2, \ldots, k\})$. If for each vertex $v \in V(G)$ such that $f(V) = \emptyset$ we have

$$\cup_{u \in N(V)} f(u) = \{1, 2, \ldots, k\},$$

then f is called the k-rainbow dominating function(kRDF) [4]. The weight $w(f)$ of k-rainbow domination number is defined by, $w(f) = \sum_{v \in V} |f(v)|$. The minimum weight a kRDF is called k-rainbow domination number and is denoted by $\gamma_{rk}(G)$.

The study of total k-rainbow domination number is introduced by P. Kumbargoudra and J. V. Kureethara. According to total k-rainbow dominating function each and every location is secured by all type of guards. A location which is not having any type of guards should have that type of guard in its immediate neighboring location.

Definition 1.2. For a positive integer k, a total k-rainbow dominating function (TkRDF) of a graph G is defined in [3] as a function $f : V(G) \to P(\{1, 2, \ldots, k\})$ that assigns to each vertex a nonempty subset of a set $S = \{1, 2, \ldots, k\}$ i.e.

$$\forall v \in V(G),\ \cup_{u \in N[v]} f(u) = S.$$

The weight $w(f)$ of total k-rainbow domination number is defined by,

$$w(f) = \sum_{v \in V} |f(v)|.$$

The minimum weight of a TkRDF is known as total k-rainbow domination number and is denoted by $\gamma_{trk}(G)$.

Later S. M. Sheikholeslami and L. Volkmann [9] defined k-rainbow domatic number $d_{rk}(G)$. They found 2-rainbow domatic number for $P_n,\ C_n$ and they found some bounds for 2-rainbow domatic number.

In this paper we introduce total k-rainbow damatic number and initiated the study of the total k-rainbow domatic number of some classes of graphs. We obtain basic bounds for the total k-rainbow domatic number of a graph.

Definition 1.3. A set $\{f_1, f_2, \ldots, f_d\}$ of total k-rainbow dominating functions of a graph G with the property that $\sum_{i=1}^{d} |f_i(v)| \leq k$ for each $v \in V(G)$, is called a total k-rainbow dominating family (functions) on G. The maximum number of functions in a total k-rainbow dominating family (TkRD family) on G is called the total k-rainbow domatic number of G, is denoted by $d_{trk}(G)$.

The total k-rainbow domatic number is well defined and

$$d_{trk} \geq 1, for\ all\ graph\ G, \tag{1}$$

since the set consisting of any total k-rainbow dominating function (TkRDF) forms a TkRD family on G.

2 Properties of Total k-rainbow Domatic Number

Proposition 2.1. If K_n is a complete graph of order $n \geq 3$, then $d_{trk}(K_n) = min\{n, k\}$.

Proposition 2.2. If W_n is a Wheel graph of order n then, $d_{trk}(W_n) = 3$.

Proposition 2.3. If T is a tree then, $d_{trk}(T) = 2$.

Theorem 2.4. If G is graph, then $d_{trk}(G) = 1$ if and only if G is empty.

Theorem 2.5. For any graph G, $2 \leq d_{trk} \leq k$.

Theorem 2.6. Let G be a graph with $\delta = 1$ then, $d_{trk}(G) = 2$.

Theorem 2.7. If G is graph of order n, then $\gamma_{trk}(G) \cdot d_{trk}(G) \leq kn$.
If $\gamma_{trk}(G) \cdot d_{trk}(G) = kn$, then for each TkRD family $\{f_1, f_2, \ldots, f_d\}$ on G with $d = d_{trk}(G)$, each function f_i is a $\gamma_{trk}(G)-$function, and $\sum_{i=1}^{d} |f_i(v)| = k$ for all $v \in V$.

Theorem 2.8.

$$For \quad n > 2, \quad d_{trk}(C_n) = \begin{cases} 3 & If\ n \equiv 0\ (mod\ 3) \\ 2 & Otherwise. \end{cases}$$

3 Conclusion and perspectives

We conclude this paper with the following open problems.

Open Problem 3.1. Caclculate $d_{trk}(P_n \Box P_m)$, $d_{trk}(C_n \Box C_m)$ and $d_{trk}(P_n \Box C_m)$.

Open Problem 3.2. Calculate $d_{trk}(P(n,k))$ where $P(n,k)$ is generalized Peterson graph.

Open Problem 3.3. If G be any graph with given δ and Δ then $d_{trk}(G) =?$

Open Problem 3.4. For which classes of graphs is $d_{rk}(G) = d_{trk}(G)$ for every graph G of a class?

References

[1] M. A. Henning and S. T. Hedetniemi, "Defending the Roman Empire—A new strategy," in *Discrete Applied Mathematics*, vol. 266, pp. 239-251, 2003.

[2] M. A. Henning, "Defending the Roman Empire from multiple attacks," in *Discrete Applied Mathematics*, vol. 271, pp. 101-115, 2003.

[3] P. Kumbargoudra and J. V. Kureethara, "Total k-rainbow Domination in Graphs," in *IJCIET*, 8, pp. 867-875, 2017.

[4] B. Bresar, M. A. Henning and D. F. Rall, "Rainbow domination in graphs," *Taiwanse Journal of Mathematics*, vol. 12, pp. 213-225, 2008.

[5] L. Volkmann and B. Zelinka, "Signed domaic number of graphs", Discrete Mathematics 150 (2005), 261-267.

[6] D. Meierling, L. Volkmann and S. Zitzen "The signed domatic number of some regular graphs", Discrete Mathematics 157 (2009), 1905-1912.

[7] M. Atapour, S. M. Sheikholeslami, A. N. Ghameshloub and L. Volkmannc, "Signed star domatic number of a graph", Discrete Mathematics 158 (2010), 213-218.

[8] S. M. Sheikholeslami and L. Volkmann, "The Roman domatic number of a graph", Applid Mathematics letters 23 (2010), 1295-1300.

[9] S. M. Sheikholeslami and L. Volkmann, "THE k-RAINBOW DOMATIC NUMBER OF A GRAPH", Discussiones Mathematicae Graph Theory 32 (2012), 129-140.

Sufficient degree conditions for traceability of claw-free graphs

Tao Tian[1,2], Hajo Broersma[2], Liming Xiong[3]

[1] School of Mathematics and Statistics, Beijing Institute of Technology, Beijing, PR China
`taotian0118@163.com`

[2] Faculty of EEMCS, University of Twente, Enschede, The Netherlands
`h.j.broersma@utwente.nl`

[3] School of Mathematics and Statistics, Beijing Key Laboratory on MCAACI, Beijing Institute of Technology, Beijing, PR China
`lmxiong@bit.edu.cn`

Abstract

We present new results on the traceability of claw-free graphs. In particular, we consider sufficient minimum degree and degree sum conditions that imply that these graphs admit a Hamilton path, unless they have a small order or they belong to well-defined classes of exceptional graphs. Our main result implies that a 2-connected claw-free graph G of sufficiently large order n with minimum degree $\delta(G) \geq 22$ is traceable if the degree sum of any set of t independent vertices of G is at least $\frac{t(2n-5)}{14}$, where $t \in \{1, 2, \ldots, 7\}$, unless G belongs to one of a number of well-defined classes of exceptional graphs depending on t. Our results also imply that a 2-connected claw-free graph G of sufficiently large order n with $\delta(G) \geq 18$ is traceable if the degree sum of any set of six independent vertices is larger than $n - 6$, and that this lower bound on the degree sums is sharp.

Keywords : *Claw-free graph, traceable graph, line graph, spanning trail, closure.*

1 Introduction

In this talk, we are mainly interested in degree and neighborhood conditions for traceability of 2-connected claw-free graphs, motivated by recent results on counterparts for hamiltonicity, and in an attempt to unify several existing results.

We consider finite, undirected and loopless graphs only, but we allow multiple edges. A graph is called a *multigraph* if it may contain multiple edges; otherwise, it is called a *simple graph* or simply a graph. For a vertex x of a (simple) graph G, we denote by $N_G(x)$ the *neighborhood* of x in G, i.e., the set of vertices adjacent to x in G, and by $d_G(x) = |N_G(x)|$ (or simply $d(x)$) the *degree* of x in G. We let $\overline{\sigma}_2(G) = \min\{d(u) + d(v) \mid uv \in E(G)\}$. The *circumference* of G, denoted by $c(G)$, is the length of a longest cycle of G. A graph G is *hamiltonian* (*traceable*) if it has a *Hamilton cycle (path)*, i.e., a spanning cycle (path). A graph is *claw-free* if it has no induced subgraph isomorphic to $K_{1,3}$. A graph is *triangle-free* if it contains no cycle with exactly 3 vertices. The *independence number* of a graph G is denoted by $\alpha(G)$.

An edge-cut X of G is called *essential* if $G - X$ has at least two non-trivial components, i.e., components that contain at least one edge. For an integer $k \geq 1$, a graph G is said to be *essentially k-edge-connected* if G does not admit an essential edge-cut X with $|X| < k$. The *line graph* of a graph G, denoted by $L(G)$, has $E(G)$ as its vertex set, and two vertices in $L(G)$ are adjacent if and only if the corresponding edges in G have a vertex in common. Note that a graph G is essentially k-edge-connected if and only if $L(G)$ is k-connected (or complete).

In the context of investigating the hamiltonicity or traceability of claw-free graphs, Ryjáček [5] defined the closure $cl(H)$ of a claw-free graph H, obtained by recursively adding edges to

H that join two nonadjacent vertices in the neighborhood of any locally connected vertex of the current graph as long as this is possible. A graph H is said to be closed if $H = cl(H)$. The following theorem summarizes the basic properties of $cl(H)$.

Theorem 1 ([5]). *Let H be a claw-free graph. Then*

(i) $cl(H)$ is well-defined;

(ii) there is a triangle-free graph G such that $cl(H) = L(G)$;

(iii) H and $cl(H)$ have the same circumference.

Later, this result was extended to an analogous statement for traceability of claw-free graphs.

Theorem 2 ([1]). *A claw-free graph H is traceable if and only if $cl(H)$ is traceable.*

The above results have been the key ingredient for proving many results on hamiltonicity and traceability of claw-free graphs, in combination with the concepts defined in the next section.

2 Trails, contractions and neighborhood conditions

Another key ingredient is the following equivalence between dominating (closed) trails in graphs and traceability (hamiltonicity) of their line graphs. We only state the result on traceability. Here, a (closed) trail Ψ of G is called a *dominating (closed) trail* (DT and DCT for short) of G if $E(G - V(\Psi)) = \emptyset$.

Theorem 3 ([4]). *Let G be a graph with $|E(G)| \geq 1$. Then the line graph $L(G)$ of G is traceable if and only if G has a DT.*

Let G be a connected multigraph. For $X \subseteq E(G)$, the *contraction* G/X is the graph obtained from G by successively identifying the two end vertices of each edge $e \in X$ and deleting the resulting loops. Note that, in general G/X is a multigraph, also in case G is a simple graph. If Γ is a connected sub(multi)graph of G, then we write G/Γ for $G/E(\Gamma)$; in this case, we use v_Γ to denote the only remaining vertex of Γ in G/Γ, i.e., the vertex in G/Γ to which Γ is contracted, and we call this vertex v_Γ a *contracted vertex* if $\Gamma \neq K_1$ in order to distinguish it from the remaining vertices of G.

Let t be a positive integer and let H be a graph. We define $\sigma_t(H)$ and $U_t(H)$ as follows. If $t \leq \alpha(H)$, then:

- $\sigma_t(H) = \min\{\sum_{i=1}^{t} d_H(v_i) : \{v_1, v_2, \ldots, v_t\}$ is an independent set of $H\}$;
- $U_t(H) = \min\{|\bigcup_{i=1}^{t} N_H(v_i)| : \{v_1, v_2, \ldots, v_t\}$ is an independent set of $H\}$.

If $t > \alpha(H)$, we set $\sigma_t(H) = U_t(H) = \infty$. Obviously, $\delta(H) = \sigma_1(H) = U_1(H)$, and $\sigma_t(H) \geq U_t(H)$. We let $\Omega(H) = \{\sigma_t(H), U_t(H)\}$.

For $d_t(H) \in \Omega(H)$, we consider claw-free graphs H that satisfy the following condition:

$$d_t(H) \geq \frac{t(n+\epsilon)}{p}. \tag{1}$$

Here $t \geq 1$ and $p \geq t$ are positive integers, and ϵ is a given real number. Depending on the values of p and ϵ, we define $N(p,\epsilon) = \max\{36p^2 - 34p - \epsilon(p+1), 20p^2 - 10p - \epsilon(p+1), (3p+1)(-\epsilon - 4p)\}$. In the next two results, we let H be a k-connected claw-free graph of order $n > N(p,\epsilon)$ with $k \in \{2,3\}$.

3 Main results

In [2], Chen proved the following hamiltonicity result.

Theorem 4 ([2]). *If $\delta(H) \geq 3$ and $d_t(H) \geq \frac{t(n+\epsilon)}{p}$, then either H is hamiltonian or $cl(H) = L(G)$, where G is an essentially k-edge-connected triangle-free graph without a DCT and G satisfies one of the following:*

(i) $k = 2$ and G is contractible to a graph in $Q_0(c, 2)$, where $c \leq$ max $\{4p - 5, 2p + 1\}$;

(ii) $k = 3$ and G is contractible to a graph in $Q_0(c, 3)$, where $c \leq$ max $\{3p - 5, 2p + 1\}$.

Here $Q_0(c, k)$ denotes a well characterized class of graphs that we do not specify here due to the page limit. We recently obtained the following traceability counterpart of Theorem 4.

Theorem 5 *If $\delta(H) \geq 3$ and $d_t(H) \geq \frac{t(n+\epsilon)}{p}$, then either H is traceable or $cl(H) = L(G)$, where G is an essentially k-edge-connected triangle-free graph without a DT and G satisfies one of the following:*

(i) $k = 2$ and G is contractible to a graph in $R_0(c, 2)$, where $c \leq$ max$\{4p - 5, 2p + 1\}$ and $p \geq 4$;

(ii) $k = 3$ and G is contractible to a graph in $R_0(c, 3)$, where $c \leq$ max$\{3p - 5, 2p + 1\}$ and $p \geq 7$.

Here $R_0(c, k)$ denotes a class similar to $Q_0(c, k)$. As an application of Theorem 4, the following special case was obtained in [2] as an illustration.

Theorem 6 ([2]). *Let H be a 2-connected claw-free graph of sufficiently large order n with $\delta(H) \geq 3$. If $d_t(H) \geq \frac{tn}{4}$ for $t \in \{1, 2, 3, 4\}$, then either H is hamiltonian or $cl(H)$ belongs to a family of well characterized graphs.*

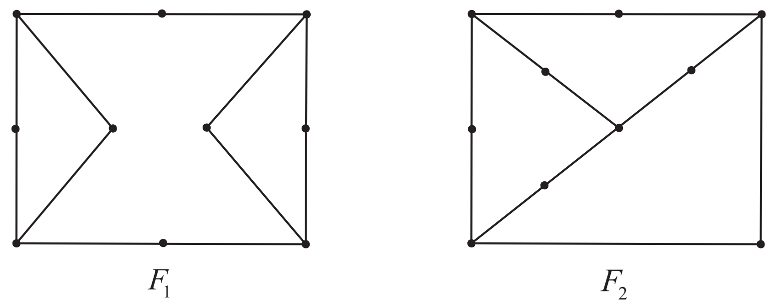


FIG. 1: The graphs F_1 and F_2.

For each $F \in \{F_1, F_2\}$, let $D_2(F) = \{v_1, v_2, \ldots, v_6\}$ denote the vertices with degree 2, and let $\mathcal{F}(n, s)$ be the family of essentially 2-edge-connected graphs obtained from F by replacing each $v_i \in D_2(F)$ by a connected triangle-free graph Φ_i of size $s_i \geq s$ such that $n = 12 + \sum_{i=1}^{6} s_i$. Let $\mathcal{F}_1(n, s)$ denote the subfamily of $\mathcal{F}(n, s)$ in which each Φ_i is isomorphic to K_{1,s_i}. Furthermore, let $\mathcal{R}_{\mathcal{F}}(n, s) = \{H = L(G) : G \in \mathcal{F}(n, s)\}$, and let $\mathcal{R}^1_{\mathcal{F}}(n, s) = \{H = L(G) : G \in \mathcal{F}_1(n, s)\}$.

These classes have been used by Tian et al. in [6] to characterize the exceptional cases for traceability of claw-free graphs satisfying the following degree sum condition in terms of $\overline{\sigma}_2$ for pairs of *adjacent* vertices.

Theorem 7 ([6]). *Let H be a 2-connected claw-free graph of sufficiently large order n with $\delta(H) \geq 3$. If $\overline{\sigma}_2(H) \geq \frac{2n-5}{7}$, then either H is traceable or $\overline{\sigma}_2(H) \leq \frac{n-6}{3}$ and $cl(H) \in \mathcal{R}_{\mathcal{F}}(n, \frac{2n-19}{14})$.*

As an application of Theorem 5, we obtain the following results.

Theorem 8 *Let H be a 2-connected claw-free graph of sufficiently large order n with $\delta(H) \geq 22$. If $d_t(H) \geq \frac{t(n-2.5)}{7}$ with $t \in \{1, 2, \ldots, 7\}$, then either H is traceable or $cl(H) \in \mathcal{R}_{\mathcal{F}}(n, 1)$.*

Theorem 9 *Let H be a 2-connected claw-free graph of sufficiently large order n with $\delta(H) \geq 22$. If $d_t(H) \geq \frac{t(n-2.5)}{7}$ with $t \in \{1, 2, \ldots, 7\}$, then either H is traceable or $cl(H) = L(G)$, where G is an essentially 2-edge-connected triangle-free graph that can be contracted to either F_1 or F_2, in such a way that all vertices of degree two are nontrivial.*

Theorem 10 *Let H be a 2-connected claw-free graph of sufficiently large order n with $\delta(H) \geq 18$. If $\sigma_6(H) \geq n-6$, then either H is traceable or $\sigma_6(H) = n-6$ and $cl(H) \in \mathcal{R}^1_{\mathcal{F}}(n, 1)$.*

We also obtained the following known result as a corollary.

Corollary 1 ([6]). *Let H be a 2-connected claw-free graph of sufficiently large order n. If $\delta(H) \geq \frac{n-6}{6}$, then either H is traceable or $\delta(H) = \frac{n-6}{6}$ and $cl(H) \in \mathcal{R}^1_{\mathcal{F}}(n, \frac{n-12}{6})$.*

The proofs of the above results all use the equivalences that are demonstrated in Theorems 1, 2 and 3. We aim to present sketches of some of these proofs in our talk. We will also discuss examples to show the asymptotically sharpness of the bound $\frac{n-2.5}{7}$ in Theorems 8 and 9.

Our results also extend earlier results that are based on the notion of the generalized t-degree, as introduced in [3] by Faudree et al. The generalized t-degree, $\delta_t(H)$, of a graph H is defined as $\delta_t(H) = \min\{|\bigcup_{i=1}^{t} N_H(v_i)| : \{v_1, v_2, \cdots, v_t\}$ is a t-subset in $H\}$. Since obviously $\sigma_t(H) \geq U_t(H) \geq \delta_t(H)$, the statements in Theorems 5, 8, 9 and 10 are also valid if we replace $d_t(H)$ by $\delta_t(H)$.

References

[1] S. Brandt, O. Favaron, and Z. Ryjáček. *Closure and stable hamiltonian properties in claw-free graphs.* J. Graph Theory, 34(1):30–41, 2000.

[2] Z.-H. Chen. *Degree and neighborhood conditions for hamiltonicity of claw-free graphs.* Discrete Math., 340(12):3104–3115, 2017.

[3] R. Faudree, R. Gould, L. Lesniak, and T. Lindquester. *Generalized degree conditions for graphs with bounded independence number.* J. Graph Theory, 19(2):397–409, 1995.

[4] D. Li, H. Lai, and M. Zhan. *Eulerian subgraphs and Hamilton-connected line graphs.* Discrete Appl. Math., 145(3):422–428, 2005.

[5] Z. Ryjáček. *On a closure concept in claw-free graphs.* J. Combin. Theory Ser. B, 70(2): 217–224, 1997.

[6] T. Tian, L. Xiong, Z.-H. Chen, and S. Wang. *Degree sums of adjacent vertices for traceability of claw-free graphs.* Submitted.

Star forest polytope on complete graphs

Lamia Aoudia[1], Zohra Aoudia[2], Viet Hung Nguyen[3], Méziane Aider[1]

[1] LaROMaD, Fac. Maths, USTHB, PB 32, 16111 Bab Ezzouar, Algeria
laoudia@usthb.dz ; m-aider@usthb.dz

[2] LaMOS, University Abderrahmane Mira, Béjaia, Algeria
zo.aoudia@gmail.com

[3] Sorbonne Universités, UPMC, Lip6, Paris, France
hung.nguyen@lip6.fr

Abstract

Given a complete graph, a star is a subgraph where one specific node is incident to all the edges of the subgraph. A star forest is a subgraph in G where each connected component is a star. The maximum weighted spanning star forest problem consists on finding a star forest with maximum total weight on its edges. It can be formulated as an integer linear program. We consider the convex hull of all its solutions, this defines a polyhedron with integer extreme points. The aim of the present work is to provide a description of this polyhedron by means of a system of linear inequalities.

Keywords : *Support graph, star forest, valid inequalities.*

1 Introduction

Let $K_n = (V_n, E_n)$ be a complete graph on n nodes without loops and multiple edges, i.e. every two different nodes K_n are linked by exactly one edge. A *star* in a G is a subgraph $S = (V_S, E_S)$ where there is one node which is either an isolated node or incident to every edge in E_S. This node is called the *center* of the star. A *star forest* is a spanning subgraph $F = (V_n, E_F)$ where each component is a star. *A maximum weighted spanning star forest problem* (for short $MWSFP$) is the task to find, given a complete graph K_n with edge weights $c_e \in \mathbb{R}$ for $e \in E_n$, a spanning star forest $F^* = (V_n, E_{F^*})$ such that $c(F^*) = \sum_{e \in E_F} c_e$ has maximum value.
The spanning star forest problem is a combinatorial optimization version of the clustering problem in data analysis with many interesting applications, among others, computational biology, automobile industry (see for instance [2] and [1]). This problem is $\mathcal{NP}$-hard [2]. In order to solve an instance coming up from practical applications, Nguyen *et al.* proposed in [2] a linear time algorithm when the graph is a tree, and a $\frac{1}{2}$-approximation algorithm in general case.
In what follows, we use the standard graph theory terminology.
A graph is denoted by $G = (V, E)$, where V is the node set and E the edge set of G. For the propblem under consideration, loops and multiple edges are irrelevant, so we assume throughout the paper, that all graphs considered are simple. If $H = (W, F)$ and $G = (V, E)$ are graphs with $W \subseteq V$ and $F \subseteq E$, then H is called a subgraph of G. We use the symbol V_n for the node set and the symbol E_n for the edge set of K_n. For $v \in V$, $G - v$ denotes the graph obtained from G by removing the vertex v. For $W \subseteq V$, $G[W]$ is the subgraph of G induced by W. It will be convenient to use the following notation, where $S, T, S_1, \ldots, S_k \subseteq V$ and $F \subseteq E$.
$E(S) = \{uv \in E | u, v \in S\}$,
$\delta(S) = \{uv \in E | u \in S, v \notin S\}$
$V(F) = \{v \in V | v \text{ is the end node of some edge in } F\}$.

A *cycle* C of length k is an edge set of the form $\{v_1v_2, v_2v_3, \ldots, v_{k-1}v_k, v_1v_k\}$, where $v_i \neq v_j$ if $i \neq j$. For $k \geq 4$, the set $\bar{C} = \{v_iv_{i+2} | i = 1, \ldots, k-2\} \cup \{v_1v_{k-1}, v_2v_k\}$ is called the set of 2-chords of C. A *triangle* is a cycle of length 3. A graph $G = (V, E)$ is *bipartite* if its node set can be partitioned into two nonempty subsets V_1, V_2 such that all edges of G have one end node in V_1 and the other in V_2. Every partition of V with this property is called a bipartition of V. If $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are two graphs then the graph $(V_1 \cup V_2, E_1 \cup E_2)$ is called the union of G_1 and G_2 and is denoted by $G_1 \cup G_2$. We assume that the union edges do not produce multiple edges, so $G_1 \cup G_2$ is a simple graph.

2 The star forest polytope

To formulate the maximum weighted spanning star forest problem ($MWSFP$) in polyhedral and linear programming terms, we consider the associated polyhedron.
Let $\mathbb{R}^{E_n}$ denote the real vector space where every component x_e of a vector $x \in \mathbb{R}^{E_n}$ is indexed by an edge e of the complete graph $K_n = (V_n, E_n)$. To avoid triviality, we assume throughout the paper that $n \geq 3$. For every edge set $F \subseteq E_n$, that is a star forest, $\mathcal{X}^F \in \mathcal{R}^{E_n}$ denotes the incidence vector of F in K_n. The convex hull of the set of these vectors is denoted by $\mathcal{P}_n$ i.e.,

$$\mathcal{P}_n = conv\{\mathcal{X}^F \in \mathbb{R}^{E_n} | F \text{ is a star forest of } K_n\}$$

Since the vertices of $\mathcal{P}_n$ are in one to one correspondence with the star forests of K_n, it follows immediately that the $MWSFP$ can be formulated as the problem:

$$\begin{aligned} \text{maximize} \quad & c^T x, \\ \text{subject to} \quad & x \in \mathcal{P}_n. \end{aligned}$$

This is a linear program in the sense that a linear objective function is to be minimized over a polytope. To apply LP-techniques, this formulation is of no-use unless $\mathcal{P}_n$ can be represented by a system of linear inequalities. Since the maximum weighted spanning star forest problem($MWSFP$) is $\mathcal{NP}$-hard, it follows from general results of complexity theory that this linear representation does not exist unless $\mathcal{P} = \mathcal{NP}$. Here, we determine a large class of valid and facet defining inequalities for $\mathcal{P}_n$.
Recall that an inequality $a^T x \leq \alpha$ is called *valid* for $\mathcal{P}_n$ if $\mathcal{P}_n \subseteq \{x \in \mathbb{R}^{E_n} | a^T x \leq \alpha\}$. A valid inequality $a^T x \leq \alpha$ is said to *define a facet* of $\mathcal{P}_n$ if the face $F_a = \{a^T x \leq \alpha\}$ of $\mathcal{P}_n$ is a facet, i.e., if F_a is a face of dimension one less than the dimension of $\mathcal{P}_n$(the dimension of a set S is the cardinality of the largest set of affinely independent points in S minus one). If S is a subset of E_n then we use the symbol $x(F)$ as a short hand notation for the sum $\sum_{e \in F} x(e)$. We have the following results :

Theorem 1 *$\mathcal{P}_n$ is a monotone full dimensional polytope.*

Theorem 2 *The trivial inequalities*

$$0 \leq x(e) \leq 1 \textit{ for all } e \in E$$

define facets of $\mathcal{P}_n$.

A star forest is a subgraph with no cycle of length > 2 and no path of length > 2. Let $\mathcal{P}_k$ (resp. $\mathcal{C}_k$) The collection of all paths of length $k-1$ (resp. of length k), then we have the following proposition:

Proposition 1 *The following inequalities*

$$x(P) \leq 2 \textit{ for all } P \in \mathcal{P}_4, \tag{1}$$
$$x(C) \leq 2 \textit{ for all } C \in \mathcal{C}_3, \tag{2}$$

are valide for $\mathcal{P}_n$.

Call inequality (1) 3-*path*-inequality and the inequality (2) a 3-*cycle inequality.* The following proposition summarizes a few structural properties of facet defining inequalities of $\mathcal{P}_n$:

Proposition 2 *Let $a^T x \leq \alpha$ be a nontrivial facet-defining inequality for $\mathcal{P}_n$ and let $E_a = \{e \in E_n | a_e \neq 0\}$ the following holds.*

(a) $\alpha \geq 0$,

(b) a has positive entries,

(c) The subgraph $H = (V_n(E_a), E_a)$ of K_n is connected.

Notice that the subgraph $H = (V_n(E_a), E_a)$ is called the support graph of the inequality $a^T x \leq \alpha$.

Theorem 3 • *Every 3-cycle inequality defines a facet of $\mathcal{P}_n$ and,*

- *No 3-path inequality define a facet of $\mathcal{P}_n$.*

Theorem 4 *Every 4-cycle inequality defines a facet of P_n.*

Let $H = (V_H, E_H)$ be a graph defined as 3-cycle plus one pendent edge. We call H a *paw.* Let us write $E_H = C_H \cup \{e_H\}$ and let M_H be a perfect matching on H. We define the paw inequality as follows:

$$x(C_H) + x(M_H) \leq 3. \tag{3}$$

Theorem 5 *The paw inequality (3) is valid and define a facet of $\mathcal{P}_n$.*

Now we would like to prove useful lifting theorem that shows that every inequality defining a facet $\mathcal{P}_k$ (and satisfies a condition) defines a facet of $\mathcal{P}_n$, $n > k$.

Theorem 6 *(Lifting theorem). Suppose $\sum_{e \in E_k} a_e x(e) \leq \alpha$ define a non trivial facet of $\mathcal{P}_k$, then this inequality also defines a facet of $\mathcal{P}_n$ for all $n > k$ provided the following condition is satisfied:*

(L) There exists a star forest F of K_k with a star centered at v such that $\sum_{e \in E_k} a_e x^F(e) = \alpha$. for all $v \in V(K_k)$.

Condition (L) in theorem 6 always holds. Theorem 6 which is a *trivial lifting*, allows to propagate the facet defining inequality on $\mathcal{P}_n$ to $\mathcal{P}_{n'}$ for all $n' \geq n$

2.1 H_p inequalities

Let H_p be the graph obtained from a cycle C_4 by replacing a given vertex v by a clique K_p and creating an edge between every vertex of K_p with the neighbors of v.

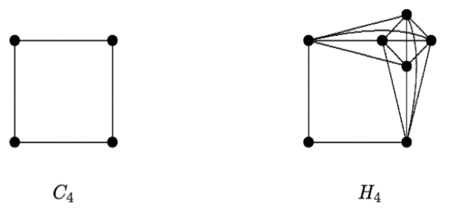


FIG. 1: Construction of H_p from C_4, ($p = 4$).

We have the following inequality :

$$x(H_p) \leq p + 1. \tag{4}$$

We call it the H_p inequality. Then we have the result stated below:

Theorem 7 *The H_p inequality defines a facet for $\mathcal{P}_n$ with $n \geq p + 3$.*

2.2 H'_p-inequalities

Let H'_p be the graph obtained from a paw graph by applying the following operation: From a graph with the structure of a paw. Choose a vertex with a maximum degree v' and clone i.e.,

- Create a vertex that is clone (identical) of v', and another clone for the pendent vertex adjacent to v'.
- Create an edge between the clone of v' and the vertex v'.
- Create an edge between the clone of v' and the non-pendent vertices neighbors of v'.

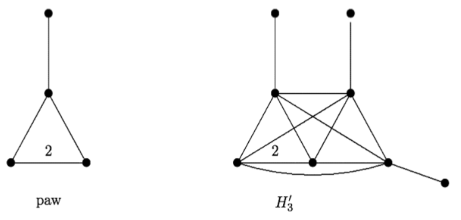


FIG. 2: Construction of H'_p from a paw, ($p = 3$)

This leads to the graph H'_1. We apply the same operation iteratively on H'_i. H'_p is obtained at the end of the p^{th} iteration.

Remark 1

- *The graph H'_p contains one $p+3$-clique and $p+1$ pendent edges.*
- *H'_p has a perfect matching consisting on the $p+1$ pendent edges and one edge from the $p+3$-clique.*

We define perfect a matching M' on H' and S' be the set of all non pendent vertices. We define the H'-inequality as follows:

$$x(E(S')) + x(M'_p) \leq |M'_p| + 1. \tag{5}$$

Theorem 8 *The H'_p-inequalities are valid and define facets for $\mathcal{P}_n$ when $n \geq 2p+3$.*

As a conclusion we have the following result:

Theorem 9 *$\mathcal{P}_4$ is completely characterized by the following inequalities :*

$$\begin{aligned} x(C) &\leq 2 \text{ for all } C \in \mathcal{C}_3 \cup \mathcal{C}_4 \text{ in } K_4, \\ x(C_H) + x(M_H) &\leq 3 \text{ for all paw } H \text{ in } K_4, \\ 0 \leq x(e) &\leq 1 \text{ for all } e \in E_4. \end{aligned}$$

References

[1] A. Agra, D. Cardoso, O. Cerdeira, E. Rocha, A spanning star forest model for the diversity problem in automobile industry, 2005.

[2] C.T. Nguyen, J. Shen, M. Hou, L. Sheng, W. Miller, L.Zhang: Approximating the spanning star forest problem and its applications to genomic sequence alignment. SIAM Journal of Computing. 38(3), pages 946-962 , 2007.

[3] V. H. Nguyen. The maximum weight spanning star forest problem on cactus graphs. Discrete Mathematics, Algorithms and Applications, Vol.07, No.02, 2015.

Decomposition Methods for Quadratic Programming

Enrico Bettiol[1], Alberto Ceselli[2], Lucas Létocart[1], Francesco Rinaldi[3], Emiliano Traversi[1]

[1] University of Paris 13
99 Avenue Jean Baptiste Clément, 93430 Villetaneuse, France
{enrico.bettiol,lucas.letocart,emiliano.traversi}@lipn.univ-paris13.fr

[2] University of Milan
Via Bramante, 65 26013 Crema, Italy
alberto.ceselli@unimi.it

[3] University of Padova
Via Trieste, 63 35121 Padova - Italy
rinaldi@math.unipd.it

Abstract

In this work we present two decomposition methods for quadratic problems.
First, we propose a methodological analysis on a family of reformulations combining Dantzig-Wolfe decomposition and Quadratic Convex Reformulation principles for binary quadratic problems. As a representative case study, we apply them to a cardinality constrained quadratic knapsack problem.
Secondly, we analyze a simplicial decomposition like algorithmic framework that handles convex quadratic programs in an effective way. In particular, we propose two tailored strategies for solving the master problem and we describe a few techniques for speeding up the solution of the pricing problem. We report extensive numerical experiments on both real-world and generic quadratic programs.

Keywords : *Mathematical Programming, Decomposition Methods, Quadratic Optimization.*

Column Generation for the Energy-Efficient in Multi-Hop Wireless Networks Problem

Sonia Vanier

SAMM, Université Paris1 Panthéon-Sorbonne

Sonia.Vanier@univ-paris1.fr

Abstract

Energy-efficient designs are one of the most outstanding challenges in wireless communication networks. Saving energy in multi-hop wireless networks usually consists in maximizing the lifetime of the network. This can be done by using the minimum number of nodes to route the traffic and turning of the maximum number of nodes. This problem was studied in [1] using integer programming and simulation methods.
We present a new model based on the arc-path formulation of the unsplittable multicommodity flow problems. We propose a column generation approach to solve the problem. Then we introduce new classes of valid inequalities, and give separation algorithms for a branch-and-cut-and-price framework.

Keywords : *Mixed Integer Linear Programming, Network Optimization, Column Generation, Graph theory, Valid Inequalities.*

1 Problem Formulation

Given a network $G = (V, E)$ defined by a set of nodes V and a set of arcs E, each arc has a capacity y_{ij}. Let D denote the set of commodities. Each commodity has an origin s, a destination t and a flow value to route M^{st}. We would like to concurrently route every demand on a single path from s to t without violating the capacities and interference constraints.
Interferences can be avoided if we can prohibit interfering nodes to transmit at the same time [3][4]. More precisely, we can consider a transmission from node i to node j as succeeded if the distance between those nodes is lower then the distance between j and any other node transmitting at the same time than i. For the interferences we use the model based on conflict graph G' proposed in [1] [2].

Let $\wp^{st}$ be the set of all possible simple paths for commodity st in the graph G and C the set of all cliques of the conflict graph G'.

Our problem can be described using a binary flow variable x_p^{st} for each commodity st and each path $p \in \wp^{st}$ that takes value of 1 if the commodity uses the path p, 0 otherwise.
We introduce also the binary variable x_i for each node i of the network that takes value of 1 if the node i is used and 0 otherwise.

To respect the clique's utilization rate, we have the following residual capacity constraint for each clique c of G':

$$\sum_{st \in D} \sum_{p \in \wp^{st}} \sum_{(i,j) \in p \cap c} \frac{M^{st}}{y_{ij}} x_p^{st} \leq 1 \; \forall c \in C$$

The goal is to minimize the number of used nodes $\min \sum_{i \in V} x_i$.
The node i is used if $\exists p \in \wp^{st}$ such that $i \in p$ and $x_p^{st} > 0$.

Using this notations the Energy-Efficient in Multi-Hop Wireless Networks problem can be formulated as the following integer linear program:

$$\begin{cases} z = \min \sum_{i \in V} x_i \\ \sum_{st \in D} \sum_{p \in \wp^{st}} \sum_{(i,j) \in p \cap c} \frac{M^{st}}{y_{ij}} x_p^{st} \leq 1 \; \forall c \in C \qquad (1.1) \\ \sum_{p \in \wp^{st}} x_p^{st} = 1 \; \forall st \in D \qquad (1.2) \\ -\sum_{st \in D} \sum_{p \in \wp^{st}, i \in p} M^{st} x_p^{st} + (\max_j y_{ij} + \max_j y_{ji}) \; x_i \geq 0 \; \forall i \in V \quad (1.3) \\ \sum_{st \in D} \sum_{p \in \wp^{st}, i \in p} M^{st} x_p^{st} - (\min_{st \in D} M^{st}) \; x_i \geq 0 \; \forall i \in V \quad (1.4) \\ x_i \in \{0,1\} \forall i \in V, x_p^{st} \in \{0,1\} \; \forall st \in D, \forall p \in \wp^{st} \end{cases}$$

Constraints (1.1) are capacity constraints for each clique $c \in C$ of the conflict graph. Constraints (1.2) are unsplittable demand contraints, while contraints (1.3) and (1.4) ensure that a node i is used if it receives or sends traffic.

We have developed a column génération approach to solve the problem, introduced new classes of valid inequalities, and gived separation algorithms for a branch-and-cut-and-price framework.

References

[1] Alexandre Laubé, Phd Thesis, Agrégation de trafic pour réduire la consommation énergétique globale dans les réseaux sans fil multi-sauts, 2017.

[2] Alexandre Laubé, Steven Martin, Dominique Quadri, Khaldoun Al Agha, Guy Pujolle. *A Flow Aggregation Metric for Shortest Path Routing Algorithms in Multi-hop Wireless Netwoks.* WCNC, San Francisco, 1-6, 2017.

[3] V. Padmanabhan K.Jain, J.Padhye and L.Qiu. *Impact of interference in Multi-hop Wireless Netwok performance.* MobiCom ACM, 2003.

[4] J. Musacchio, R. Gupta and J. Walrand *Sufficient rate constraint for QoS flows in ad-hoc netwoks.* Ad hoc Networks, ACM, vol. 5, 429-443, 2006.

A Multiplicative Weights Update Algorithm for a Class of Pooling Problems

Luca Mencarelli[1]

CNRS, UMR 7161, LIX, École Polytechnique
Palaiseau, France
mencarelli@lix.polytechnique.fr

Abstract

In this short paper, we give several intuitions about the application of the Multiplicative Weights Update Algorithm to a broad class of Pooling Problem without inter-layers flows. First, we describe two different pointwise reformulations for the Pooling Problem. Secondly, we employ the Multiplicative Weights Update Algorithm in order to determine a solution whose grade of infeasibility is relatively small. We compare the performances of the Multiplicative Weights Update Algorithm against the standard MultiStart Algorithm.

Keywords : *Pooling Problem, Multiplicative Weights Update Algorithm, Heuristic Algorithms.*

1 The Pooling Problem

The *Pooling Problem* (PP) identifies a well-known class of non-convex non-concave optimization problem arising in blending industries where the set of final products is the result of the mixture of several input raw materials. Given is a network topology consisting in a graph whose nodes are partitioned into three sets: the set $[I] := \{1, \ldots, I\}$ of the input nodes, the set $[L] := \{1, \ldots, L\}$ of the pools, and the set $[J] := \{1, \ldots, J\}$ of the output nodes. The topology of the network is also described by the set T_X of arcs (i, ℓ) connecting input i and pool ℓ and the set T_Y of arcs (ℓ, j) connecting pool ℓ and output j. A fraction of each input flows through the pool where it is blended with the other raw materials, and, finally, composes the final output, whose quality attributes $k \in \{1, \ldots, K\} =: [K]$ are monitored. For a survey about the PP we refer the interested reader to the excellent paper by Misener and Floudas [7], from which we borrow the mathematical notation.

In this extended abstract we consider the following (restrictive) assumptions:

(i) the pools define a single layer in the network topology,

(ii) there is no inter-layers flow, i.e., all flows are intra-layers, and

(iii) there is no flow bypassing the pools.

Let $x_{i\ell}$ and $y_{\ell j}$ be the decision variables representing the flow from input i to pool ℓ and the flow from pool ℓ to output j, respectively. Moreover, let $p_{\ell k}$ be the decision variables indicating the quality level of the attribute k in the pool ℓ. The following mathematical formulation

$$\min \sum_{(i,\ell)\in T_X} c_i\, x_{i\ell} - \sum_{(\ell,j)\in T_Y} d_j\, y_{\ell j} \tag{1}$$

$$\text{s.t.} \sum_{i:(i,\ell)\in T_X} x_{i\ell} \le S_\ell \qquad \forall \ell \in [L] \tag{2}$$

$$\sum_{\ell:(\ell,j)\in T_Y} y_{\ell j} = D_j \qquad \forall j \in [J] \tag{3}$$

$$\sum_{i:(i,\ell)\in T_X} x_{i\ell} - \sum_{\ell:(\ell,j)\in T_Y} y_{\ell j} = 0 \qquad \forall \ell \in [L] \tag{4}$$

$$\sum_{i:(i,\ell)\in T_X} C_{ik}\, x_{i\ell} = p_{\ell k} \sum_{j:(\ell,j)\in T_Y} y_{\ell j} \qquad \forall \ell \in [L],\ \forall k \in [K] \tag{5}$$

$$\sum_{\ell:(\ell,j)\in T_Y} p_{\ell k}\, y_{\ell j} \geq P^L_{jk} \sum_{\ell:(\ell,j)\in T_Y} y_{\ell j} \qquad \forall j \in [J],\ \forall k \in [K] \tag{6}$$

$$\sum_{\ell:(\ell,j)\in T_Y} p_{\ell k}\, y_{\ell j} \leq P^U_{jk} \sum_{\ell:(\ell,j)\in T_Y} y_{\ell j} \qquad \forall j \in [J],\ \forall k \in [K] \tag{7}$$

is generally referred to as the *P-formulation* for the PP. The objective (1) aims at minimizing the overall net cost, while the constraints represent the topological characteristics of the network and the technological limitations. Equations (2) and (3) represent the pool capacity and the product demand constraints, respectively. Constraints (4) impose the flow conservation for each pool, while Equations (5) force the quality balance for each attribute in each pool, and finally Equations (6)-(7) take into account the level quality of each attribute in the composition of the final product.

Ben-Tal et al. [2] introduce a new decision variable $q_{i\ell}$ representing the flow rate from input i to pool ℓ, and defined as followed,

$$x_{i\ell} = q_{i\ell} \sum_{j:(\ell,j)\in T_Y} y_{\ell j} \qquad \forall (i,\ell) \in T_X, \tag{8}$$

giving rise to the so-called *Q-formulation* for the PP:

$$\min \sum_{\substack{(i,\ell)\in T_X\\(\ell,j)\in T_Y}} c_i\, q_{i\ell}\, y_{\ell j} - \sum_{(\ell,j)\in T_Y} d_j\, y_{\ell j} \tag{9}$$

$$\text{s.t.} \sum_{i:(i,\ell)\in T_X} x_{i\ell} \leq S_\ell \qquad \forall \ell \in [L] \tag{10}$$

$$\sum_{\ell:(\ell,j)\in T_Y} y_{\ell j} = D_j \qquad \forall j \in [J] \tag{11}$$

$$\sum_{\substack{\ell:(\ell,j)\in T_Y\\ i:(i,\ell)\in T_X}} C_{ik}\, q_{i\ell}\, y_{\ell j} \geq P^L_{jk} \sum_{\ell:(\ell,j)\in T_Y} y_{\ell j} \qquad \forall j \in [J],\ \forall k \in [K] \tag{12}$$

$$\sum_{\substack{\ell:(\ell,j)\in T_Y\\ i:(i,\ell)\in T_X}} C_{ik}\, q_{i\ell}\, y_{\ell j} \leq P^U_{jk} \sum_{\ell:(\ell,j)\in T_Y} y_{\ell j} \qquad \forall j \in [J],\ \forall k \in [K] \tag{13}$$

$$\sum_{i:(i,\ell)\in T_X} q_{i\ell} = 1 \qquad \forall \ell \in [L] \tag{14}$$

$$0 \leq q_{i\ell} \leq 1 \qquad \forall i \in [I],\ \forall \ell \in [L] \tag{15}$$

Finally, Quesada and Grossmann [8] and Tawarmalani and Sahinidis [9] add the following RLT-based contraint to the *Q-formulation*, obtaining the *PQ-formulation*:

$$\sum_{i:(i,\ell)\in T_X} q_{i\ell}\, y_{\ell j} = y_{\ell j} \qquad \forall \ell \in [L],\ \forall j \in [J]. \tag{16}$$

2 The Multiplicative Weights Update Algorithm

The Multiplicative Weights Update (MWU) Algorithm is a stochastic procedure where a decision maker has to choose between a set of decisions each of them characterized by an unknown payoff. The goal of the decision maker consists in limiting the resulting losses up to the overall payoff obtained by repeatedly taking the best decision. The MWU Algorithm has a very broad range of application from Machine Learning to Game Theory. We refer the interested reader to the survey [1]. In [6] the MWU Algorithm was extended to the Mixed Integer Nonlinear Programming (MINLP). The authors proposed a two-phase algorithm: in the first phase a solution point is generated by solving the so-called *pointwise reformulation*, a reformulation of the original problem where several terms are substituted by a fixed parameter θ; while, in the second (optional) phase, *the refinement*, the original problem was solved starting from the solution obtained in the previous step.

3 Pointwise Reformulations

In the formulation (9)-(13), the decision variables are clearly partitioned into two sets: the $q_{i\ell}$ variables and the $y_{\ell j}$ variables, indicating the flow ratio from the input i to pool ℓ and the absolute flow from pool ℓ to output j, respectively. Therefore, we explore two possible pointwise reformulations involving the terms where the variables occour nonlinearly, i.e., the bilinear terms in the objective function (9), and the constraints (12)-(13). In the first reformulation we replace the first class of decision variables by parameters θ, while in the other one we operate the same substitution for the second class of variables.

In the pointwise reformulation we should add slack variables to each constraint, since we are fixing several variables to a given numerical value: this operation might generate infeasibilities. We could also consider slack variables in the original problem: this choice allows us to deal with cases in which the original PP instance is infeasible. The pointwise reformulation is characterized by two conflicting targets: optimality and feasibility. Hence, we apply standard scalarization approach, such as, e.g., Weighted Sum technique [4], setting, as objective function, the convex combination between the original objective function and the maximum slack. Let $\delta \in [0,1]$ be a given penalty parameter. For instance, the first pointwise reformulation can be written as follows:

$$\min\ (1-\delta)\left(\sum_{\substack{(i,\ell)\in T_X\\(\ell,j)\in T_Y}} c_i\,\theta_{i\ell}\,y_{\ell j} - \sum_{(\ell,j)\in T_Y} d_j\,y_{\ell j}\right) + \delta\, s \tag{17}$$

$$\text{s.t.}\ \sum_{i:(i,\ell)\in T_X} x_{i\ell} \le S_\ell + s_{1,\ell} \qquad \forall \ell\in[L] \tag{18}$$

$$\sum_{\ell:(\ell,j)\in T_Y} y_{\ell j} = D_j + s_{2,j} \qquad \forall j\in[J] \tag{19}$$

$$\sum_{\substack{\ell:(\ell,j)\in T_Y\\ i:(i,\ell)\in T_X}} C_{ik}\,\theta_{i\ell}\,y_{\ell j} \ge P^L_{jk} \sum_{\ell:(\ell,j)\in T_Y} y_{\ell j} - s_{3,jk} \qquad \forall j\in[J],\ \forall k\in[K] \tag{20}$$

$$\sum_{\substack{\ell:(\ell,j)\in T_Y\\ i:(i,\ell)\in T_X}} C_{ik}\,\theta_{i\ell}\,y_{\ell j} \le P^U_{jk} \sum_{\ell:(\ell,j)\in T_Y} y_{\ell j} + s_{4,jk} \qquad \forall j\in[J],\ \forall k\in[K] \tag{21}$$

$$\sum_{i:(i,\ell)\in T_X} q_{i\ell} = 1 + s_{5,\ell} \qquad \forall \ell\in[L] \tag{22}$$

$$\sum_{i:(i,\ell)\in T_X} \theta_{i\ell}\,y_{\ell j} = y_{\ell j} + s_{6,\ell j} \qquad \forall \ell\in[L],\ \forall j\in[J] \tag{23}$$

$$0 \le q_{i\ell} \le 1 \qquad \forall i\in[I],\ \forall \ell\in[L] \tag{24}$$

$$s_{1,\ell} \ge 0 \qquad \forall \ell\in[L] \tag{25}$$

$$s_{3,jk} \ge 0 \qquad \forall j\in[J],\ \forall k\in[K] \tag{26}$$

$$s_{4,jk} \ge 0 \qquad \forall j\in[J],\ \forall k\in[K] \tag{27}$$

$$s \geq s_{1,\ell} \qquad \forall \ell \in [L] \tag{28}$$
$$s \geq s_{2,j} \qquad \forall j \in [J] \tag{29}$$
$$s \geq -s_{2,j} \qquad \forall j \in [J] \tag{30}$$
$$s \geq s_{3,jk} \qquad \forall j \in [J],\ \forall k \in [K] \tag{31}$$
$$s \geq s_{4,jk} \qquad \forall j \in [J],\ \forall k \in [K] \tag{32}$$
$$s \geq s_{5,\ell} \qquad \forall \ell \in [L] \tag{33}$$
$$s \geq -s_{5,\ell} \qquad \forall \ell \in [L] \tag{34}$$
$$s \geq s_{6,\ell j} \qquad \forall \ell \in [L],\ \forall j \in [J] \tag{35}$$
$$s \geq -s_{6,\ell j} \qquad \forall \ell \in [L],\ \forall j \in [J] \tag{36}$$

4 Computational Experiments

We compare the two possible pointwise reformulations, considering for each of them several approaches to compute the costs/gains for the MWU Algorithm. Furthermore, we add slack variables to the original problem in the refinement phase since we would like to handle possibly infeasibilities in the PP instances. The benchmark we consider consists in the standard MultiStart (MS) Algorithm, in which a starting point is randomly generated and a local algorithm, i.e., a procedure producing solutions with local optimal certificate, is applied from the starting point. We use Ipopt [5] as local solver for both MWU and MS and Couenne [3] as global solver. For both the local solver and the global one we maintain the default settings. Preliminary computational experiments show that the MWU Algorithm is able to recover a solution to which a lower value of the slack variable is associated; on the contrary the MS produces a worse solution in terms of feasibility and optimality.

References

[1] S. Arora, E. Hazan, and S. Kale. *The multiplicative weights update method: A meta-algorithm and applications.* Theory of Computing, 8:121–164, 2012.

[2] A. Ben-Tal, G. Eiger, and V. Gershovitz. *Global minimization by reducing the duality gap.* Mathematical Programming, 63(1–3):193–212, 1994.

[3] Couenne. URL `https://projects.coin-or.org/Couenne`.

[4] M. Ehrgott. *Multicriteria Optimization.* Springer-Verlag, Berlin, 2005.

[5] Ipopt. URL `https://projects.coin-or.org/Ipopt`.

[6] L. Mencarelli, Y. Sahraoui, and L. Liberti. *A multiplicative weights update algorithm for MINLP.* EURO Journal on Computational Optimization, 5(1–2):31–86, 2017.

[7] R. Misener and C.A. Floudas. *Advances for the pooling problems: Modeling, global optimization, and computational studies.* Applied Mathematics and Computation, 8(1):2–22, 2009.

[8] I. Quesada and I.E. Grossman. *Global optimization of bilinear process networks with multicomponent flows.* Computers & Chemical Engineering, 19(12):1219–1242, 1995.

[9] M. Tawarmalani and N.V. Sahinidis. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: Theory, applications, software, and applications.* Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

Implicit heavy subgraph conditions for hamiltonicity of almost distance-hereditary graphs

Wei Zheng[1,2], Hajo Broersma[1], Ligong Wang[2]

[1] Faculty of EEMCS, University of Twente, Enschede, The Netherlands
zhengweimath@163.com,h.j.broersma@utwente.nl

[2] Department of Applied Mathematics, Northwestern Polytechnical University, Xi'an, China
lgwangmath@163.com

Abstract

We recently proved two new results on the hamiltonicity of almost distance-hereditary graphs, involving implicit degree conditions on claws, i.e., induced subgraphs isomorphic to $K_{1,3}$. A graph G of order n is called implicit 1-heavy if at least one of the end vertices of each induced claw of G has implicit degree at least $n/2$, and G is called implicit claw-heavy if each induced claw of G has a pair of end vertices with implicit degree sum at least n. A graph G is said to be almost distance-hereditary if each connected induced subgraph H of G has the property $d_H(x,y) \leq d_G(x,y) + 1$ for any pair of vertices $x, y \in V(H)$. We recently proved that every 2-connected implicit claw-heavy almost distance-hereditary graph is hamiltonian, and that every 3-connected implicit 1-heavy almost distance-hereditary graph is hamiltonian. These results improve two recent results due to Chen and Ning.

Keywords : *Implicit degree, (almost) distance-hereditary graph, Hamilton cycle, induced claw, claw-heavy.*

1 Introduction

The general motivation for our research is to obtain new results in hamiltonian graph theory that improve or extend earlier results. Hamiltonicity has been one of the central and well-studied concepts in graph theory since the early results involving sufficient degree conditions due to Dirac and Ore, that date back to the 1950s and 1960s. These results have been generalized and extended in many different ways, and the subject still attracts a lot of attention. From our perspective of studying hamiltonicity, claw-free graphs or conditions on claws are particularly interesting. One of the reasons is that during the last decades many results have demonstrated that sufficient degree conditions for guaranteeing hamiltonicity of general graphs can be substantially relaxed if one restricts oneself to claw-free graphs or puts conditions on claws. We refer the interested reader to the surveys [2, 4, 8] for more background and details about claw-free graphs and line graphs.

A natural way of extending known results on claw-free graphs is to put restrictions on claws instead of forbidding them as induced subgraphs. The most popular way of relaxing the claw-freeness, has been to impose restrictions on the degrees of the end vertices of induced claws. We refer the interested reader to the papers [3, 5, 11] as examples in which this approach was explored successfully by imposing Dirac-type and Ore-type degree conditions on the end vertices of induced claws. We have established more recent examples of this approach, in which the degree conditions were further relaxed by replacing them by implicit degree conditions. Our new results are based on the concept of implicit degree that was introduced by Zhu, Li, and Deng [15].

To put our new results in the right perspective, we will first focus on some earlier results in which the class of claw-free graphs is narrowed down by adding other structural conditions, in particular the condition that the graphs are almost distance-hereditary. We first introduce some of the essential terminology and notation.

2 Preliminaries

We use the textbook of Bondy and Murty [1] for any terminology and notation not defined here. Let G be a connected graph. Then, for any two vertices $u, v \in V(G)$, the *distance* between u and v in G, denoted by $d_G(u, v)$, is the length of a shortest (u, v)-path in G. The graph G is called *almost distance-hereditary* if each connected induced subgraph H of G has the property $d_H(x, y) \leq d_G(x, y) + 1$ for any pair of vertices $x, y \in V(H)$. Considering the intersection of the classes of almost distance-hereditary graphs and claw-free graphs, about ten years ago Feng and Guo [9] proved the following result.

Theorem 1 *Every 2-connected almost distance-hereditary claw-free graph is hamiltonian.*

By relaxing the condition of being claw-free by putting different degree conditions on the end vertices of every induced claw, several results on hamiltonicity of almost distance-hereditary graphs were obtained. A vertex v of a graph G on n vertices is called *heavy* if the degree $d(v) \geq n/2$. The graph G is called *1-heavy* (respectively *2-heavy*) if at least one (respectively two) of the end vertices of each induced claw of G are heavy. By using this concept of a 2-heavy graph, Feng and Guo [10] extended Theorem 1 in the following way.

Theorem 2 *Every 2-connected almost distance-hereditary 2-heavy graph is hamiltonian.*

Replacing the Dirac-type degree condition by an Ore-type degree condition, Fujisawa and Yamashita [11] introduced the notion of a claw-heavy graph. A graph G on n vertices is called *claw-heavy* if each claw of G has a pair of end vertices with degree sum at least n. Chen and Ning [7] recently used the above notions to obtain the following two results related to Theorems 1 and 2.

Theorem 3 *Every 2-connected almost distance-hereditary claw-heavy graph is hamiltonian.*

Theorem 4 *Every 3-connected almost distance-hereditary 1-heavy graph is hamiltonian.*

Our aim was to improve the above two results by further relaxing the degree conditions to implicit degree conditions. We first recall the definition of the concept of implicit degree due to Zhu et al. [15]

For a vertex $v \in V(G)$, let $N(v) = \{u \in V(G) \mid uv \in E(G)\}$, let $N_2(v) = \{u \in V(G) \mid d(u, v) = 2\}$, and let $M_2(v) = \max\{d(u) \mid u \in N_2(v)\}$. Suppose that $d(v) = \ell + 1$ for some integer $\ell \geq 0$. If $N_2(v) \neq \emptyset$ and $d(v) \geq 2$, then let $d_1 \leq d_2 \leq d_3 \leq \ldots \leq d_\ell \leq d_{\ell+1} \leq \ldots$ denote the degree sequence of the vertices of $N(v) \cup N_2(v)$. Define $d^*(v) = d_{\ell+1}$ if $d_{\ell+1} > M_2(v)$, and $d^*(v) = d_\ell$ otherwise. Then the *implicit degree* of v, denoted by $id(v)$, is defined as $id(v) = \max\{d(v), d^*(v)\}$. If $N_2(v) = \emptyset$ or $d(v) \leq 1$, then we define $id(v) = d(v)$.

Clearly, by the definition $id(v) \geq d(v)$ for every vertex v. Replacing the degree conditions in the above definitions by implicit degree conditions, we say that a graph G on n vertices is *implicit 1-heavy* if at least one end vertex of each claw of G is *implicit heavy*, i.e., has implicit degree at least $n/2$. We call G *implicit claw-heavy* if each claw of G has a pair of end vertices with implicit degree sum at least n.

3 Our results

We have recently proved the following two improvements of Theorems 3 and 4.

Theorem 5 *Every 2-connected almost distance-hereditary implicit claw-heavy graph is hamiltonian.*

Theorem 6 *Every 3-connected almost distance-hereditary implicit 1-heavy graph is hamiltonian.*

In the presentation, we will sketch the key ingredients of the proofs of these two new results.

It is obvious that the implicit degree conditions in the statements of Theorems 5 and 6 cannot be omitted. The complete bipartite graphs $K_{m,m+1}$ show that a large connectivity together with the condition of almost distance-hereditary cannot guarantee a graph to be hamiltonian.

By the following examples, we present an infinite class of graphs that do not satisfy the conditions of Theorems 3 and 4, but that can be easily verified to be hamiltonian using Theorem 5 or 6.

Let k be a nonnegative integer. For any $m \geq k+1$, let G_m denote the join of a complete graph K_{2m} and a graph H, where H is the disjoint union of a K_4 and m copies of a K_2. Then G_m is a $2m$-connected graph of order $n = 4m+4$, and it is easy to check that G_m is almost distance-hereditary and hamiltonian. The degrees and implicit degrees of the vertices of G_m can also be determined in a straightforward way. For any vertex u belonging to the m copies of a K_2 of H, we obtain that $d(u) = 2m+1 < \frac{n}{2}$ and $id(u) = 2m+3 > \frac{n}{2}$; for any vertex v belonging to the K_{2m}, we obtain that $id(v) \geq d(v) = 4m+3 > \frac{n}{2}$; for any vertex w belonging to the K_4 of H, we obtain that $id(w) \geq d(w) = 2m+3 > \frac{n}{2}$. Using this, it is easy to check that G_m is implicit claw-heavy (and so implicit 1-heavy) but not 1-heavy (and so not claw-heavy) for all $m \geq 3$.

References

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

[2] H.J. Broersma, *On some intriguing problems in hamiltonian graph theory – a survey.* Discrete Math., 251:47–69, 2002.

[3] H.J. Broersma, Z. Ryjáček, and I. Schiermeyer, *Dirac's minimum degree condition restricted to claws.* Discrete Math., 167/168:155–166, 1997.

[4] H.J. Broersma, Z. Ryjáček, and P. Vrána, *How many conjectures can you stand – a survey.* Graphs Combin., 28(1):57–75, 2012.

[5] R. Čada, *Degree conditions on induced claws.* Discrete Math., 308(23):5622–5631, 2008.

[6] J. Cai and Y. Zhang, *Fan-type implicit-heavy subgraphs for hamiltonicity of implicit claw-heavy graphs.* Inform. Process. Lett., 116:668–673, 2016.

[7] B. Chen and B. Ning, *Hamilton cycles in almost distance-hereditary graphs.* Open Math., 14:19–28, 2016.

[8] R.J. Faudree, E. Flandrin, and Z. Ryjáček, *Claw-free graphs – a survey.* Discrete Math., 164:87–147, 1997.

[9] J. Feng and Y. Guo, *Hamiltonian problem on claw-free and almost distance-hereditary graphs.* Discrete Math., 308(24):6558–6563, 2008.

[10] J. Feng and Y. Guo, *Hamiltonian cycle in almost distance-hereditary graphs with degree condition restricted to claws.* Optimization, 57(1):135–141, 2008.

[11] J. Fujisawa and T. Yamashita, *Degree conditions in induced subgraphs for hamiltonicity.* Discrete Math., Preprint.

[12] X. Huang, *Hamilton cycles in implicit claw-heavy graphs.* Inform. Process. Lett., 114:676–679, 2014.

[13] H. Li, W. Ning, and J. Cai, *An implicit degree condition for cyclability in graphs.* FAW-AAIM 2011, LNCS 6681:82–89, 2011.

[14] M. Matthews and D. Sumner, *Hamiltonian results in $K_{1,3}$-free graphs.* J. Graph Theory, 8:139–146, 1984.

[15] Y. Zhu, H. Li, and X. Deng, *Implicit-degrees and circumferences.* Graphs Combin., 5:283–290, 1989.

On matching and distance property of m-barrele Fullerene

Afshin Behmaram[1], Cédric Boutillier[2]

[1] Faculty of Mathematical Sciences, University of Tabriz, Tabriz, Iran
`behmaram@tabrizu.ac.ir`

[2] Laboratoire de Probabilités et Modèles Aléatoires, UPMC Univ. Paris 06, 4 place Jussieu, F-75005 Paris, France
`cedric.boutillier@upmc.fr`

Abstract

A connected planar cubic graph is called m-barrel fullerene and denoted by $F(m, k)$, if have the following structure: The first circle is an m-gon. Then m-gon is bounded by m pentagons. After that we have additional k layers of hexagon. At the last circle m-pentagons connected to the second m-gon.In this paper we enumerate asymptotic perfect matching number in m-barrel fullerene graphs by two different methods and show that the results are equal.then we show some distance property of m-barrel fullerene.

Keywords : *Perfect matching, distance, fullerene graph, m-barrel fullerne.*

1 Introduction

The m-barrel fullerene with k layers of hexagons, denoted by $F(m, k)$, can be defined as a sequence of concentric layers as follows: the first circle is an m-gon. This m-gon is bounded by m pentagons. After that we have additional k layers of m of hexagon. Then one again has a circular layer with m-pentagons connected to the second m-gon, represented by the outer face. m-barrell fullerenes can be neatly represented graphically using a sequence of $k + 3$ concentric circles with monotonically increasing radii such that the innermost and the outermost circle each have m vertices (representing, hence, two m-gons), while all other circles have $2m$ vertices each, connecting alternatively to vertices of the larger or smaller circle to create hexagonal an pentagonal faces. An example is shown in Figure 1.

The m-barrel fullerenes are the main subjects of the present paper, since their highly symmetric structure allows for obtaining good bounds and even exact results on their quantitative graph properties. For example Kutnar and Marušič in [2] studied Hamiltonicity and cyclic edge-conectivity of $F(5, k)$. See also [1] for some structural results about m-barrel fullerene graphs, such as the diameter, Hamiltonicity and the leapfrog transformation. A *matching* M in a graph G is a collection of edges of G such that no two edges of M share a vertex. If every vertex of G is incident to an edge of M, the matching M is said to be *perfect*. A perfect matching is also often called a *dimer configuration* in mathematical physics and chemestry. Perfect matchings have played an important role in the chemical graph theory.

The goal of this paper is to compute the growth constant $\rho(m)$ for the number of perfect matchings for the family of graphs $F(m, k)$ for a fixed m, as k goes to infinity:

$$\rho(m) = \lim_{k\to\infty} \Phi(F(m, k))^{1/k}. \tag{1}$$

then we proved some distance property of m-barrel Fullerene.

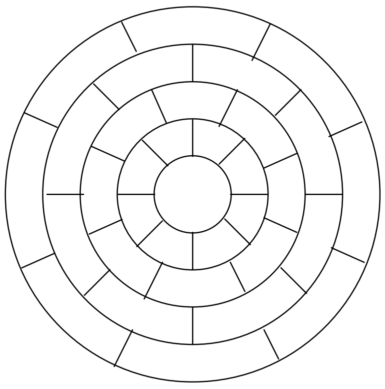


FIG. 1: The m-barrel fullerene $F(8,2)$.

1.1 Figures, equations and theorems

Theorem 1 *Let $m \geq 3$. The growth constant for the family of m-barrel fullerenes is equal to*

$$\rho(m) = \prod_{j=1}^{\lfloor \frac{m+1}{3} \rfloor} \left(2 \cos \frac{\pi(2j-1)}{m} \right)^2 . \tag{2}$$

References

[1] Afshin Behmaram , Tomislav Doslic, Shmuel Friedlsnd. *Matchings in generalized Fullerene.* Ars Mathematica contemporanea ,Vo11, No2 (2016), pp 301-311.

[2] K. Kutnar, D. Marušič, *On cyclic edge-connectivity of fullerenes.* Discrete Appl. Math, 156 (2008) 1661–1669.

Solving the Green Vehicle Routing Problem with Capacitated Alternative Fuel Stations

Maurizio Bruglieri[1], Simona Mancini[2], Ornella Pisacane[3]

[1] Dipartimento di Design, Politecnico di Milano, Milano, Italy
maurizio.bruglieri@polimi.it

[2] Diparimento di Matematica e Informatica, Universitá degli Studi di Cagliari, Cagliari, Italy
simona.mancini@unica.it

[3] Dipartimento di Ingegneria dell'Informazione, Universitá Politecnica delle Marche, Ancona, Italy
pisacane@dii.univpm.it

Abstract

The *Green Vehicle Routing Problem* (GVRP) aims to efficiently route a fleet of Alternative Fuel Vehicles (AFVs), in order to serve a set of customers, minimizing the total travel distance. Each AFV leaves from a common depot, serves a subset of customers and returns to the depot, without exceeding a maximum duration. Due to their limited driving range, the AFVs may need to refuel one or more times at the Alternative Fuel Stations (AFSs), along their route. In this work, we introduce the GVRP with Capacitated AFSs (GVRP-CAFS) in which only a limited number of AFVs can refuel at the same time at each AFS to account for their limited capacity. In order to solve the GVRP-CAFS, we propose an exact approach in which a route is the composition of paths, each handling a subset of customers without intermediate stops at AFSs. Firstly, all feasible non-dominated paths are generated. Secondly, via a path-based Mixed Integer Programming model, the paths are selected and properly combined each other to generate the routes of the optimal GVRP-CAFS solution. To reduce the computational times, a relaxed version of the path-based model is solved and then, the violated constraints are iteratively added. Some preliminary results are also discussed.

Keywords : *Vehicle Routing Problem, Alternative Fuel Vehicles, Mixed Integer Programming*

1 Introduction and statement of the problem

Nowadays, the transportation companies are requested to provide more competitive services in a more sustainable way, through efficient trip planning, smart distribution systems and the use of new technologies. Among the latter, the *Alternative Fuel Vehicles* (AFVs), i.e., vehicles that use alternative fuel (e.g., methanol and electricity), play a key role, contributing to reduce both the CO_2 emissions and the noise pollution. However, purchasing an AFV still remains very expensive. Moreover, the AFV driving range is still limited and, in fact, it may require several stops at the *Alternative Fuel Stations* (AFSs) in a trip. In addition, the AFSs are currently not widespread on the territory. Therefore, it becomes very significant to properly plan the AFV trips in order to prevent drivers remaining without enough fuel to either reach the closest AFS or return to the depot.

A relatively new operational research area is focused on the *Green Vehicle Routing Problem* -GVRP ([1]), introduced in [2]. It aims to route a fleet of m AFVs, based on a common depot (denoted by 0), minimizing the total travel distance. Each route starts/ends from/to 0, handling a subset of customers within the time limit T_{max} and refueling (even more than once) at AFSs. The GVRP is formally represented on a complete directed graph $G = (N, A)$, where the set of the nodes $N = I \bigcup F \bigcup \{0\}$ contains the set I of customers and the set F of AFSs while A indicates the arc set. For each $(i, j) \in A$, both the travel time and distance, t_{ij} and d_{ij}, are known. Moreover, for each $s \in F$ and for each $i \in I$, the refueling time p_s and the service time p_i are given. For each AFV, both the maximum fuel capacity Q and the average speed v are known. The fuel consumption is linearly proportional to the travel distance through the fuel consumption rate r. Due to the limited fuel capacity, the maximum distance D_{max} an AFV can travel without stopping at any AFS is given by Q/r. Each AFV is supposed to leave the depot fully refueled and to be fully refueled when it stops at an AFS. Finally, each AFS s is assumed to have an infinite capacity, i.e., there is no limit on the number of AFVs that can simultaneously refuel at s. But, really, the AFSs have a limited number of refueling docks. Neglecting this aspect, during the route planning, may yield long waiting times at the AFSs with a significant negative impact on the solution, especially when the refueling time is high (e.g., in the case of electric vehicles) and/or T_{max} is very tight. In these cases, omitting AFSs capacity may produce infeasible route plans. To overcome this issue, we introduce a new variant of the GVRP, i.e., the GVRP with Capacitated AFSs (GVRP-CAFS), in which only a limited number of AFVs can refuel at the same time at the same AFS. Without loss of generality, we assume that the capacity η_s of each AFS s is unitary. In fact, the case $\eta_s > 1$ can be reduced to the one with $\eta_s = 1$ by properly adding clones of s with unitary capacity. We solve the GVRP-CAFS through the exact approach described in Section 2.

2 An exact approach for the GVRP-CAFS

Each route in a GVRP solution can be seen as the composition of *paths*, each handling a subset of customers without intermediate stops at AFSs. Each path can be between: 0 and AFS, AFS and 0, two AFSs and finally, 0 and itself (complete route). Moreover, for each path k, the origin (starting node) s_k, the destination (arrival node) a_k, the travel distance d_k and the duration γ_k are known. In particular, γ_k is the sum of the travel times and the service times at the nodes of k. In the GVRP solution in Figure 1, where $C1, C2, C3, C4$ denote the customers, the route $\{0, C1, C2, C3, AFS2, C4, 0\}$ is the composition of the paths $\{0, C1, C2, C3, AFS2\}$ and $\{AFS2, C4, 0\}$.

The proposed solution approach is then based on two steps. Firstly, the set K of all

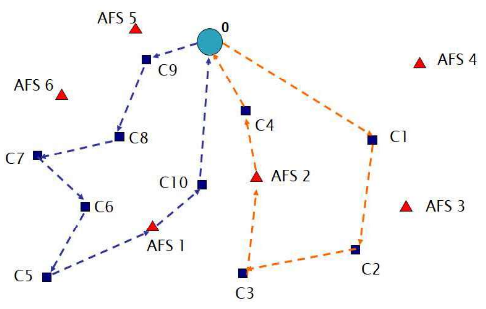


FIG. 1: A solution feasible for a GVRP.

feasible non-dominated paths is generated. A path k is feasible (*feasibility rules*) if: $d_k \leq D_{max}$ and $\gamma_k + t_{0s_k} + t_{a_k0} + p_{a_k} \leq T_{max}$. Moreover, a feasible path k_1 dominates a feasible path k_2 (*dominance rules*) if: $s_{k_1} = s_{k_2}$, $a_{k_1} = a_{k_2}$, they handle the same customers and $d_{k_2} \geq d_{k_1}$. From the set K, the set P of all the pairs of paths is generated. Given $k_1, k_2 \in K$, a pair (k_1, k_2) exists (*compatibility rules*) if: $a_{k_1} = s_{k_2}$, $s_{k_2} \neq 0$, the set of customers handled in the two paths are disjoint and $t_{0s_{k_1}} + \gamma_{k_1} + \gamma_{k_2} + p_{a_{k_1}} + t_{a_{k_2}0} \leq T_{max}$. In the second step, a path-based Mixed Integer Programming (MIP) model is used to select the paths and properly combine them to generate the routes of the optimal GVRP-CAFS solution. A coverage parameter, c_{ik}, is then introduced, equal to 1 if $i \in I$ is handled in $k \in K$, 0 otherwise. The refueling time p_{a_k} needed at a_k is equal to 0 if $a_k = 0$. The following decision variables are introduced: z_k, equal to 1 if $k \in K$ is selected, 0 otherwise; x_{kl}, equal to 1 if $l \in K$ is covered just after $k \in K$, 0 otherwise and finally, τ_k, a positive variable representing the starting refueling time at a_k of $k \in K$. The path-based MIP model is given in the following.

$$\min \sum_{k \in K} d_k z_k \tag{1}$$

$$\sum_{k \in K} c_{ik} z_k = 1 \quad \forall i \in I \tag{2}$$

$$\sum_{k \in K} x_{0k} \leq m \tag{3}$$

$$\sum_{k_1 \in K:(k_1,k_2) \in P} x_{k_1k_2} = \sum_{k_1 \in K:(k_2,k_1) \in P} x_{k_2k_1} \quad \forall k_2 \in K \tag{4}$$

$$\sum_{k_1 \in K:(k_1,k_2) \in P} x_{k_1k_2} = z_{k_2} \quad \forall k_2 \in K | k_2 \neq 0 \tag{5}$$

$$\tau_{k_2} \geq \tau_{k_1} + p_{a_{k_1}} + \gamma_{k_2} - T_{max} x_{k_1k_2} \quad \forall (k_1, k_2) \in P \tag{6}$$

$$|\tau_{k_1} - \tau_{k_2}| \geq p_{a_{k_1}} \quad \forall k_1, k_2 \in K | a_{k_1} = a_{k_2}, a_{k_1} \neq 0 \tag{7}$$

$$\gamma_k \leq \tau_k \leq T_{max} - p_{a_k} + T_{max}(1 - z_k) \quad \forall k \in K \tag{8}$$

$$z_k \in \{0, 1\} \quad \forall k \in K \tag{9}$$

$$x_{k_1k_2} \in \{0,1\} \quad \forall (k_1,k_2) \in P \tag{10}$$

The objective function (1) concerns the minimization of the total travel distance. Each customer has to be visited exactly once (2) and the number of routes selected does not exceed the number of available AFVs (3). Route continuity is ensured by constraints (4). A path can be inserted in a route only if it is selected (5). If $x_{k_1k_2} = 1$, τ_{k_2} cannot start before both the refueling operation at a_{k_1} is completed and k_2 is performed (6). Two AFVs cannot simultaneously refuel at the same AFS (7). The refueling of the AFV in the path k cannot start before the time necessary to travel the path and it cannot finish after T_{max} (8). Constraints (7) are linearized by (11)-(14), through auxiliary variables ξ and μ. In the Linear MIP (MILP) model, constraints (13) avoid overlapped refueling operations.

$$\xi_{k_1k_2} \geq -\frac{1}{T_{max} - p_{a_{k_1}}}(\tau_{k_1} - \tau_{k_2} - p_{a_{k_1}}) \quad \forall k_1, k_2 \in K | a_{k_1} = a_{k_2}, a_{k_1} \neq 0 \tag{11}$$

$$\mu_{k_1k_2} \geq -\frac{1}{T_{max} - p_{a_{k_1}}}(\tau_{k_2} - \tau_{k_1} - p_{a_{k_1}}) \quad \forall k_1, k_2 \in K | a_{k_1} = a_{k_2}, a_{k_1} \neq 0 \tag{12}$$

$$\xi_{k_1k_2} + \mu_{k_1k_2} \leq 3 - z_{k_1} - z_{k_2} \quad \forall k_1, k_2 \in K | a_{k_1} = a_{k_2}, a_{k_1} \neq 0 \tag{13}$$

$$\xi_{k_1k_2}, \mu_{k_1k_2} \in \{0,1\} \quad \forall k_1, k_2 \in K | a_{k_1} = a_{k_2}, a_{k_1} \neq 0 \tag{14}$$

For limiting the amount of time required by our approach, a relaxation of the GVRP-CAFS (RP) generated by omitting (11)-(14) is solved. On each iteration, we solve RP to optimality and check if any of those constraints are violated in the optimal RP solution. If this is the case, $\forall (k_1, k_2) \in P$ for which those constraints are violated, i.e., for which the related refueling operations overlap, we add to the RP the corresponding violated constraints (11)-(14) and we reiterate; otherwise, we stop because the current optimal RP solution is optimal for the GVRP-CAFS too. This approach allows us strongly limiting the number of constraints involved to address the capacity issue.

3 Results and conclusions

We introduced the GVRP-CAFS, a more realistic variant of the GVRP, in which the AFS capacity, i.e., the number of AFVs that can simultaneously refuel at the same AFS, is limited. For the GVRP-CAFS, we formulated a MILP model and we proposed an exact method to solve it in more reasonable time. Preliminary tests were carried out on a set of challenging instances with tight AFS capacity and on average 15 customers and 3 AFSs. The proposed exact approach solved to optimality all the instances within an average computational time of 22 seconds against an average of 557 seconds of the MILP model.

References

[1] Bektaş, T. and Demir, E. and Laporte, G. (2016), “Green vehicle routing”, *Green Transportation Logistics*, 243-265, Springer International Publishing.

[2] S. Erdoğan and E. Miller-Hooks, “A green vehicle routing problem”, *Transportation Research Part E: Logistics and Transportation Review* 48(1),100-114, 2012.

The Electric Vehicle Relocation Problem in Carsharing Systems with Collaborative Operators

Maurizio Bruglieri[1], Fabrizio Marinelli[2], Ornella Pisacane[2]

[1] Politecnico di Milano, Milano, Italy
maurizio.bruglieri@polimi.it

[2] Universitá Politecnica delle Marche, Ancona, Italy
{marinelli,pisacane}@dii.univpm.it

Abstract

We address the problem of balancing the demand and the availability of vehicles between stations in urban one-way electric carsharing systems through operator relocations. Unlike the previous papers, we assume that the operators can collaborate among them through the *carpooling*, i.e., giving a lift to the others when moving an EV from a pick-up request station to one of delivery. For this new problem, we propose a Mixed Integer Linear Programming formulation and a column generation based heuristic solution approach.

Keywords : *Mixed Integer Linear Programming, column generation, Pick-up and Delivery Problem with Time Windows, operator based relocation, one-way carsharing.*

1 Introduction

The carsharing systems allow users renting cars by paying a charge that depends on the actual time of use (also a fraction of an hour) eliminating the fixed costs due to both the ownership and the maintenance of the vehicles. However, the *one-way carsharing systems*, in which a user can deliver the vehicle to a station different from the one of pick-up, pose the management problem of balancing the demand and the availability of vehicles between the stations [4]. Moreover, when the carsharing fleet is made up of Electric Vehicles (EVs), the relocation is more complicated due to their recharge needs.

We address the operator-based EV relocation problem in urban one-way carsharing systems assuming that: the requests are known in advance (*exact predictive relocation*); the operators directly drive the EVs from stations with exceeding EVs (pick-up requests) to stations that need EVs (delivery requests); they move from the latter to the former by folding bicycles as introduced in [1]. A revenue is associated with each relocation request as well as a fixed cost with each operator used. The objective is to maximize the total profit given by the difference between the total revenue due to the requests satisfied and the total cost of the operators employed, as introduced in [2, 3].

Unlike the previous papers, where the operators do not interact with each other, we assume that they can collaborate among them through the *carpooling*, i.e., giving a lift to other operators when moving an EV from a pick-up request station to one of delivery. We assume that the lift is given with no intermediate stop, i.e., all the passengers can get out of the EV only at the driver's delivery station. We call this new version of the *Electric Vehicle Relocation Problem* (E-VReP), the *E-VReP with Collaborative Operators.*

For this problem, we propose a Mixed Integer Linear Programming (MILP) formulation and a column generation based solution method.

2 Statement of the problem and MILP formulation

Let L be the maximum distance a fully recharged EV can cover. When the EV is not fully recharged, such distance is supposed linearly proportional to its residual battery charge. Although the charging time function depends on the battery technology used, it is assumed to be linear and Γ is the time necessary for a full recharge. We assume that each parking station has a charger to which an EV is always connected when it is unused.

Let K be the number of operators available and C the cost associated with their employment. Let D and P be the set of delivery requests (i.e. EVs delivery to try to prevent a station from running out of EVs) and of pick-up requests (i.e, to try to prevent a station from being full of EVs), respectively. For each relocation request $r \in P \cup D$, the parking location v_r, the residual battery charge ρ_r, the earliest and the latest time allowed to carry out r, $[\tau_r^{min}, \tau_r^{max}]$, are known. We assume that the requests are not mandatory and a revenue π_r is obtained if the request r is satisfied. Since the carsharing fleet is supposed homogeneous, each request $r \in D$ can be satisfied bringing to v_r an EV of a pick-up request, compatible for both time window and the battery charge level.

We want both to route and to schedule the operators, leaving from a common depot (0), at two different times, t_0' and t_0'', e.g., corresponding to the start of the Morning Shift (MS) and of the Afternoon Shift (AS), in order to maximize the total profit. In each route, a request of pick-up is always alternated to a one of delivery. Moreover, since we assume that the operators can collaborate through the *carpooling* with no intermediate stop, their routes can share some ordered pairs of pick-up and delivery requests. The same pair can be shared in at most $\tilde{C}$ routes, being $\tilde{C}$ the capacity of an EV.

The problem is represented on a directed graph $G = (N, A)$ where $N = P \cup D \cup \{0\}$ and arc set A is the union of the arcs A^{EV} traveled by EV, and those of A^B traveled by bike. Arcs $(i, j) \in A^{EV}$, with $i \in P$ and $j \in D$, model the action of an operator that goes from a station of pick-up to one of delivery by EV, also possibly giving a lift to other operators. Arcs $(j, i) \in A^B$, with $j \in D$ and $i \in P$, model the action of an operator that moves from a station of delivery to one of pick-up by bike. For each $(i, j) \in A^{EV}$, d_{ij} is the length of the shortest path from v_i to v_j by EV, while c_{ij} is the corresponding operational time taking into account the time to load the bike in the EV trunk, to go from i to j by EV, to park the EV and to take the bike from the EV trunk. Instead, $\forall (i, j) \in A^B$, c_{ij} is the time to go from i to j by bike.

The problem is mathematically modeled by introducing the following decision variables: x_{ij}, number of operators traversing $(i, j) \in A$; y_{ij}, equal to 1 if $(i, j) \in A$ is traveled by at least one operator, 0 otherwise; t_i, the latest arrival time at $i \in N$ and ξ_r equal to 1 if $r \in P \cup D$ is handled in the MS, 0 if it is served in the AS.

$$\max \sum_{(i,j)\in A^{EV}} (\pi_i + \pi_j) y_{ij} - \sum_{j\in\delta^+(0)} C x_{0j} \tag{1}$$

$$\sum_{j\in\delta^+(0)} x_{0j} \le K \tag{2}$$

$$\sum_{j\in\delta^+(i)} y_{ij} \le 1 \qquad \forall i \in P \tag{3}$$

$$\sum_{i\in\delta^-(j)} y_{ij} \le 1 \qquad \forall j \in D \tag{4}$$

$$\sum_{j\in\delta^+(i)} x_{ij} - \sum_{j\in\delta^-(i)} x_{ji} = 0 \qquad \forall i \in N \tag{5}$$

$$x_{ij} \le \tilde{C} y_{ij} \qquad \forall (i, j) \in A \tag{6}$$

$$y_{ij} \le x_{ij} \qquad \forall (i, j) \in A \tag{7}$$

$$t'_0\xi_j + t''_0(1-\xi_j) + c_{0j}y_{0j} \leq t_j \quad \forall j \in \delta^+(0) \tag{8}$$
$$t_i + c_{ij}y_{ij} - T(1-y_{ij}) \leq t_j \quad \forall (i,j) \in A : i \neq 0, j \neq 0 \tag{9}$$
$$t_i + c_{i0}y_{i0} - t'_0\xi_i - t''_0(1-\xi_i) \leq T \quad \forall i \in \delta^-(0) \tag{10}$$
$$\xi_i - \xi_j \leq 1 - y_{ij} \quad \forall (i,j) \in A : i \neq 0, j \neq 0 \tag{11}$$
$$\xi_j - \xi_i \leq 1 - y_{ij} \quad \forall (i,j) \in A : i \neq 0, j \neq 0 \tag{12}$$
$$\tau_i^{min} \leq t_i \leq \tau_i^{max} \quad \forall i \in P \cup D \tag{13}$$
$$L(\rho_i + \frac{t_i - \tau_i^{min}}{\Gamma}) \geq d_{ij}y_{ij} \quad \forall (i,j) \in A^{EV} \tag{14}$$
$$\rho_i + \frac{t_i - \tau_i^{min}}{\Gamma} - \frac{d_{ij}}{L}y_{ij} \geq \rho_j - \frac{\tau_j^{max} - t_j}{\Gamma} - (\rho_j + 1)(1 - y_{ij}) \quad \forall (i,j) \in A^{EV} \tag{15}$$
$$1 - \frac{d_{ij}}{L}y_{ij} \geq \rho_j - \frac{\tau_j^{max} - t_j}{\Gamma} - (\rho_j + 1)(1 - y_{ij}) \quad \forall (i,j) \in A^{EV} \tag{16}$$
$$x_{i,j} \geq 0, integer, \quad y_{ij} \in \{0,1\}, \ \forall (i,j) \in A, \quad t_i \geq 0 \ \forall i \in N \tag{17}$$

where $\delta^-(i)$ and $\delta^+(i)$ denote the ingoing and outgoing arcs in/from $i \in N$, respectively. The objective function (1) represents the total profit to be maximized. Constraint (2) ensures that no more than K operators are employed. Constraints (3) avoid that the same picked up EV is used to satisfy more than one delivery request. Vice versa, constraints (4) avoid that more than one pick-up request are used to satisfy the same delivery request. Conditions (5) ensure the flow conservation on x.

The x variables are linked to the y ones in (6) and (7): these constraints ensure that if $y_{ij} = 1$, the arc $(i,j) \in A$ can be traversed by at most $\tilde{C}$ operators; otherwise, it cannot be traveled. In each route, the arrival times at both the first node visited and the next ones are ruled by constraints (8) and (9), respectively.

The total duration of a route cannot exceed T thanks to (10). Constraints (11) and (12) ensure that, if two requests are served in the same route, they are served in the same time shift too. The time window of each request $i \in P \cup D$ is imposed in conditions (13). The distance traveled by each EV is proportional to its residual battery level (14) and each EV is delivered satisfying the required battery level (15)-(16).

3 A Column Generation based heuristic

Since the model described in the previous section can solve in reasonable time only instances of few tens of requests through a state of the art MILP solver (CPLEX), we propose a different solution method based on column generation.

For this purpose, let Ω the set of all feasible routes for the E-VReP. The following *route-based* formulation models a relaxation of the original problem (1)-(17) since no synchronization constraint is imposed on the routes that share the same arcs.

It is based on the binary variables $\theta_\omega = 1$ if the route ω is chosen, 0 otherwise and on binary variables $y_{ij} = 1$ if arc $(i,j) \in A$ is chosen in at least one route, 0 otherwise:

$$max \sum_{(i,j) \in A^{EV}} (\pi_i + \pi_j)y_{ij} - C \sum_{\omega \in \Omega} \theta_\omega \tag{18}$$
$$\sum_{\omega \in \Omega} \theta_\omega \leq K \tag{19}$$
$$\sum_{\omega \in \Omega} a_{ij\omega}\theta_\omega \leq \tilde{C}y_{ij} \quad \forall (i,j) \in A \tag{20}$$
$$y_{ij} \leq \sum_{\omega \in \Omega} a_{ij\omega}\theta_\omega \quad \forall (i,j) \in A^{EV} \tag{21}$$
$$\sum_{j \in \delta^+(i)} y_{ij} \leq 1 \quad \forall i \in P \tag{22}$$

$$\sum_{i \in \delta^-(j)} y_{ij} \leq 1 \quad \forall j \in D \tag{23}$$

$$\theta_\omega \in \{0,1\} \quad \forall \omega \in \Omega, \quad y_{ij} \in \{0,1\} \;\; \forall (i,j) \in A \tag{24}$$

where the parameter $a_{ij\omega}$ is equal to 1 if the pair of requests $(i,j) \in A$ is served in route ω, 0 otherwise. Indeed constraints (19) guarantees that no more than K operators are used; constraints (20) ensure that no more than $\tilde{C}$ operators are carpooling along each arc, at the same time; while (21) guarantee the coherency between variables θ_ω and y_{ij}, i.e., if $y_{ij} = 1$ then at least one route ω containing the arc (i,j) must be selected; (22) ensure that with the vehicle picked up from i only one delivery request can be satisfied; vice versa, (23) guarantees that a delivery request j can only one be satisfied by one pick-up request; finally, (24) model the variables nature.

The continuous relaxation of the route-based formulation (18)-(24) is solved through a column generation approach. Then, an integer solution is heuristically detected by solving (18)-(24) restricted to the only "good" routes found (i.e., those selected along the column generation) and possibly adding other routes. However, such integer solution could not satisfy the synchronization constraints among the operators, relaxed in the formulation (18)-(24). This solution is then heuristically repaired according to the synchronization infeasibilities, through proper *forward* and/or *backward* shifts of the relocation request execution times.

We notice that in this procedure we have to guarantee not only that each route satisfies the time windows of the requests handled after the shifts, but we also have to carefully consider the battery charge levels. Indeed, if an EV is picked up too early then it may have not enough battery recharge to reach the next delivery request. Vice versa, if it is picked up too late then it may arrive to the delivery request just before the maximum allowed time window without the required battery level since there is not enough time to recharge it at the delivery station.

4 Results and conclusions

In this work, we extended the E-VReP problem concerning the relocation of electric vehicles in a carsharing system, allowing the operators to collaborate among them through the *carpooling*, i.e., giving a lift to other operators when moving an EV from a pick-up request station to one of delivery. The problem was formulated by MILP and solved in more efficient way through a column generation based heuristic. Preliminary results show that thanks to the collaboration among the operators not only it is possible to decrease the distance covered via bike by the operators, but sometimes also to increase the total profit.

References

[1] M. Bruglieri, A. Colorni, A. Luè, "The vehicle relocation problem for the one-way electric vehicle sharing", *Networks*, 64 (4), 292–305, 2014.

[2] M. Bruglieri, F. Pezzella, O. Pisacane, "Heuristic algorithms for the operator-based relocation problem in one-way electric carsharing systems", *Discrete Optimization*, 23, 56–80, 2017.

[3] M. Bruglieri, F. Pezzella, O. Pisacane, "An Adaptive Large Neighborhood Search for Relocating Vehicles in Electric Carsharing Services", *Discrete Applied Mathematics*, DOI: 10.1016/j.dam.2018.03.067, 2018.

[4] G. Laporte, F. Meunier, R., Wolfler Calvo, "Shared mobility systems", *4OR-Q J Operational Research*, 13:341–360, 2015.

Make or Buy: Revenue Maximization in Stackelberg Scheduling Games

Toni Böhnlein[1], Oliver Schaudt[2], Joachim Schauer[3]

[1] Universität zu Köln, Institut für Informatik, Weyertal 80, 50931 Köln,
boehnlein@zpr.uni-koeln.de

[2] RWTH Aachen, Institut für Mathematik, Pontdriesch 10, 52062 Aachen,
schaudt@mathc.rwth-aachen.de

[3] University of Graz, Department of Statistics and Operations Research, Universitätsstr. 15, 8010 Graz,
joachim.schauer@uni-graz.at

Abstract

In a Stackelberg pricing game a distinguished player, the *leader*, chooses prices for a set of items, and the other player, the *follower*, seeks to buy a minimal cost feasible subset of the items. The goal of the leader is to maximize her revenue, which is determined by the sold items and their prices. Typically, the follower is given by a combinatorial covering problem, e.g., his feasible subsets are the edges of a spanning tree or the edges of an s-t-path in a network.

We initiate the study of Stackelberg pricing games where the follower solves a maximization problem. In this model, the leader offers a payment to include her items in the follower's solution.

Our motivation stems from the following situation: assume the leader has a set of jobs $1, \dots, k$ to complete. A job i may either (a) be executed for a given cost $b(i)$ using her own resources or (b) offered to the follower at a variable price $p(i)$ to complete it for her. The objective function to be maximized by the leader is the sum of the margins $b(i) - p(i)$ over those jobs i that are completed by the follower. Informally, the question is which jobs should be outsourced and what profit the leader has to offer. Our main result says that the problem can be solved to optimality in polynomial-time when the jobs have fixed starting and terminating times and the follower solves a maximum weight scheduling on a single machine.

To show that the situation changes when the follower is given by other optimization problems, we prove APX-hardness for a scheduling problem that can be modeled as a bipartite maximum weight matching problem. Moreover, we show APX-hardness in the case of the maximum weight spanning tree problem.

On a more general note, we prove Σ_2^p-completeness if the follower has a general combinatorial optimization problem given in the form of a finite ground set and a feasibility oracle. This shows that while the follower's problem is NP-complete, the leader's problem is hard even if she has an NP-oracle at hand.

Keywords : *Algorithmic pricing, Stackelberg games, Revenue maximization.*

1 Introduction

Suppose an agent seeks to complete a set of jobs $1, \dots, k$. Job i may either (a) be executed for a given fixed cost $b(i)$ using the agent's own resource or (b) offered to a manufacturer at a variable price $p(i)$ to carry it out for him. If the manufacturer finishes an offered job i, the agent pays the price $p(i)$. The agent's objective is to maximize the sum of the margins $b(i) - p(i)$

over those jobs i that are finished by the manufacturer. Typically, this is called a *make or buy* decision.

Whether it is profitable to outsource a job or not, depends on the manufacturer's offer situation. The agent might have competitors who also offer a payment to the manufacturer to carry out their jobs. Moreover, the manufacturer's schedule has to obey a number of constraints. For instance, it might be impossible to execute two jobs in the same time window since the manufacturer only has one machine available.

When setting the prices, the agent is aware of the competitors' jobs, the constraints they imply and the offered payments. After prices are set, the manufacturer selects a feasible subset of all jobs offered by the agent and her competitors. His objective is to maximize the income. We study the problem of computing prices that are optimal for the leader, for different types of constraints of the manufacturer's schedule.

This class of pricing problems features a hierarchical dependency. First, the agent sets prices; then the manufacturer selects a set of jobs. In the literature, such problems are captured by a game-theoretic model called *Stackelberg Pricing Games*. Originally, in a Stackelberg Pricing Game one player chooses prices for a number of items. After that, one or several other players are interested in buying these items. Following the standard terminology, the player to choose the prices is called the *leader* while the other players are called *followers*. The goal of the leader is to maximize her revenue while followers want to minimize their costs. Depending on the follower's preferences, computing optimal prices can be a highly non-trivial problem.

A major line of research studies Stackelberg Pricing Games where the follower's preferences are given by a combinatorial optimization problem. Labbé et al. [5] model road-toll setting problems by a Stackelberg Pricing Game based on the shortest path problem. In this game, the leader sets prices for a subset of priceable edges of a network graph while the remaining edges have fixed costs. Each follower has a pair of vertices (s, t) and buys a minimum cost path from s to t. The cost of a path depends on both the fixed cost and the prices set by the leader. Roche et al. [7] show that the problem is NP-hard, even if there is only one follower, and it has later been shown to be APX-hard [1, 4]. More recently, other combinatorial optimization problems were studied in their Stackelberg Pricing Game version. For example, Cardinal et al. [2, 3] investigate the Stackelberg Minimum Spanning Tree Game, proving APX-hardness and giving some approximation results. Our contribution is a model to capture scenarios where the follower solves a maximization problem.

To model the make or buy problem sketched above, we introduce a Stackelberg Pricing Game that is based on the well-known *Interval Scheduling Problem*. In this problem, there is one machine and a set of weighted jobs I. Each job i has a fixed starting time $s_i \in \mathbb{R}$ and terminating time $t_i \in \mathbb{R}$. Hence, a job can be represented by an interval $[s_i, t_i]$ on the line. We say that two intervals *overlap* if their intersection is non-empty. The objective is to find a subset of non-overlapping intervals of maximum total weight. On the left-hand side of Figure 1 we have an instance of an interval scheduling problem if we only consider the solid intervals a, b, c, d. An optimal solution is the set $\{a, b\}$ with a total weight of 7. According to every algorithms textbook, this problem can be solved efficiently via dynamic programming.

We call the agent leader and the manufacturer follower. In our Stackelberg Pricing Game, the solid intervals a, b, c, d represent the jobs of the competitors. The dashed lines x, z, y are the jobs of the leader with their respective costs $4, 3, 5$. First, the leader has to set prices p_x, p_y and p_z. Higher prices are more appealing to the follower. However, lower prices are more profitable for the leader. On the right hand side of Figure 1 the prices $p_x = 3$, $p_y = 0$, $p_z = 4$ are set. The follower selects the jobs c, x, z with total weight 9. Note that the prices p_x and p_z are optimal in the following sense: if either of p_x or p_z is decreased by some ε, the intervals x or z are not selected by the follower. The leader obtains a margin of 2 under these prices.

The optimal prices $p_x = 1$, $p_y = 2$, $p_z = 0$ yield margin 4. Under these prices the solutions $\{b, c, x, y\}$ and $\{a, b\}$ are optimal for the follower; both have a weight of 7. A common assumption for Stackelberg Pricing Games is that the follower is cooperative: he always chooses the optimal solution which is most profitable for the leader.

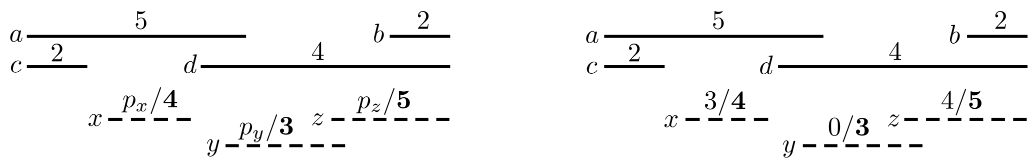


FIG. 1: An instance of the Stackelberg Interval Scheduling Game

Another scheduling problem that fits into our scenario can be formalized as a matching problem. Here, the follower has m machines, and there are n jobs. Each machine can execute at most one job in total. This situation can be modeled by a bipartite graph $G = (U \cup V, E)$. The vertices in U correspond to the m machines and the vertices in V to the n jobs. If the follower receives a payment to execute job i on machine j, there is an edge connecting the respective vertices with the payment as its weight. Hence, the follower solves a maximum weight matching problem to maximize his income. We extend this to a Stackelberg Pricing Game, as follows. Say k of the n jobs belong to the leader. An edge e which is incident to the corresponding vertices is a priceable edge and has cost $b(e)$. The leader's objective is to set prices $p(e)$ that maximize the sum of the margins $p(e) - b(e)$ over the priceable edges that are part of a maximum weight matching computed by the follower.

Our results. To make things more formally, we use a slightly different notion as in the introductory scenario. We say the leader receives a *benefit* $b(i)$ if job i is executed by the follower. The leader's objective is to maximize her *revenue*, the sum of the margins $b(i) - p(i)$ over the jobs i executed by the follower.

Our main result is a polynomial time algorithm that solves the Stackelberg Interval Scheduling Game. Since it will be more handy later on, we restate this game in terms of an independent set problem in an interval graph. Associated with an instance of the interval scheduling problem I we construct the corresponding *interval graph* $G = (V, E)$ with vertex weights. For each interval $i \in I$ there is a corresponding vertex $v_i \in V$ with weight $w(v_i) = w(i)$. If two intervals i, j overlap, there is an edge $(v_i, v_j) \in E$.

Recall that an *independent set* in a graph is a subset of mutually non-adjacent vertices. The interval scheduling problem on I is equivalent to finding a maximum weight independent set of G.

Stackelberg Interval Scheduling (SIS)
Input: An interval graph $G = (V, E)$ with priceable vertices $P \subseteq V$ and $|P| = k$. For every $v \in P$ there is a benefit $b(v) \in \mathbb{R}$ and for every $u \in F = V \setminus P$ there is a fixed weight $w(u) \in \mathbb{R}$.
Objective: Find a price function $p : P \to \mathbb{R}_{\geq 0}$ maximizing

$$\max\{b(S \cap P) - p(S \cap P) \mid S \text{ is a maximum weight independent set}\},$$

where the weight of an independent set S is defined as $w(S \cap F) + p(S \cap P)$.

The objective might seem a bit odd. But the formulation correctly reflects the cooperative behavior of the follower. The assumption that the follower picks a solution maximizing the leader's revenue is common for such pricing games and made to avoid technicalities (cf. [6]).

The main result of our paper reads as follows.

Theorem 1 *The SIS problem can be solved in time $\mathcal{O}(k^3\,(|V(G)| + |E(G)|))$ given an instance (G, P, w, b) with $|P| = k$.*

The proof builds on a careful analysis of the structure of an optimal price function. It relies on a linear programming formulation of a restricted version of the SIS problem. Based on these insights we use dynamic programming to solve the full problem. We remark that our theorem is one of the few cases in which one can solve a Stackelberg Pricing Game based on a non-trivial optimization problem to optimality.

As a complementary result, we show that the second scheduling problem mentioned above–the one that is based on a matching problem–is APX-hard. We formulate the corresponding Stackelberg Pricing Game for general graphs.

> Stackelberg Matching Game
> **Input:** A graph $G = (V, E)$ with priceable edges $P \subseteq E$ where $k = |P|$. For every $e \in P$ there is a benefit $b(e) \in \mathbb{R}$ and for every $e \in F = E \setminus P$ there are fixed weights $w(e) \in \mathbb{R}$.
> **Objective:** Find a price function $p : P \to \mathbb{R}_{\geq 0}$ maximizing
>
> $$\max\{b(M \cap P) - p(M \cap P) \mid M \text{ is a maximum weight matching}\},$$
>
> where the weight of a matching M is defined as $w(M \cap F) + p(M \cap P)$.

The main result is that this problem is hard to approximate, unless P=NP.

Theorem 2 *The* Stackelberg Matching Game *is APX-hard even when the graph G is bipartite, $w(f) \in \{1, 2\}$ for all $f \in F$, and $b(e) = 4$ for all $e \in P$.*

As we will see in the proof of the theorem, the graph used for the reduction covers the model of the second scheduling problem. Therefore, it belongs to the class of APX-hard problems. Moreover, the theorem shows that the Stackelberg Interval Scheduling Game on perfect graphs–instead of interval graphs–is APX-hard. The matching problem on a graph is equivalent to the independent set problem on its line graph. Line graphs of bipartite graphs are perfect graphs.

We can also show that a maximization version of the Stackelberg Pricing Game for spanning trees is APX-hard. This follows readily with the proof by Cardinal et al. [2] for the minimization version. Yet, it shows that a Stackelberg Pricing Game based on a maximization problem is hard if the follower optimizes over a matroid. This theorem and its proof are not presented in the extended abstract.

Finally, we take a step back from our example problems and study the complexity of the Stackelberg Pricing Game in its own right. It turns out that there are combinatorial optimization problems in NP such that the corresponding Stackelberg Pricing Problem is Σ_2^p complete. In other words, such a pricing problem is computationally difficult even if an NP oracle is provided, unless the polynomial hierarchy collapses to the second level.

References

[1] Patrick Briest, Parinya Chalermsook, Sanjeev Khanna, Bundit Laekhanukit, and Danupon Nanongkai. Improved Hardness of Approximation for Stackelberg Shortest-Path Pricing. In *Internet and Network Economics*, pages 444–454. Springer, 2010.

[2] J. Cardinal, E.D. Demaine, S. Fiorini, G. Joret, S. Langerman, I. Newman, and O. Weimann. The Stackelberg Minimum Spanning Tree Game. *Algorithmica*, 59:129–144, 2011.

[3] Jean Cardinal, Erik D Demaine, Samuel Fiorini, Gwenaël Joret, Ilan Newman, and Oren Weimann. The Stackelberg Minimum Spanning Tree Game on Planar and Bounded-Treewidth Graphs. *Journal of combinatorial optimization*, 25(1):19–46, 2013.

[4] Gwenaël Joret. Stackelberg Network Pricing is Hard to Approximate. *Networks*, 57(2):117–120, 2011.

[5] M. Labbé, P. Marcotte, and G. Savard. A Bilevel Model of Taxation and Its Application to Optimal Highway Pricing. *Management Science*, 44:1608–1622, 1998.

[6] Martine Labbé and Alessia Violin. Bilevel programming and price setting problems. *Annals OR*, 240(1):141–169, 2016.

[7] S. Roche, G. Savard, and P. Marcotte. An approximation algorithm for Stackelberg network pricing. *Networks*, 46:57–67, 2005.

Two Stackelberg Knapsack games

Gaia Nicosia[1], Andrea Pacifici[2], Ulrich Pferschy[3], Joachim Schauer[3]

[1] Università degli studi "Roma Tre"
`nicosia@ing.uniroma3.it`
[2] Università degli studi di Roma "Tor Vergata"
`andrea.pacifici@uniroma2.it`
[3] University of Graz
`{ulrich.pferschy, joachim.schauer}@uni-graz.at`

Abstract

We consider a bilevel decision problem, namely a Stackelberg strategic game in a knapsack setting. One player, the leader $\mathcal{L}$, may control the weights or profits of a subset of all the items (the $\mathcal{L}$-items). The second player, the follower, selects—according to a publicly known strategy—a solution consisting of a subset of all items, in order to utilize a bounded resource and to maximize the overall profit. The leader obtains a payoff from the $\mathcal{L}$-items included in the solution so it may alter the parameters in order to maximize its revenues.

Two variants of the problem are considered, depending on whether the leader is able to control (i) the weights of its items or (ii) their profits. For each version of the problem we analyze the leader's problem for three natural strategies of the follower and discuss the complexity of the corresponding problems.

Keywords : *Stackelberg game, Knapsack problem, Dynamic programming*

1 Introduction

A Stackelberg game (named after the market model [8] due to Heinrich Freiherr von Stackelberg) is a strategic game in which there are two interacting players at two distinct levels. First, one player, called the *leader* $\mathcal{L}$, makes its choice by choosing some elements or by setting certain parameters. Then, in view of the leader's decision, the other player $\mathcal{F}$, called the *follower*, chooses its response. Both players aim at optimizing their own objectives, which are usually conflicting or at least not positively correlated. To this purpose the leader needs to anticipate the optimal response of the follower. In this setting, one usually assumes that the players have complete and mutual knowledge about each other's models.

Stackelberg games may be viewed as special bilevel programming problems (BP), that is, optimization problems in which some of the decision variables in an upper level problem must be optimal to some other, lower level problem. BP is, in general, computationally hard to solve: Jeroslow [5] showed that this problem is $\mathcal{NP}$-hard even when the objective and constraints are linear functions.

In the Stackelberg context, the upper level optimization problem is commonly referred to as the *leader's problem*, while the lower level one as the *follower's problem*. For Stackelberg games it is frequently assumed in the literature that the follower has polynomially bounded computation power for deriving its optimal solution, see, for instance, [1, 2].

In this work we analyze a Stackelberg-type leader-follower scenario for the classical binary *knapsack problem* (KP). In KP we are given a discrete finite set N of items, each having a positive integer weight and a profit, and a knapsack with bounded weight-capacity c. One asks for a subset of items with maximum total profit and total weight not exceeding the capacity of the knapsack (a comprehensive collection of results on KP can be found in [6]). We also provide additional results concerning the *sub set sum* case in which weight and profit of an item are equal, which corresponds to the well-known *subset sum problem* (SSP) [7]. Besides its importance in a huge number of both theoretical and practical applications, KP has been also investigated under a game theoretic perspective, as witnessed by several studies (see e.g. [4].)

The following leader-follower situation is considered: A given subset of items $L \subset N$ is controlled by the leader $\mathcal{L}$, who may alter the parameters (either weights or profits) of all items in L. After $\mathcal{L}$ has fixed these parameters, the follower $\mathcal{F}$ computes a solution set S of the knapsack problem over all items in N aiming at the maximization of the total profit, irrespective of whether an item is in L or not. The leader receives a payoff only from its own items which are included in the solution, i.e., from items in $S \cap L$. Thus, $\mathcal{L}$ has a strong incentive to choose the parameters of its items so that these are more likely to be included in the solution computed by $\mathcal{F}$. However, the payoff of an item in L for the leader is inversely correlated to its chances of being included in the solution set.

A related problem has been studied in [1], where the authors address a Min-Knapsack game in which the follower seeks to purchase a min-cost selection of objects of some bounded (from below) weight. They show that when $\mathcal{F}$ uses a greedy 2-approximation algorithm the problem is strongly $\mathcal{NP}$-hard but admits a polynomial-time $(2 + \varepsilon)$-approximation algorithm for the leader's revenue maximization problem. An overview and complexity results for various bilevel knapsack problems are given in [3].

In this work, we consider two variants of our Stackelberg Knapsack Problem where the leader controls (and receives a payoff associated to) either the weights (SKPW) or the profits (SKPP) of the items included by $\mathcal{F}$ in the solution and, moreover, we also address their corresponding subset sum variantes SSPW and SSPP[1].

In the first version, SKPW, $\mathcal{L}$ may set the weights of the items in L. Its payoff is the total weight of the items in $S \cap L$. Thus, the larger $\mathcal{L}$ sets the weights, the larger its potential payoff, but the smaller the chances that an item is included in S. This setting corresponds to the practical situation where the follower has a fixed investment budget (knapsack capacity) for buying assets and wants to maximize the total utility or profit gained from these assets. The leader is offering certain assets with a given utility and wants to maximize its total revenue, i.e. the total price (that is the weight) paid by the follower for its assets.

In the second version, SKPP, $\mathcal{L}$ can set new profits for the items in L. $\mathcal{F}$ computes its solution based on the new profit values while $\mathcal{L}$ receives as payoff the difference between original and new profit values. If $\mathcal{L}$ chooses small new profits, it would gain large profit differences, but its items would be less likely to be included in S, and vice-versa. In this case, the new profits represent a fee or price, which the leader is offering to the follower as a reward for including an item of L in S. These values compete with the profits given for the other items in $N \setminus L$. Naturally, for any item of L the fee offered to $\mathcal{F}$ reduces the profit to be gained by $\mathcal{L}$.

Clearly, in both variants of the addressed problem, the optimization task faced by the follower would require the solution of an $\mathcal{NP}$-hard knapsack problem. As pointed out above, it is frequently assumed in the literature that the follower can only apply polynomially bounded algorithms. Thus, we restrict the selection procedure of $\mathcal{F}$ to natural, possibly suboptimal, greedy-type strategies, which are also known to $\mathcal{L}$. In particular we study three possible strategies adopted by the follower.

1. The classical Greedy algorithm considers the items in non-increasing order of profit-weight ratio and packs each item into the knapsack if the remaining capacity suffices to do so.

2. If $\mathcal{F}$ is allowed to solve the LP-relaxation of the underlying knapsack problem, we are not considering an integer problem anymore but the follower may split items in

[1]In the subset sum problem an item's profit equals its weight, hence the two variants SSPW and SSPP differ as that parameter of the $\mathcal{L}$-items may be altered only in the knapsack constraint or in the objective function, respectively.

its solution set. It is well-known that the LP-relaxation can be computed by starting the Greedy algorithm and packing a fractional part of the first item which does not fit completely. This is called the *split item*.

3. The Greedy-split algorithm reports an integer solution by simply taking the LP-relaxation and removing any split item ("rounding down" the LP-solution).

2 Our contributions

The main results of our contribution are summarized hereafter.

- When the follower $\mathcal{F}$ adopts Greedy, both variants of the leader's problem SKWP and SKPP cannot admit a polynomial time approximation algorithm with a constant approximation ratio (unless $\mathcal{P} = \mathcal{NP}$).
- Both the previous negative results still hold for the special subset sum variants SSPW and SSPP.
- For SSPW and SSPP, we may however devise solution algorithms based on dynamic programming and running in $\mathcal{O}(n^{3/2}c)$ and $\mathcal{O}(nc^2)$, respectively.
- If the follower selects the solution set using the LP-relax strategy, it is possible for the leader to optimally select weights or profits (i.e., in both the variants of the problem) in polynomial time. The complexity, for the corresponding subset sum variants becomes, in this case, linear.
- Surprisingly enough, the computational complexity of SKPW and SKPP is different when $\mathcal{F}$ follows Greedy-split. In this case, the weight selection problem is polynomially solvable. Conversely, setting the optimal profits value for the leader is $\mathcal{NP}$-hard.
- While we are not aware of an ILP formulation when the solution set is selected by the follower $\mathcal{F}$ with an exact algorithm, for all the above cases, we may provide (mixed integer) linear programming formulations.

References

[1] P. Briest, M. Hoefer, L. Gualà, and C. Ventre. *On Stackelberg Pricing with Computationally Bounded Consumers.* Networks, 60(1):31–44 , 2012.

[2] P. Briest, M. Hoefer, and P. Krysta. *Stackelberg Network Pricing Games.* Algorithmica, 62:733–753, 2012.

[3] A. Caprara, M. Carvalho, A. Lodi, and G.J. Woeginger. *A study on the computational complexity of the bilevel knapsack problem.* SIAM Journal on Optimization, 24(2):823–838, 2014.

[4] L. Ensthaler and T. Giebe. *Bayesian optimal knapsack procurement.* European Journal of Operational Research, 234(3):774–779, 2014.

[5] R.G. Jeroslow. *The polynomial hierarchy and a simple model for competitive analysis.* Mathematical Programming, 32(2):146–164, 1985.

[6] H. Kellerer, U.Pferschy, and D. Pisinger. *Knapsack Problems*, Springer, 2004.

[7] U. Pferschy, G. Nicosia, and A. Pacifici. *On a Stackelberg Subset Sum Game.* CoRR abs/1801.03698, 2018.

[8] H.F. von Stackelberg. *Marktform und Gleichgewicht (Market and Equilibrium).* Verlag von Julius Springer, 1934.