Proceedings of the 17th Cologne-Twente Workshop on Graphs and Combinatorial Optimization

Editors:

Johann Hurink Stefan Klootwijk Bodo Manthey
Victor Reijnders Martijn Schoot Uiterkamp

Enschede, Netherlands, July 1–3, 2019

Editors

Johann Hurink
Stefan Klootwijk
Bodo Manthey
Victor Reijnders
Martijn Schoot Uiterkamp

CTW 2019
Proceedings of the 17th Cologne-Twente Workshop on Graphs and Combinatorial Optimization
J.L. Hurink, S. Klootwijk, B. Manthey, V.M.J.J. Reijnders, M.H.H. Schoot Uiterkamp (eds.)
Enschede, University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science
1–3 July 2019

ISSN 2590-0870
DSI Workshop Proceedings Series (online) WP19-01
`https://www.utwente.nl/en/digital-society/`

UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

17th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2019)

CTW 2019 takes place at the University of Twente, Enschede, Netherlands, from July 1 to July 3, 2019.

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

As it was the case with previous CTWs, we will edit a special edition of *Discrete Applied Mathematics* for CTW 2019. Hereby, we invite all participants to submit full-length papers related to the topics of the workshop.

List of Abstracts

Succinct Data Structures for Families of Interval Graphs

Hüseyin Acan[1], Sankardeep Chakraborty[2], Seungbum Jo[3], and Srinivasa Rao Satti[4]

[1]Drexel University, USA
[2]RIKEN Center for Advanced Intelligence Project, Japan
[3]University of Siegen, Germany
[4]Seoul National University, South Korea

Abstract

We consider the problem of designing succinct data structures for *interval graphs* with n vertices while supporting degree, adjacency, neighborhood and shortest path queries in optimal time. Towards showing succinctness, we first show that at least $n \log n - 2n \log \log n - O(n)$ bits[1]. are necessary to represent any unlabeled interval graph G with n vertices, answering an open problem of Yang and Pippenger [Proc. Amer. Math. Soc. 2017]. This is augmented by a data structure of size $n \log n + O(n)$ bits while supporting not only the above queries optimally but also capable of executing various combinatorial algorithms (like proper coloring, maximum independent set etc.) on interval graphs efficiently. Finally, we extend our ideas to other variants of interval graphs, for example, *proper/unit, k-improper interval graphs, and circular-arc graphs*, and design succinct data structures for these graph classes as well along with supporting queries on them efficiently.

1 Introduction

A simple undirected graph G is called an *interval graph* if its vertices can be assigned to intervals on the real line so that two vertices are adjacent in G if and only if their assigned intervals intersect. The set of intervals assigned to the vertices of G is called a *realization* of G. These graphs were first introduced by Hajós [5] who also asked for the characterization of them. The same problem was also asked, independently, by Benser [2] while studying the structure of genes. Interval graphs naturally appear in a variety of contexts, for example, operations research and scheduling theory, biology especially in physical mapping of DNA, temporal reasoning and many more. We refer the reader to [4] for a thorough treatment of interval graphs and its applications. Eventually answering the question of Hajós [5], several researchers came up with different characterizations of interval graphs, including linear time algorithms for recognizing them; see, for example, [4, Chapter 8] for characterizations, and linear time algorithms. Moreover, exploiting the special structure of interval graphs, many otherwise NP-hard problems in general graphs are also shown to have polynomial time algorithms for interval graphs [4]. These include computing maximum independent set, reporting a proper coloring, returning a maximum clique etc. In spite of having many applications in practically motivated problems, we are not aware of, to the best of our knowledge, any study of interval graphs from the point of view of *succinct data structures* where the goal is to store a set Z of objects using the information theoretic minimum $\log(|Z|) + o(\log(|Z|))$ bits of space while still being able to support the relevant set of queries efficiently, and which is what we focus on in this paper. We also assume the usual model of computation, namely a $\Theta(\log n)$-bit word RAM model where n is the size of the input.

[1]throughout the paper, we use log to denote the logarithm to the base 2

1.1 Our main Results

Given an unlabeled interval graph G with n vertices, in Section 2 we first show that at least $n \log n - 2n \log \log n - O(n)$ bits are necessary to represent G, answering an open problem of Yang and Pippenger [7]. More specifically, Yang and Pippenger [7] showed a lower bound of $(n \log n)/3 + O(n)$-bit for representing any unlabeled interval graph and asked whether this lower bound can be further improved. Augmenting this lower bound, in Section 3 we also propose a succinct representation of G using $n \log n + O(n)$ bits while still being able to support the relevant queries optimally, where the queries are defined as follows. For any two vertices $u, v \in G$,

- $\mathsf{degree}(v)$: returns the number of vertices that are adjacent to v in G,
- $\mathsf{adjacent}(u, v)$: returns true if u and v are adjacent in G, and false otherwise,
- $\mathsf{neighborhood}(v)$: returns all the vertices that are adjacent to v in G, and
- $\mathsf{spath}(u, v)$: returns the shortest path between u and v in G.

We show that all these queries can be supported optimally using our succinct data structure for interval graphs. More precisely, for any two vertices $v, u \in G$, we can answer $\mathsf{degree}(v)$ and $\mathsf{adjacent}(u, v)$ queries in $O(1)$ time, $\mathsf{neighborhood}(v)$ queries in $O(\mathsf{degree}(v))$ time, and $\mathsf{spath}(u, v)$ queries in $O(|\mathsf{spath}(u, v)|)$ time. Furthermore, we also show how one can implement various fundamental graph algorithms in interval graphs, for example depth-first search (DFS), breadth-first search (BFS), computing maximum independent set, determining a maximum clique etc, both time and space efficiently using our succinct representation for interval graphs. We also extend our ideas to other variants of interval graphs, for example, *proper/unit interval graphs, k-proper and k-improper interval graphs, and circular-arc graphs*, and design succinct data structures for these graph classes as well along with supporting queries on them efficiently.

2 Counting the number of unlabeled interval graphs

This section deals with counting unlabeled interval graphs on n vertices, and let $\mathcal{I}_n$ denote this quantity. Initial values of this quantity are given by Hanlon [6] but he did not prove an asymptotic form for enumerating the sequence. Answering a question posed by Hanlon [6], Yang and Pippenger [7] proved that the generating function $\mathcal{I}(x) = \sum_{n \geq 1} \mathcal{I}_n x^n$ diverges for any $x \neq 0$ and they established the bounds

$$\frac{n \log n}{3} + O(n) \leq \log \mathcal{I}_n \leq n \log n + O(n). \tag{1}$$

The upper bound in (1) follows from $\mathcal{I}_n \leq (2n-1)!! = \prod_{j=1}^{n} (2j-1)$, where the right hand side is the number of matchings on $2n$ points on a line. For the lower bound, the authors showed $\mathcal{I}_{3k} \geq k!/3^{3k}$ by finding an injection from S_k, the set of permutations of length k, to three-colored interval graphs of size $3k$. Furthermore, they left it open whether the leading terms of the lower and upper bounds in (1) can be matched, which is what show in affirmative by improving the lower bound. In other words, we find the asymptotic value of $\log \mathcal{I}_n$. In what follows, for a set S, we denote by $\binom{S}{k}$ the set of k-subsets of S.

Theorem 1. *Let $\mathcal{I}_n$ be the number of unlabeled interval graphs with n vertices. As $n \to \infty$, we have*

$$\log \mathcal{I}_n \geq n \log n - 2n \log \log n - O(n). \tag{2}$$

Proof. We consider certain interval graphs on n vertices with colored vertices. Let k be a positive integer smaller than $n/2$ and ε a positive constant smaller than $1/2$. For $1 \le j \le k$, let B_j and R_j denote the intervals $[-j-\varepsilon, -j+\varepsilon]$ and $[j-\varepsilon, j+\varepsilon]$, respectively. These $2k$ pairwise-disjoint intervals will make up $2k$ vertices in the graphs we consider. Now let $\mathcal{W}$ denote the set of k^2 closed intervals with one endpoint in $\{-k, \ldots, -1\}$ and the other in $\{1, \ldots, k\}$. We color $B_1, \ldots, B_k$ with blue, $R_1, \ldots, R_k$ with red, and the k^2 intervals in $\mathcal{W}$ with white.

Together with $\mathcal{S} := \{B_1, \ldots, B_k, R_1, \ldots, R_k\}$, each $\{J_1, \ldots, J_{n-2k}\} \in \binom{\mathcal{W}}{n-2k}$ gives an n-vertex, three-colored interval graph. For a given $\mathcal{J} = \{J_1, \ldots, J_{n-2k}\}$, let $G_{\mathcal{J}}$ denote the colored interval graph whose vertices correspond to n intervals in $\mathcal{S} \cup \mathcal{J}$, and let $\mathcal{G}$ denote the set of all $G_{\mathcal{J}}$.

Now let $G \in \mathcal{G}$. For a white vertex $w \in G$, the pair $(d_B(w), d_R(w))$, which represents the numbers of blue and red neighbors of w, uniquely determine the interval corresponding to w; this is the interval $[-d_B(w), d_R(w)]$. In other words, $\mathcal{J}$ can be recovered from $G_{\mathcal{J}}$ uniquely. Thus $|\mathcal{G}| = \binom{k^2}{n-2k}$. Since there are at most 3^n ways to color the vertices of an interval graph with blue, red, and white, we have

$$\mathcal{I}_n \cdot 3^n \ge |\mathcal{G}| = \binom{k^2}{n-2k} \ge \left(\frac{k^2}{n-2k}\right)^{n-2k} \ge \left(\frac{k^2}{n}\right)^{n-2k}$$

for any $k < n/2$. Setting $k = \lfloor n/\log n \rfloor$ and taking the logarithms, we get

$$\log \mathcal{I}_n \ge (n-2k)\log(k^2/n) - O(n) = n\log n - 2n\log\log n - O(n).$$

□

3 Succinct representation of interval graphs

In this section, we introduce a succinct $n\log n + (2+\epsilon)n + o(n)$-bit representation of unlabeled interval graph G on n vertices with constant $\epsilon > 0$, and show that the navigational queries (degree, adjacent, neighborhood, and spath queries) and some basic graph algorithms (BFS, DFS, PEO traversals, proper coloring, computing the size of maximum clique and maximum independent set etc.) on G can be answered/executed efficiently using our representation of G.

3.1 Succinct Representation of G

We first label the vertices of G using the integers from 1 to n, as described in the following. It's a well-known result that the vertices in G can be represented by n intervals $I = \{I_1 = [l_1, r_1], I_2 = [l_2, r_2], \ldots, I_n = [l_n, r_n]\}$ where all the endpoints in I are distinct integers in the range $[1, 2n]$. Since there are $2n$ distinct endpoints for the n intervals in I, every integer in $[1, 2n]$ corresponds to a unique l_i or r_i for some $1 \le i \le n$. We assign the labels to the vertices in G based on the sorted order of left endpoints of their corresponding intervals, i.e., for any two vertices $a, b \in G$, $a < b$ if and only if $l_a < l_b$. Now we describe the representation of G. Let $S = s_1 \ldots s_{2n}$ be the binary sequence of length $2n$ such that for $1 \le i \le 2n$, $s_i = 0$ if $i \in \{l_1, l_2, \ldots, l_n\}$ (i.e., if i corresponds to the left end point of an interval in I), and $s_i = 1$ otherwise. If $i = l_k$ or $i = r_k$, we say that s_i corresponds to the interval I_k. We represent the sequence S using $2n + o(n)$ bits to support rank and select queries on S in $O(1)$ time [3]. Next, we store the sequence $r = r_1 \ldots r_n$, and for some fixed constant $\epsilon > 0$, we also store an ϵn-bit data structure to support RMax and RMin queries on r in $O(1)$ time. Using the representations of S and r, it is easy to show that for any vertex $v \in G$, we can return its corresponding interval $I_v = [l_v, r_v]$ in $O(1)$ time by computing $l_v = \mathsf{select}_0(S, v)$, and r_v can be accessed from the sequence r. Thus, the total space usage of our representation is $n\log n + (3+\epsilon)n + o(n)$ bits. See Figure 1 for an example.

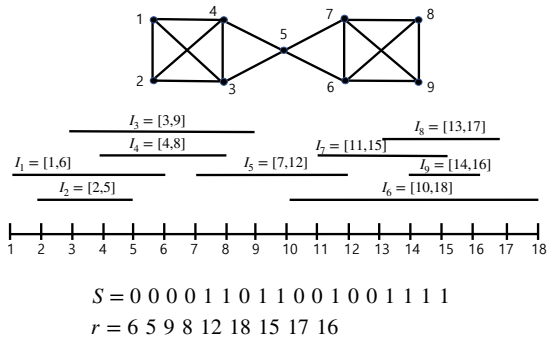


Figure 1: Example of an input interval graph and its representation.

3.2 Supporting Navigational Queries

In this section, we show that degree, adjacent, neighborhood, and spath queries on G can be answered in asymptotically optimal time using the representation described in the Section 3.1.

degree(v) query: We count the number of vertices in G which are not adjacent to v, which is a disjoint union of the two sets: (i) the set of intervals that end before the starting point l_v, and (ii) the set of intervals that start after the end point r_v. Using our representation the cardinalities of these two sets can be computed as follows. The number of intervals u with $r_u < l_v$ is given by $\mathsf{rank}_1(S, l_v)$. Similarly, the number of intervals u with $r_v < l_u$ is given by $n - \mathsf{rank}_0(S, r_v)$. Therefore, we can answer $\mathsf{degree}(v)$ query in $O(1)$ time by returning $n - \mathsf{rank}_1(S, l_v) - (n - \mathsf{rank}_0(S, r_v)) = \mathsf{rank}_0(S, r_v) - \mathsf{rank}_1(S, l_v)$.

adjacent(u, v) query: Since we can compute the intervals I_u and I_v in $O(1)$ time, $\mathsf{adjacent}(u, v)$ query can be answered in $O(1)$ by checking $r_u < l_v$ or $r_v < l_u$ (u and v are not adjacent if and only if one of these conditions is satisfied).

Due to lack of space, we omit here the rest of the proofs of all the other results that we mention in Section 1.1, and these can be found in the full version of this paper [1].

References

[1] H. Acan, S. Chakraborty, S. Jo, and S. R. Satti. Succinct data structures for families of interval graphs. *CoRR*, abs/1902.09228, 2019.

[2] S. Benser. On the topology of the genetic fine structure. *Proc. Nat. Acad. Sci.*, 45:1607–1620.

[3] D. R. Clark and J. I Munro. Efficient suffix trees on secondary storage. SODA '96, pages 383–391, 1996.

[4] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* 2004.

[5] G. Hajós. Über eine art von graphen. *Int. Math. Nachr.*, 11:1607–1620.

[6] P. Hanlon. Counting interval graphs. *Transactions of the American Mathematical Society*, 272(2):383–426, 1982.

[7] J. C. Yang and N. Pippenger. On the enumeration of interval graphs. *Proc. Amer. Math. Soc. Ser. B*, 4(1):1–3, 2017.

An enhanced lower bound for the Time-Dependent Traveling Salesman Problem

Tommaso Adamo*[1], Gianpaolo Ghiani[1], and Emanuela Guerriero[1]

[1]Dipartimento di Ingegneria dell'Innovazione - Università del Salento, Lecce, Italy

Abstract

Given a graph whose arc traversal times vary over time, the Time-Dependent Travelling Salesman Problem amounts to find a Hamiltonian tour of least total duration. In this paper we define a new lower bounding scheme whose parameters are determined by fitting the traffic data. Computational results show that, when embedded into a branch-and-bound procedure, this lower bounding mechanism allows to solve to optimality a larger number of instances than state-of-the-art algorithms.

1 Introduction

Vehicle routing is concerned with the design of routes for fleets of vehicles, in order to optimize a given objective (such as minimizing the travelled time), possibly subject to side constraints, such as vehicle capacity limitations or delivery time windows. In recent years there has been a flourishing of scholarly work in time-dependent routing. See Gendreau et al. [1] for a review of the field. Given a graph $G = (V \cup \{0\}, A)$ (V is the set of vertices, A is the set of arcs and 0 is the vertex representing the depot) whose arc traversal times vary over time, the Time-Dependent Travelling Salesman Problem (TDTSP) amounts to find a Hamiltonian tour of least total duration. In this work we define a new lower bounding scheme whose parameters are determined by fitting the traffic data.

2 Problem definition and background

Let $[0, T]$ be the time horizon partitioned into H subintervals $[T_h, T_{h+1}]$ ($h = 0, \ldots, H-1$), where $T_0 = 0$ and $T_H = T$. The travel time $\tau_{ij}(t)$ functions are continuous piecewise linear with breakpoints T_h ($h = 0, \ldots, H$), and satisfy the *first-in-first-out* (FIFO) property (Gendreau et al. [1]). Ghiani and Guerriero [2] proved that this class of travel time functions can be generated from the model defined by Ichoua et al. [3] (IGP model) in which the velocity of a vehicle is not constant over the entire arc, but varies when the boundary between two consecutive time periods is crossed. Under these hypotheses, the IGP speeds are nonnegative (Ghiani and Guerriero [2]) and can be decomposed according to the following *speed factorization* [4]:

$$v_{ijh} = u^0_{ij} b^0_h \delta^0_{ijh}, \qquad (i,j) \in A, h = 0, \ldots, H-1 \tag{1}$$

*Corresponding author: tommaso.adamo@unisalento.it

where u^0_{ij} is the maximum speed of arc $(i,j) \in A$ over $[0,T]$, i.e. $u^0_{ij} = \max_{h=0,\dots,H-1} v_{ijh}$; $b^0_h \in [0,1]$ is the lightest congestion factor during interval $[T_h, T_{h+1}]$ on the entire graph, i.e. $b^0_h = \max_{(i,j)\in A} v_{ijh}/u^0_{ij}$; and $\delta^0_{ijh} = v_{ijh}/u^0_{ij}b^0_h \in [0,1]$ is the degradation of the congestion factor of arc (i,j) in interval $[T_h, T_{h+1}]$ w.r.t. the less congested arc in $[T_h, T_{h+1}]$.

Definition 1. $\Delta^0 = \min_{\substack{(i,j)\in A \\ h=0,\dots,H-1}} \delta^0_{ijh}$ *is the worst degradation of the congestion factor of any arc* $(i,j) \in A$ *over the entire planning horizon.*

Δ^0 plays a fundamental role: indeed, when $\Delta^0 = 1$, then all arcs $(i,j) \in A$ have the same congestion factor b^0_h during interval $[T_h, T_{h+1}]$ $(h = 0, \dots, H-1)$.
Cordeau et al. [4] derived a first relaxation of the problem by removing δ_{ijh} for each arc (i,j) and each time period $h = 0, \dots, H-1$. This amounts to solve a TDTSP w.r.t. speeds

$$v^0_{ijh} = b^0_h u^0_{ij}, \quad (i,j) \in A, h = 0, \dots, H-1. \tag{2}$$

A second relaxation can be obtained by giving each arc its maximum speed over the time horizon. This amounts to solve an Asymmetric TSP [5] w.r.t. (constant) speeds

$$\underline{\underline{v}}^0_{ijh} = u_{ij}, \quad (i,j) \in A, h = 0, \dots, H-1. \tag{3}$$

We denote with $z(c,t)$, $\underline{z}(c,t)$, $\underline{\underline{z}}(c,t)$ the duration of a circuit c assuming that the vehicle leaves the depot at time t and speed laws (1), (2) or (3) hold, respectively.

2.1 An enhanced lower bound

We preliminary observe that the speed factorization (1) for arc $(i,j) \in A$ still holds if parameters b_h and δ_{ijh} $(h = 0, \dots, H-1)$ are computed on the basis of a maximum speed u_{ij} greater than u^0_{ij}: i. e. $u_{ij} \geq u^0_{ij} \quad (i,j) \in A,\ h = 0, \dots, H-1$.

This is equivalent to add an additional time slot $h = H$ (in which the vehicle has already returned to the depot) with speed $u_{ij} = v_{ijH} \geq v_{ijh}$ $(h = 0, \dots, H-1)$. Let $\mathbf{u}$ be the vector of u_{ij} associated to arcs $(i,j) \in A$. Then, the travel speeds can be expressed as

$$v_{ijh} = u_{ij} b_h(\mathbf{u}) \delta_{ijh}(\mathbf{u}), \tag{4}$$

where:

- $b_h(\mathbf{u}) \in [0,1]$ is the best congestion factor during interval $[T_h, T_{h+1}]$ w.r.t. $\mathbf{u}$, i.e.,

$$b_h(\mathbf{u}) = \max_{(i,j)\in A} \frac{v_{ijh}}{u_{ij}};$$

- $\delta_{ijh}(\mathbf{u}) = \dfrac{v_{ijh}}{b_h(\mathbf{u}) u_{ij}}$ belongs to $[0,1]$ and represents the degradation of the congestion factor of arc (i,j) in interval $[T_h, T_{h+1}]$ w.r.t. the least congested arc in $[T_h, T_{h+1}]$.

With each vector $\mathbf{u}$ are associated a lower bound $LB(\mathbf{u})$ and an upper bound $UB(\mathbf{u})$. In particular, let $\underline{\underline{c}}(\mathbf{u})$ be the optimal solution value of an Asymmetric TSP whereas arc (i,j) has a cost L_{ij}/u_{ij}. The upper bound is simply $UB(\mathbf{u}) = z(\underline{\underline{c}}(\mathbf{u}))$ while the lower bound $LB(\mathbf{u})$ is:

$$LB(\mathbf{u}) = \phi(\underline{\underline{z}}(\underline{\underline{c}}(\mathbf{u})), 0, \mathbf{b}(\mathbf{u})) \tag{5}$$

where, $\mathbf{b}$ is the vector of traffic factors b_h $(h = 0, \ldots, H-1)$ and $\phi(l, t, \mathbf{b})$ is the traversal time of a dummy arc of length l assuming it is traversed starting at instant t with speeds $\mathbf{b}$. It is worth noting that, by increasing the u_{ij} variables, $\underline{z}(\underline{c}(\mathbf{u}))$ decreases (or remains the same). At the same time, the traffic factors b_h decrease (or remain the same). Hence, the ϕ value increases or remains unchanged. As a result, LB($\mathbf{u}$) may increase, decrease or remain unchanged. In order to find the best (larger) lower bound, the following problem has to be solved:

$$\begin{aligned} &\max LB(\mathbf{u}) \\ s.t. \quad &u_{ij} \geq v_{ijh} \qquad (i,j) \in A, h = 0, \ldots, H-1 \end{aligned} \tag{6}$$

Unfortunately, this problem is nonlinear nonconvex and non-differentiable. So there is little hope to solve it to optimality with a moderate computational effort. Instead, we aim at finding a good lower bound as follows. We first determine a $\mathbf{u}$ vector by fitting the traffic data (solving a linear programming model). More specifically, we determine $\mathbf{u}$ in such a way the average residual,

$$\overline{\delta} = \frac{1}{H|A|} \sum_{h=0,\ldots,H-1} \sum_{(i,j)\in A} \delta_{ijh}, \tag{7}$$

is as large as possible in the hope to get $\Delta(\mathbf{u}) = 1$, or, at least, improve on lower bound $LB(\mathbf{u}^0)$. Then, we solve the Asymmetric TSP w.r.t. costs L_{ij}/u_{ij} in order to compute the associated $LB(\mathbf{u})$.

3 Computational results

We compare the new procedure with the Arigliano et al. [6] branch-and-bound algorithm. We utilize the same instance generation scheme described in Cordeau et al. [4] with 72 periods, and we impose a time limit of 3600 seconds. Two scenarios are generated: a first traffic pattern A in which a limited traffic zone is located in the center; a second traffic pattern B in which a heaviest traffic congestion is situated in the center. The results for the second scenario are shown in Table 1 in which 30 instances are generated for each combination of $|V| = 15, 20, 25, 30, 35, 40, 45, 50$ and $\Delta = 0.90, 0.80, 0.70$. The headings are as follows:

- OPT: number of instances solved to optimality out of 30;
- UB_I/LB_F: average ratio of the initial upper bound value UB_I on the best lower bound LB_F available at the end of the search;
- GAP_I: average initial optimality gap $\frac{UB_I - LB_I}{LB_I}$ (%);
- GAP_F: average final optimality gap $\frac{UB_F - LB_F}{LB_F}$ (%);
- $NODES$: average number of nodes;
- $TIME$: average computing time in seconds.

Except for columns OPT, we report results on two distinct rows: the first row is the average across instances solved to optimality, and the second row is the average for the remaining instances. For the sake of conciseness, the first or the second row has been omitted whenever none or all instances are solved to optimality. For columns from $NODES$ and $TIME$ we report only averages for instances that are solved to optimality.

Computational results show that, when embedded into a branch-and-bound procedure, this lower bounding mechanism allows to solve to optimality a larger number of instances than state-of-the-art algorithms.

Table 1: Computational results for instances with traffic pattern B

		Arigliano et al. [6] branch-and-bound						Arigliano et al. [6] branch-and-bound with the enhanced LB					
Δ	$\|V\|$	OPT	UB_I/LB_F	GAP_I	GAP_F	$NODES$	$TIME$	OPT	UB_I/LB_F	GAP_I	GAP_F	$NODES$	$TIME$
0.70	15	24	1.014 1.010	18.682 23.106	0.000 22.315	15642 41860	868.76 –	28	1.011 1.000	5.550 21.579	0.000 9.327	5074 66059	643.15 –
	20	11	1.007 1.012	9.306 20.970	0.000 17.579	7464 32595	1270.79 –	14	1.007 1.004	4.508 15.856	0.000 5.468	6362 51346	1348.98 –
	25	3	1.020 1.005	6.000 14.399	0.000 11.087	1363 37336	1145.96 –	4	1.016 1.007	3.243 11.152	0.000 4.581	847 35810	526.58 –
	30	1	1.019 1.002	6.303 11.191	0.000 8.887	6326 29854	3263.82 –	2	1.003 1.007	0.317 9.454	0.000 4.688	48 41335	14.03 –
	35	1	1.003 1.004	14.075 9.608	0.000 8.074	576 15242	488.47 –	2	1.000 1.008	0.000 7.583	0.000 5.083	7 39228	14.18 –
	40	0	– 1.003	– 9.902	– 7.381	– 26742	– –	0	– 1.002	– 7.910	– 4.856	– 28032	– –
	45	0	– 1.001	– 11.016	– 8.842	– 27832	– –	1	1.003 1.002	0.343 8.806	0.000 5.793	283 15943	380.00 –
	50	0	– 1.001	– 12.023	– 9.469	– 16771	– –	2	1.000 1.002	0.000 7.581	0.000 5.492	0 14310	5.16 –
0.80	15	27	1.008 1.005	11.804 16.003	0.000 14.780	8590 51281	631.16 –	30	1.007 –	4.145 –	0.000 –	4172 –	287.60 –
	20	17	1.007 1.004	6.218 15.414	0.000 14.421	5656 28191	710.83 –	19	1.008 1.001	3.950 3.836	0.000 3.755	6232 55019	1033.32 –
	25	5	1.011 1.002	3.975 7.940	0.000 5.737	3324 27730	1141.50 –	7	1.007 1.004	2.446 5.920	0.000 3.212	6253 28420	1265.33 –
	30	3	1.007 1.002	3.483 6.756	0.000 4.705	1853 22813	1587.37 –	4	1.001 1.003	1.264 4.905	0.000 3.202	867 20060	763.58 –
	35	1	1.002 1.003	9.882 5.342	0.000 4.290	553 18755	458.66 –	2	1.000 1.002	0.000 4.517	0.000 3.696	10 21260	13.43 –
	40	0	– 1.002	– 5.616	– 4.131	– 24105	– –	0	– 1.002	– 5.136	– 3.633	– 17467	–
	45	0	– 1.001	– 6.213	– 4.829	– 21979	– –	1	1.003 1.001	0.339 5.740	0.000 4.074	186 14109	187.29 –
	50	0	– 1.002	– 6.357	– 5.047	– 13732	– –	2	1.000 1.001	0.000 4.792	0.000 3.742	0 9640	5.59 –
0.90	15	30	1.004	5.718	0.000	1913	157.60	30	1.003	1.898	0.000	358	29.58
	20	24	1.004 1.003	4.318 7.125	0.000 6.789	2873 22047	411.66 –	30	1.003 –	1.904 –	0.000 –	3271 –	351.53 –
	25	17	1.003 1.001	2.311 6.214	0.000 5.187	8304 18914	1057.77 –	22	1.002 1.002	1.359 2.698	0.000 1.857	4406 21989	1108.16 –
	30	14	1.003 1.001	2.300 3.905	0.000 3.191	5105 8551	1657.75 –	15	1.003 1.001	1.345 2.077	0.000 1.531	3668 22643	1660.69 –
	35	4	1.004 1.001	2.398 2.769	0.000 2.107	2035 10269	1761.74 –	9	1.002 1.000	0.878 2.091	0.000 1.633	1608 15296	1440.52 –
	40	4	1.003 1.001	1.747 2.843	0.000 2.217	1697 12462	2596.14 –	1	1.000 1.001	1.680 2.148	0.000 1.565	1768 12088	2473.61 –
	45	1	1.005 1.001	2.040 3.028	0.000 2.435	953 7727	3310.03 –	2	1.001 1.001	0.088 2.357	0.000 1.810	45 9494	162.17 –
	50	0	– 1.002	– 3.069	– 2.462	– 6579	– –	2	1.000 1.001	0.000 2.138	0.000 1.706	0 6766	4.99 –
AVG		187	1.007	7.432	0.000	6146	875.70	229	1.005	2.792	0.000	3561	665.36

References

1. Gendreau, M., Ghiani, G., Guerriero, E.. Time-dependent routing problems: A review. *Computers & Operations Research* 2015;64:189–197.

2. Ghiani, G., Guerriero, E.. A note on the Ichoua, Gendreau, and Potvin (2003) travel time model. *Transportation Science* 2014;48(3):458–462.

3. Ichoua, S., Gendreau, M., Potvin, J.Y.. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research* 2003;144:379–396.

4. Cordeau, J.F., Ghiani, G., Guerriero, E.. Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. *Transportation Science* 2014;48(1):46–58.

5. Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.. The traveling salesman problem: a computational study. Princeton University Press; 2011.

6. Arigliano, A., Calogiuri, T., Ghiani, G., Guerriero, E.. A branch-and-bound algorithm for the time-dependent travelling salesman problem. *Networks* 2018;72(3):382–392.

Vertex-Weighted Realizations of Graphs

Amotz Bar-Noy[1], Toni Böhnlein[2], David Peleg[3], and Dror Rawitz[2]

[1]City University of New York (CUNY), USA. amotz@sci.brooklyn.cuny.edu
[2]Bar Ilan University, Ramat-Gan, Israel. {toni.bohnlein,dror.rawitz}@biu.ac.il
[3]Weizmann Institute of Science, Rehovot, Israel. david.peleg@weizmann.ac.il

Abstract

Given a degree sequence $\bar{d}$ of length n, the degree realization problem is to decide if $\bar{d}$ has a realization. That is a n-vertex graph whose degree sequence is $\bar{d}$, and if this is the case, to construct such a realization (cf. [6, 7, 8]).

We consider the following natural generalization of the problem: Let $G = (V, E)$ be a simple undirected graph on $V = \{1, 2, \ldots, n\}$. Let $\bar{f} \in \mathbb{N}^n$ be a vector of vertex-requirements, and let $w \in \mathbb{N}^n$ be a vector of vertex-weights. The weight vector w satisfies the requirement vector $\bar{f}$ on G if the constraints $\sum_{j \in \Gamma(i)} w_j = f_i$ are satisfied for all $i \in V$, where $\Gamma(i)$ denotes the neighborhood of i. The vertex-weighted realization problem is now as follows: Given a requirements vector $\bar{f}$, find a suitable graph G and a weight vector w that satisfy $\bar{f}$ on G. In the original degree realization problem, all vertex weights are equal to one.

1 Vertex-Weighted Realizations

We start by introducing the problem formally. For $i, j \in \mathbb{N}$ such that $i \leq j$, we use the notation $[i, j] = \{i, \ldots, j\}$. Let $G = (V, E)$ be a simple (no self-loops and no parallel edges) undirected graph with n vertices, where $V = [1, n]$. Let $f = (f_1, \ldots, f_n) \in \mathbb{R}^n_+$ be a vector of *requirements*. Without loss of generality, we assume that $0 \leq f_1 \leq f_2 \leq \ldots \leq f_n$ and define

$$\mathcal{F}_n \triangleq \left\{ f \in \mathbb{R}^n_+ : 0 \leq f_1 \leq f_2 \leq \cdots \leq f_n \right\}.$$

Let $w = (w_1, \ldots, w_n) \in \mathbb{R}^n_+$ be a vector of *provided services* at the vertices. The *available services* at the vertex i, for $i \in [1, n]$, denoted a_i, are those provided in its (exclusive) neighborhood $\Gamma(i)$ (not including i itself), i.e., $a_i := \sum_{j \in \Gamma(i)} w_j$.

We say that the provided services vector w *satisfies* the requirement vector f on the graph G if for all $i \in V$, the available services equal the requirement exactly, i.e., the weights satisfy the following n *requirement constraints*

$$a_i = f_i \ , \tag{RC$_i$}$$

for $i \in [1, n]$. Given a vector f, we say that a vector w and a graph $G = (V, E)$ *realize* f if (RC$_i$) is satisfied for all i. A *domain* $\mathcal{D} \subseteq \mathcal{F}_n$ can be realized if for every $f \in \mathcal{D}$ there exists a pair (G, w) that realizes f. (We usually define domains by one or more linear constraints involving the n requirements.)

Bar-Noy, Peleg, and Rawitz [5] presented the following results.

Theorem 1 ([5])**.** *If n is even, then any $f \in F_n$ can be realized using a perfect matching.*

If graph G is a perfect matching, i.e., each vertex has degree 1, each requirement can be satisfied by the service of its exclusive neighbor. This result shifts the focus to odd sequences. For odd n, the three domains

$$\mathcal{D}^0 := \{f \in \mathcal{F}_n : f_1 = 0\}\,,\ \mathcal{D}^{=} := \{f \in \mathcal{F}_n : \exists i \text{ s.t. } f_i = f_{i+1}\}$$

and

$$\mathcal{D}^{\Delta} := \{f \in \mathcal{F}_n : \exists i < j < k \text{ s.t. } f_k < f_i + f_j\}$$

can be realized with an approach that utilizes a matching graph. Hence, we can focus on odd sequences where

$$0 < f_1 < \ldots < f_n.$$

As a negative result, they showed that a vector $f \in \mathcal{F}_n$ does not have a realization if it belongs to the *exponential growth* domain

$$\mathcal{D}_n^{\exp} = \left\{ f : \forall i \in [1,n], \sum_{j<i} f_j < f_i \right\}\ .$$

Theorem 2 ([5]). *Let $n \geq 3$ be an odd integer. $f \in \mathcal{D}_n^{exp}$ cannot be realized.*

However, f does have a realization if it is found in the *sub-exponential growth* domain

$$\mathcal{D}_n^{\text{sub}} = \left\{ f : \exists i \in [1,n-1], f_i \leq \sum_{j<i} f_j \right\}\ .$$

Theorem 3 ([5]). *Let $n \geq 3$ be an odd integer. $f \in \mathcal{D}_n^{sub}$ can be realized.*

These two theorems are sufficient to fully characterize the case where $n = 3$. Note that the Theorem 3 does not give us conditions on f_n. Basically, this leaves the following domain as unknown:

$$\left\{ f : \forall i \in [1,n-1], \sum_{j<i} f_j < f_i \wedge f_{n-2} + f_{n-1} < f_n < \sum_{j=1}^{n-1} f_j \right\}$$

In this unknown domain Bar-Noy, Peleg, and Rawitz, find two constructions that realize parts of the range. The *windmill* domain:

$$\mathcal{D}_n^{\bowtie} = \left\{ f \in \mathcal{F}_n : \sum_{j=2}^{n-1} (f_j - f_1) \leq f_n \leq \sum_{j=1}^{n-1} f_j \right\}\ ,$$

and the *kite* domain:

$$\mathcal{D}_n^{\triangleright\!\!-} = \left\{ f \in \mathcal{F}_n : \sum_{j=3}^{n-1} f_j \leq f_n \leq f_1 + \sum_{j=3}^{n-1} f_j \right\}\ .$$

The constructions are depicted in Figure 1

Theorem 4 ([5]). *Let $n \geq 5$ be an odd integer. $f \in \mathcal{D}_n^{\bowtie}$ can be realized.*

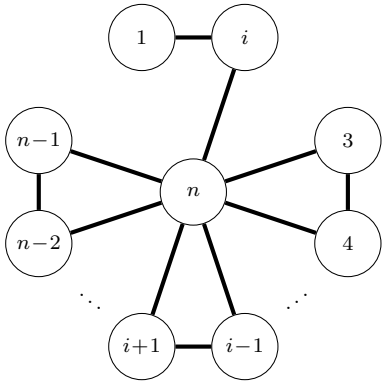


Figure 1: Realization for the kite domain. The Windmill is given by adding the edge $(1, n)$

The windmill domain can be extended using a matching to the following domain.

Theorem 5 ([5])**.** *Let $n \geq 5$ be an odd integer and let $f \in \mathcal{F}^n$. Let k be an odd integer s.t. $k \leq n-4$. If f satisfies*

$$\sum_{j=k+1}^{n-1} (f_j - f_k) \leq f_n < \sum_{j=k}^{n-1} f_j,$$

then it can be realized.

Theorem 6 ([5])**.** *Let $n \geq 5$ be an odd integer. $f \in \mathcal{D}_n^{\triangleright\!-}$ can be realized.*

Similarly, the kite domain can be extended using a matching to the following domain.

Theorem 7 ([5])**.** *Let $n \geq 5$ be an odd integer and let $f \in \mathcal{F}^n$. Let k be an odd integer s.t. $k \leq n-4$. If f satisfies*

$$\sum_{j=k+2}^{n-1} f_j \leq f_n < f_k + \sum_{j=k+2}^{n-1} f_j,$$

then it can be realized.

This leaves us with an unknown domain for each odd integer k: $1 \leq k \leq n-4$, $(n-k)f_k < f_{k+1}$ and

$$f_k + \sum_{j=k+2}^{n-1} f_j < f_n < \sum_{j=k+1}^{n-1} (f_j - f_k).$$

In this work, we show that the gap "between" the windmill and the kite of Theorem 4 & 6, i.e., $k = 1$ cannot be realized.

Theorem 8. *Let $n \geq 5$ be an odd integer, and a vector $f \in \mathcal{F}_n$ cannot be realized if*

(I) $\sum_{j<i} f_j \;<\; f_i$ *for* $i \in [1, n-1]$,

(II) $f_n + (n-1)f_1 \;<\; \sum_{i=1}^{n-1} f_i \;<\; f_n + f_2$.

For the remaining gaps we show that the realizable range can be extended. This is done by using different permutations for the kite and windmill constructions and augmenting the constructions appropriately.

Theorem 9. *Let $n \geq 5$ be an odd integer and let $f \in \mathcal{F}^n$. Let k be an odd integer s.t. $3 \leq k \leq n-4$. If f satisfies*

$$\sum_{j=k+1}^{n-1} (f_j - f_k) \leq f_n < \sum_{j=1}^{n-1} f_j - f_{k+1},$$

then it can be realized.

This narrows the remaining gaps. On the way to showing that there are more un-realizable domains, we discovered additional constructions that let us realize more domains. In a way these constructions generalize the windmill and kite. These domains are not necessarily "connected" to the other realizable domains and split the unknown gaps into several smaller gaps.

While we make an important step with Theorem 8, a full characterization of the problem is an open question.

2 Variations and open Problems

Several variations have been considered. A survey is given by Bar-Noy et. al [3]. Instead of the sum of the neighbor's weights, maximum and minimum-versions were studied [1, 2]. Moreover, notions of vertex-happiness have been investigated [4].

In this work exclusive neighborhoods were considered. In the inclusive neighborhood variant any vertex is part of its own neighborhood. This is an interesting direction for further research. Another intriguing restriction is to allow the realizing graph to come from a given family of graphs, e.g., trees, forests, or bipartite graphs.

References

[1] A. Bar-Noy, K. Choudhary, D. Peleg, and D. Rawitz. Graph realizations for min-neighborhood degree profiles. Unpublished manuscript, 2018.

[2] A. Bar-Noy, K. Choudhary, D. Peleg, and D. Rawitz. Graph realizations: Maximum degree in vertex neighborhoods. Unpublished manuscript, 2018.

[3] A. Bar-Noy, K. Choudhary, D. Peleg, and D. Rawitz. Realizability of graph specifications: Characterizations and algorithms. In *25th International Colloquium on Structural Information and Communication Complexity*, volume 11085 of *LNCS*, pages 3–13, 2018.

[4] A. Bar-Noy, K. Choudhary, D. Peleg, and D. Rawitz. Graph profile realizations and applications to social networks. In *Proc. WALCOM*, LNCS, 2019. To appear.

[5] A. Bar-Noy, D. Peleg, and D. Rawitz. Vertex-weighted realizations of graphs. Unpublished manuscript, 2017.

[6] S. A. Choudum. A simple proof of the Erdös-Gallai theorem on graph sequences. *Bulletin of the Australian Mathematical Society*, 33(1):67–70, 1986.

[7] P. Erdös and T. Gallai. Graphs with prescribed degrees of vertices [hungarian]. *Matematikai Lapok*, 11:264–274, 1960.

[8] S. L. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph –I. *SIAM J. Appl. Math.*, 10(3):496–506, 1962.

Extending Bigraphical Language with Labels

Wissal Ben Amor[1,2], Amal Gassara[1,2], and Ismael Bouassida Rodriguez[1,2]

[1]ReDCAD laboratory, University of Sfax, National School of Engineers of Sfax, 3038 Sfax, Tunisia
[2]Digital Research Center of Sfax, 3021, Sfax, Tunisia

Abstract

In this paper, we present an extension of bigraphs validated by a mathematical proof. The extension aims at enhancing the expressivity of bigraphs through adding labels that carry more properties of modeled components.

Keywords: Bigraphs, Labels, Bigraph extension.

1 Introduction

Bigraphs [3] are a mathematical and graphical model that represents locality and connectivity when dealing with mobile distributed systems. To capture the dynamicity of systems, bigraphs are equipped with transformation rules that enable them to reconfigure themselves Bigraphs [2].

With the diversity of components involved in ubiquitous systems, bigraphs are expected to describe these systems in a clear and expressive way. Actually, each component can be represented in the bigraph with a node that has a type called a control. However, these controls are insufficient to present component properties and characteristics which are very important in modeling such systems.

To tackle this lack of expressivity, we propose, in this paper, an extension that consists in enhacing nodes with labels. These labels enable nodes to carry more information (properties) of the components they represent. To do this, we addressed both abstract bigraphs and concrete bigraphs (i.e., bigraphs where nodes have identifiers).

2 Extending Abstract Bigraphs

Attaching label to nodes in abstract bigraphs can be done by extending the signature (i.e., the set of controls). Thus, it carries not only the types of nodes (controls) and their arities (i.e., the number of ports which enable a node to connect to other nodes), but also an n-tuple of labels.

Definition 1 (Extended Signature)**.** *An extended signature is composed of a set of pairs $K = K' \times L$ and an arity map ar, where K' is a set of controls, L is a set of n-tuple of labels and ar: $K' \times L \longrightarrow N$ arity map.*

Our extension does not alter the bigraph structure in terms of locality neither in terms of linking. Thus, it is sufficient to demonstrate that the elements of K are disjoint to guarantee that our extended signature is confirming to the formal definition of R. Milner. Actually, this is obvious since $K = K' \times L$ and the elements of K' are disjoint.

Nonetheless, extending signature gives bigraphs a limited expressivity since nodes are bound to share the same control and so the same set of labels (properties). However, in modeling ubiquitous

systems, components that have the same type, may have different characteristics and properties. For this reason, we addressed, in our work, concrete bigraphs enabling each node to carry its owns labels.

3 Extending Concrete Bigraphs

For concrete bigraphs, labels are added through the extension of the definition of a bigraph by adding a new function l_v, which assigns labels to nodes. We propose the following definition.

Definition 2 (Labeled bigraph)**.** *A labeled bigraph takes the following form:*
$G = (V, E, ctrl, l_v, prnt, link) : I \longrightarrow J$ *with*
V: the set of nodes $V \subset \upsilon$*, with* υ *a set of node-identifiers.*
E: the set of edges $E \subset \varepsilon$*, with* ε *a set of hyperedge-identifiers.*
$l_v : V \longrightarrow L$*, a labeling function where L is the set of n-tuples of alphabetic tags assigned to nodes.*
Ctrl: $V \longrightarrow K$*, a control map, where the signature K is a set of controls.*
prnt: $m \uplus V \longrightarrow V \uplus n$ *is the parent map and it defines the nested place structure.*
link: $X \uplus P \longrightarrow E \uplus Y$ *is the link map and it defines the link structure.*
I: inner interface $I = \langle m, X \rangle$ *where m is the number of sites and X is the set of inner names.*
J: outer interface $J = \langle n, Y \rangle$ *where n is the number of roots and Y is the set of outer names.*

In order to validate our definition, we should verify that labeled bigraphs form an s-category (let call it BG_L). In an s-category the composition of arrows has to satisfy three constraints, in addition to possessing a partial tensor product, unit, and symmetries. To verify this, we go through each one at a time.
Composition. Composition in BG_L should satisfy the following constraints:
(C1) $g \circ f$ is defined iff $\mathrm{cod}(f) = \mathrm{dom}(g)$ and $|f| \cap |g| = \emptyset$.
(C2) $h \circ (g \circ f) = (h \circ g) \circ f$ when either are defined.
(C3) $id \circ f = f$ and $f = f \circ id$.
Constraint 1 (C1). To prove the validity of the first constraint, we start with proving that if the composition between two labeled arrows $g \circ f$ is defined then $\mathrm{cod}(f) = \mathrm{dom}(g)$ and $|f| \cap |g| = \emptyset$.

Proof. ($\Rightarrow$)Let f and g two morphisms of an s-category BG. Let f_1 and g_1 the labeled morphisms of f and g respectively in BG_L.
Based on our definition of a bigraph, l_v does not assign labels to the elements of interfaces nor it changes the support. Therefore, the objects of BG are the objects of BG_L. Hence;

$$\left.\begin{array}{l} \text{if } g_1 \circ f_1 \text{ is defined} \Rightarrow g \circ f \text{ is defined} \Rightarrow cod(f) = dom(g) \\ cod(f_1) = cod(f) \\ dom(g_1) = dom(g) \end{array}\right\} \Rightarrow cod(f_1) = dom(g_1)$$

In addition, $|f| \cap |g| = \emptyset$. Since, $|f_1| = |f|$ and $|g_1| = |g|$, so, $|f_1| \cap |g_1| = \emptyset$
Hence, we get the desired results $\Rightarrow cod(f_1) = dom(g_1)$ and $|f_1| \cap |g_1| = \emptyset$
($\Leftarrow$) If $cod(f_1) = dom(g_1)$ and $|f_1| \cap |g_1| = \emptyset$ then

$$\left.\begin{array}{l} cod(f_1) = dom(g_1) \\ |f_1| \cap |g_1| = \emptyset \end{array}\right\} \Rightarrow \begin{cases} cod(f) = dom(g) \\ |f| \cap |g| = \emptyset \end{cases} \Rightarrow g \circ f \text{ is defined}$$

Adding labels to g and f maintains $g \circ f$ defined. So, $g_1 \circ f_1$ is also defined. □

Constraint 2 (C2) Following the steps done in the first constraint, we start with the constraint being valid for arrows and objects from an s-category and then move onto labeled arrows (labeled bigraphs) to prove that associativity is applicable on composition in BG_L.

Proof. Let f_1, g_1 and h_1 morphisms in BG_L.
$h_1 \circ (g_1 \circ f_1)$ and $(h_1 \circ g_1) \circ f_1$ are defined. So, $h \circ (g \circ f)$ and $(h \circ g) \circ f$ are also defined where $f : I \longrightarrow J$, $g : J \longrightarrow H$ and $h : H \longrightarrow K$ are the non labeled morphisms inBG.

The $dom(h \circ (g \circ f))$ and the $cod(h \circ (g \circ f))$ are objects in BG that are not impacted by the labeling function. So, $h \circ (g \circ f) = h_1 \circ (g_1 \circ f_1)$ and $(h \circ g) \circ f = (h_1 \circ g_1) \circ f_1$.

Since BG is an s-category, $h \circ (g \circ f) = (h \circ g) \circ f$. Hence, $h_1 \circ (g_1 \circ f_1)$=$(h_1 \circ g_1) \circ f_1$.

□

Constraint 3 (C3) The proof of the third constraint, composition with the identity arrow, is more-or-less like the first constraint. The only difference is that the interfaces remain intact from l_v therefore the identity arrows id in BG are the same in BG_L.

Proof. Let: $f : I \longrightarrow J$ an arrow in the s-category BG.
$f_1 : I \longrightarrow J$ the labeled arrow in BG_L.
Given: $id \circ f = f$ and $f = f \circ id$ (since f is an arrow in an s-category.)

$$\left.\begin{array}{l} id_J \circ f = f \\ cod(f) = cof(f1) \\ |f1| = |f| \end{array}\right\} \Rightarrow id_J \circ f_1 \text{ is defined and } id_J \circ f_1 = f_1$$

$$\left.\begin{array}{l} f \circ id_I = f \\ dom(f) = dom(f1) \\ |f1| = |f| \end{array}\right\} \Rightarrow f_1 \circ id_I \text{ is defined and} f_1 \circ id_I = f_1$$

□

Tensor product. A tensor product is the juxtaposition of the roots of two morphisms (bigraphs); requiring that their outer names and inner names are respectively disjoint.

Proof. As we mentioned previously, the labeling function l_v does not affect the interfaces. Hence, objects in an s-category BG are the same objects in BG_L. Thus, BG_L has a tensor product that satisfies the following:
For $f : I_0 \longrightarrow I_1$ and $g : J_0 \longrightarrow J_1$ in BG, the tensor product $f \otimes g$ is defined iff $Ii \otimes Ji$ is defined (i=0,1) and $|f| \cap |g| = \emptyset$.

Let: f_1 and g_1 the labeled versions of f and g in BG_L, $f_1 : I_0 \longrightarrow I_1$ and $g_1 : J_0 \longrightarrow J_1$ (since the object of BG are the same objects of BG_L)
($\Rightarrow$)If $f_1 \otimes g_1$ is defined then:

$$\left.\begin{array}{l} |f| = |f_1| \\ |g| = |g_1| \\ f_1 \otimes g_1 \text{is defined} \end{array}\right\} f \otimes g \text{is defined} \begin{cases} I_0 \otimes J_0 and I_1 \otimes J_1 \\ |f| \cap |g| = \emptyset \Rightarrow |f_1| \cap |g_1| = \emptyset \end{cases}$$

($\Leftarrow$)If $I_0 \otimes J_0 and I_1 \otimes J_1$ are defined and $|f| \cap |g| = \emptyset$ then:

$$\left.\begin{array}{l} I_0 \otimes J_0 \text{and} I_1 \otimes J_1 \text{are both defined} \\ \left.\begin{array}{l} |f_1| \cap |g_1| = \emptyset \\ |f| = |f_1| \\ |g| = |g_1| \end{array}\right\} \Rightarrow |f| \cap |g| = \emptyset \end{array}\right\} f \otimes g \text{is defined and with } l = l_{f1} \uplus l_{g1} \Bigg\} \Rightarrow f_1 \otimes g_1 \text{is defined.}$$

□

Symmetries. Symmetries are arrows with empty support, ensuing permutations on the place graph structure. Given the established statement that the objects of BG_L are the objects of BG, BG_L possesses symmetries.
A symmetry arrow of two objects/interfaces $I = \langle m, X \rangle$ and $J = \langle n, Y \rangle$
$\gamma_{I,J} : I \otimes J \longrightarrow J \otimes I$, is defined (when both $I \otimes J$ and $J \otimes I$ are defined) as follows: $\gamma_{\langle m,X \rangle, \langle n,Y \rangle} = \langle \gamma_{m,n}, \gamma_{X,Y} \rangle$ With:

$$\gamma_{m,n} = (\emptyset, \emptyset, prnt), \text{ where } prnt(i) = n + i(i \in m) \\ \text{and } prnt(m + j) = j(j \in n)$$

$\gamma_{X,Y} = id_{X \uplus Y}$

Unit. A unit is an object $\epsilon = \langle 0, \emptyset \rangle$, that satisfies the following equalities which are valid in BG_L since they are valid in BG for having the same objects:
$\epsilon \circ I = I \circ \epsilon = I$
$\epsilon \otimes I = I \otimes \epsilon = I$

4 Conclusion

In this paper, we presented our solution for extending bigraphs in order to increase their expressivity through labels that carry more properties of nodes. This extension was proved valid through two steps. First, for labeled abstract bigraphs, we proved that our definition of an extended signature conforms the formal definition given by Milner[3]. Second, for labeled concrete bigraphs, we proved that concrete bigraphs are casted as an s-category. In future work, we aim at implementing this extension into BiGMTE[1] tool[1].

References

[1] Amal Gassara, Ismael Bouassida Rodriguez, and Mohamed Jmaiel, *A tool for modeling SoS architectures using bigraphs*, Proceedings of the Symposium on Applied Computing, ACM, 2017, pp. 1787–1792.

[2] Amal Gassara, Ismael Bouassida Rodriguez, Mohamed Jmaiel, and Khalil Drira, *A bigraphical multi-scale modeling methodology for system of systems*, Computers & Electrical Engineering **58** (2017), 113–125.

[3] Robin Milner, *The space and motion of communicating agents*, Cambridge University Press, 2009.

[1] www.redcad.tn/projects/bigmte

The robust bilevel continuous knapsack problem

Christoph Buchheim[1] and Dorothee Henke[1]

[1]Fakultät für Mathematik, TU Dortmund, Vogelpothsweg 87, 44227 Dortmund, Germany

Abstract

We consider a bilevel continuous knapsack problem where the leader controls the capacity of the knapsack and the follower's profits are uncertain. Adopting the robust optimization approach and assuming that the follower's profits belong to a given uncertainty set, our aim is to compute a worst case optimal solution for the leader. We show that this problem can be solved in polynomial time for both discrete and interval uncertainty. In the latter case, we make use of an algorithm by Woeginger [8] for a class of precedence constraint knapsack problems.

1 Introduction

The aim of bilevel optimization is to model situations where certain decisions are taken by a so-called leader, but then one or more followers optimize their own objective functions subject to the choices of the leader. The follower's decisions in turn influence the leader's objective, or even the feasibility of her decisions. The objective is to determine an optimal decision from the leader's perspective. Even in the case that both the leader and the follower solve linear programs, the bilevel problem turns out to be strongly NP-hard in general [6]. Several surveys and books on bilevel optimization have been published, e.g., [3, 5].

Our research is motivated by the question of how much harder does bilevel optimization become, when adopting the robust optimization approach to address uncertainties. In this approach, the uncertain parameters are specified by so-called uncertainty sets which contain all possible (or likely) scenarios; the aim is to find a solution that is feasible in all scenarios and that optimizes the worst case. The only article we are aware of that addresses robustness in bilevel optimization is [2].

In classical one-level robust optimization, even if uncertainty only occurs in the objective, some classes of uncertainty sets may lead to substantially harder problems, e.g., finite uncertainty sets in the context of combinatorial optimization [7]. In other cases, the problems can be solved by an efficient reduction to the underlying certain problem. This is true in particular for the case of interval uncertainty, where each coefficient may vary independently within some interval. For an overview of complexity results in robust combinatorial optimization under objective uncertainty, we refer the reader to the recent survey [1] and the references therein.

We concentrate on a bilevel continuous knapsack problem where the leader only controls the capacity. Without uncertainty, this problem is easy to solve; see Section 2. However, with interval uncertainty on the follower's objective, the problem becomes more involved. Adapting an algorithm by Woeginger [8] for some precedence constraint knapsack problem, we show that it can still be solved in polynomial time; see Section 4. Before, we also discuss why the case of finite uncertainty sets is tractable as well; see Section 3.

Both results are problem-specific and thus do not answer the question whether an efficient oracle-based algorithm exists, using an oracle for the certain case. However, we believe that the additional difficulty of the problem in the interval case makes the existence of such an algorithm unlikely.

2 Underlying certain problem

We first discuss the deterministic variant of the bilevel optimization problem under consideration; see also [5] for more details. The overall (certain) problem can be formulated as follows:

$$\begin{array}{lll} \min & d^\top x & \\ \text{s.t.} & b^- \le b \le b^+ & \\ & x \in \operatorname{argmax} & c^\top x \\ & \quad \text{s.t.} & a^\top x \le b \\ & & 0 \le x \le 1 \end{array} \tag{P}$$

The leader's variable is $b \in \mathbb{R}$ and the follower's variables are $x \in \mathbb{R}^n$. The vectors $a, c \in \mathbb{R}^n_{\ge 0}$ and $d \in \mathbb{R}^n$ and the bounds $b^-, b^+ \in \mathbb{R}$ are given. We may assume $0 \le b^- \le b^+ \le \sum_{i=1}^n a_i$, $a > 0$ and $c > 0$. To simplify presentation, we always assume the follower's optimum solution to be unique.

The follower solves a continuous knapsack problem which can be done, for example, using Dantzig's algorithm [4]: by first sorting the items, we may assume $\frac{c_1}{a_1} \ge \cdots \ge \frac{c_n}{a_n}$. The idea is then to pack the items into the knapsack in this order until it is full; only a fraction of the so-called critical item being taken. Note that, due to the assumptions $0 \le b \le \sum_{i=1}^n a_i$ and $c > 0$, every optimum solution of the follower's problem satisfies $a^\top x = b$.

Now, as only the critical item, but not the sorting depends on b, the leader can just compute the described order of items, and her problem can be reformulated as minimizing the function

$$f(b) := \begin{cases} 0 & \text{for } b = 0 \\ \displaystyle\sum_{i=1}^{j-1} d_i + \frac{d_j}{a_j}\Big(b - \sum_{i=1}^{j-1} a_i\Big) & \text{for } b \in \Big(\sum_{i=1}^{j-1} a_i, \sum_{i=1}^{j} a_i\Big],\ j \in \{1, \dots, n\} \end{cases}$$

over $b \in [b^-, b^+]$. As f is piecewise linear, it suffices to evaluate f at the boundary points b^- and b^+ and at all feasible vertices, i.e., at $b = \sum_{i=1}^j a_i$ for all $j \in \{0, \dots, n\}$ with $\sum_{i=1}^j a_i \in [b^-, b^+]$. Hence, Problem (P) can be solved in time $\mathcal{O}(n \log n)$, which is the time needed for sorting.

3 Finite uncertainty

We now look at the robust version of the problem where the follower's objective function is uncertain for the leader, and this uncertainty is given by a finite uncertainty set U containing the possible objective vectors c. A formulation of this is given by replacing the leader's objective $d^\top x$ by $\max_{c \in U} d^\top x$ in Problem (P).

The inner maximization problem can be interpreted as being controlled by an adversary, thus leading to an optimization problem involving three actors: first, the leader takes her decision b, then the adversary chooses a follower's objective c that is worst possible for the leader, and finally the follower optimizes this objective choosing x.

Again, we aim at solving this problem from the leader's perspective, which can be done as follows: for every $c \in U$, consider the piecewise linear function f_c as described in Section 2. The task is then to minimize the pointwise maximum $f := \max_{c \in U} f_c$ over $[b^-, b^+]$. By considering the number of vertices the piecewise linear function f can have, we get:

Theorem 1. *The robust bilevel continuous knapsack problem with finite uncertainty set U can be solved in $\mathcal{O}(|U| n \log n + |U|^2 n)$ time.*

4 Interval uncertainty

We now consider $U = [c_1^-, c_1^+] \times \cdots \times [c_n^-, c_n^+]$ and assume $0 < c^- \leq c^+$. To simplify the notation, we define $p_i^- := \frac{c_i^-}{a_i}$ and $p_i^+ := \frac{c_i^+}{a_i}$, and assume that all values p_i^- and p_i^+ are pairwise different.

For the leader, the exact entries of c_i in their intervals $[c_i^-, c_i^+]$ do not matter, but only the induced sorting that the follower will use. Given U and a, the possible sortings are exactly the linear extensions of the partial order P that is induced by the intervals $[p_i^-, p_i^+]$ in the sense that we set

$$i <_P j \quad :\Leftrightarrow \quad p_i^+ < p_j^- \ .$$

Such a partial order is called an *interval order*. One could compute all linear extensions of P and the pointwise maximum over all corresponding piecewise linear functions as in Section 3, but these could be exponentially many. However, the problem can still be solved in polynomial time.

We will see that the adversary's problem for fixed b is closely related to the *precedence constraint knapsack problem*. This is a 0-1 knapsack problem, where additionally, a partial order on the items is given and it is only allowed to pack an item into the knapsack if all its predecessors are also selected. For the special case where the partial order is an interval order, Woeginger described a pseudopolynomial algorithm, see Lemma 11 in [8]. The algorithm uses the idea that every initial set (i.e. prefix of a linear extension of the interval order) consists of

- a *head*, which is the element whose interval has the rightmost left endpoint among the set,
- all predecessors of the head in the interval order, and
- some subset of the elements whose intervals contain the left endpoint of the head in their interior.

Our algorithm for the adversary's problem is a variant of this algorithm for the continuous knapsack.

For this, we need the notion of a *fractional prefix* of a partial order P, which is a triple (J, j, λ) such that $J \subseteq \{1, \ldots, n\}$, $j \in J$, $0 < \lambda \leq 1$, and there is an order of the elements in J, ending with j, that is a prefix of a linear extension of P. The follower's solution corresponding to a fractional prefix $F = (J, j, \lambda)$ is defined by $x_i^F := 1$ for $i \in J \setminus \{j\}$, $x_i^F := 0$ for $i \in \{1, \ldots, n\} \setminus J$, and $x_j^F := \lambda$. Additionally, there is the empty fractional prefix $\emptyset$ with $x^\emptyset = 0$. Let $\mathcal{P}$ be the set of all fractional prefixes of the interval order P. Then the leader's problem can be reformulated as follows:

$$\min_{b \in [b^-, b^+]} \quad \max_{\substack{F \in \mathcal{P} \\ a^\top x^F = b}} \quad d^\top x^F$$

We first focus on the inner maximization problem, the *adversary's problem*. In the case where the interval order has no relations, this problem is very similar to the ordinary continuous knapsack problem and can be solved using Dantzig's algorithm, as well. This will also be used as a subroutine on a subset of the elements.

The general adversary's problem can be solved by Algorithm 1. In the notation of Woeginger's algorithm, the k-th element is the head in iteration k, I_k^- is the set of its predecessors, and I_k^0 corresponds to the intervals containing the left endpoint of the head – not necessarily in their interior, so that, in particular, also $k \in I_k^0$. The basic difference to Woeginger's algorithm is that due to the fractionality, it is important to have a dedicated last element of the prefix. In our construction, any element of I_k^0 could be this last element, in particular it could be k, but it does not have to.

To solve the *leader's problem*, Algorithm 1 can be generalized to work for a range $[b^-, b^+]$ of capacities instead of a fixed b by using the variant of Dantzig's algorithm which returns a piecewise linear function, as described in Section 2. Then the iterations give piecewise linear functions depending on b. Finally, the minimum of their pointwise maximum, over $[b^-, b^+]$, is computed.

Algorithm 1: Algorithm for the adversary's problem

Input : $a \in \mathbb{R}^n_{>0}$, $0 \leq b \leq \sum_{i=1}^n a_i$, $d \in \mathbb{R}^n$, $0 < p^- < p^+$
Output: $F \in \mathcal{P}$ with $a^\top x^F = b$ maximizing $d^\top x^F$

if $b = 0$ **then**
  return $\emptyset$
$K := \emptyset$
for $k = 1, \ldots, n$ **do**
  $I_k^- := \{i \in \{1, \ldots, n\} : p_i^+ < p_k^-\}$
  $I_k^+ := \{i \in \{1, \ldots, n\} : p_i^- > p_k^-\}$
  $I_k^0 := \{1, \ldots, n\} \setminus (I_k^- \cup I_k^+)$
  if $0 < b - \sum_{i \in I_k^-} a_i \leq \sum_{i \in I_k^0} a_i$ **then**
    $(J'_k, j_k, \lambda_k) := \textsc{Dantzig}(I_k^0, d_{I_k^0}, a_{I_k^0}, b - \sum_{i \in I_k^-} a_i)$
    $J_k := J'_k \cup I_k^-$
    $K := K \cup \{k\}$
return (J_k, j_k, λ_k) with $k = \operatorname{argmax}\{d^\top x^{(J_k, j_k, \lambda_k)} : k \in K\}$

Analyzing the time complexity of the piecewise linear function computations needed gives:

Theorem 2. *The robust bilevel continuous knapsack problem with interval uncertainty can be solved in $\mathcal{O}(n^3)$ time.*

References

[1] Christoph Buchheim and Jannis Kurtz. Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization*, 6(3):211–238, 2018.

[2] Thai Doan Chuong and Vaithilingam Jeyakumar. Finding robust global optimal values of bilevel polynomial programs with uncertain linear constraints. *Journal of Optimization Theory and Applications*, 173(2):683–703, 2017.

[3] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, 2007.

[4] George B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2):266–277, 1957.

[5] Stephan Dempe, Vyacheslav Kalashnikov, Gerardo A. Pérez-Valdés, and Nataliya Kalashnykova. *Bilevel Programming Problems*. Springer, 2015.

[6] Pierre Hansen, Brigitte Jaumard, and Gilles Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13(5):1194–1217, 1992.

[7] Panos Kouvelis and Gang Yu. *Robust Discrete Optimization and Its Applications*. Springer, 1996.

[8] Gerhard J. Woeginger. On the approximability of average completion time scheduling under precedence constraints. *Discrete Applied Mathematics*, 131(1):237–252, 2003.

A single machine on-time-in-full scheduling problem *

Marco Casazza[1], Alberto Ceselli[1], and Giovanni Righini[1]

[1]Università degli Studi di Milano, Dipartimento di Informatica, email:{firstname.lastname}@unimi.it

A relevant feature in many production contexts is flexibility. This becomes a key issue, for instance, in the case of third-party cosmetics manufacturing [1]. There, the core business is the production of high quality, fully custom orders in limited batches. Competition is pushing companies to aggressive commercial policies, involving tight delivery dates. At the same time, the custom nature of the orders makes it impossible to keep materials in stock; lead times are always uncertain, often making release dates tight as well, and ultimately yielding unexpected peaks of production loads.

At a scheduling stage, such an on-time-in-full (never split a job, always satisfy the customer within its delivery date with a single batch) company policy produces problems which are not only hard to solve by human experts, but often infeasible. As a consequence, delivery dates are sistematically violated, thereby lowering the perceived quality of service and triggering a loop of more aggressive commercial policies.

We consider a minimal relaxation of such a policy: in case scheduling all batches is infeasible, we leave the option of splitting some of them in two fragments (at a price), postponing the delivery date of the second fragment.

In this paper we focus on the combinatorial investigation of the fundamental case in which a single machine is available, with the target of using our findings in a column generation based algorithm for the general multi-machine multi-time-slot case. We formalize our main modeling assumptions, observe a few fundamental properties, and introduce an exact dynamic programming algorithm.

1 Models and assumptions

Formally, let J be a set of jobs (modeling customer batches). For each $j \in J$, let a release date r_j and a due date d_j be given. Let $\pi_j = p_j + q_j$ be the processing time of job j; when j is fragmented, we assume the processing time p_j (resp. q_j) of the first fragment (resp. second fragment) to be given.

In our setting, it is realistic to assume that scheduling is performed in fixed time slots (e.g. weekly), such that no job is left pending at the end of each slot (e.g. over the weekend). Let P be the total processing capacity of a particular machine in a particular time slot. From an application point of view, we expect a vast majority of the jobs to have $d_j - r_j > P$. However, we assume $\pi_j \leq P$.

Furthermore, let σ be the starting time of the optimization time slot. We preprocess release and due dates as $r_j = \max\{r_j - \sigma, 0\}$ and $d_j = \min\{d_j - \sigma, P\}$.

*Partially funded by Regione Lombardia, grant agreement n. E97F17000000009, Project AD-COM

Finally, we assume that no job preemption is possible, as it would imply additional setup time and storage costs.

Indeed, at a single machine stage, two conflicting objectives need to be considered: on one side, to perform as few splits as possible, as we expect good multi-machine multi-time-slot solutions to include few splits overall; on the other side, to schedule as many jobs as possible (potentially splitting them).

Our single machine on-time-in-full with fixed fragmentation scheduling problem (1-OTIFF) can be modeled as a bi-objective optimization problem as follows:

$$\max \sum_{j \in J} x_j, \max \sum_{j \in J} z_j \tag{1}$$

$$\text{s.t. } x_j \leq z_j \qquad \forall j \in J \tag{2}$$

$$z_i + z_j \leq y_{ij} + y_{ji} + 1 \qquad \forall i, j \in J : i \neq j \tag{3}$$

$$s_j + p_j z_j + q_j x_j \leq e_j \qquad \forall j \in J \tag{4}$$

$$e_i \leq s_j + \mathcal{M}(1 - y_{ij}) \qquad \forall i, j \in J, \tag{5}$$

$$x_j, z_j, y_{ij} \in \{0, 1\} \qquad \forall i, j \in J \tag{6}$$

$$r_j \leq s_j, e_j \leq d_j \qquad \forall j \in J \tag{7}$$

where variables x_j take value 1 if job j is performed in full, 0 otherwise, z_j take value 1 if job j is scheduled (either in full or after splitting), 0 otherwise, y_{ij} take value 1 if jobs i and j are both scheduled and i preceeds j, 0 otherwise, s_j (resp. e_j) take the starting time (resp. ending time) of job j (or are not influent if job j is not scheduled). Objective functions (1) maximize the number of scheduled jobs, and that of jobs which are scheduled in full. Constraints (2) ensure consistency between split and full job selection. Constraints (3) ensure consistency between z_j and y_{ij} values. Constraints (4) force consistency between job starting and ending time, when the job is selected. Constraints (5) are non-overlapping conditions. Constraints (6) and (7) define variable domains.

2 Properties and algorithms

It is not hard to prove the following:

Proposition 1. *When all due dates are identical, there always exists an optimal solution in which jobs are processed in order of non-decreasing release date*

and symmetrically

Proposition 2. *When all release dates are identical, there always exists an optimal solution in which jobs are processed in order of non-decreasing due date.*

Therefore, we partition the set of jobs J: those whose release date is the starting of the scheduling time slot (R), those not belonging to R whose due date is the ending of the scheduling time slot (D), and the remaining ones (S). We sort jobs by considering elements of R first, elements of D second and elements of S as last ones. We also fix an arbitrary order between elements in each subset, thereby fixing a total order between the jobs. We indicate $j > i$ (resp. $j < i$) whenever job j appears after (resp. before) job i in such a total ordering.

We remark that, as discussed in the modeling section, we expect S to be very small, being composed only by those jobs having both release and due date within the scheduling time slot.

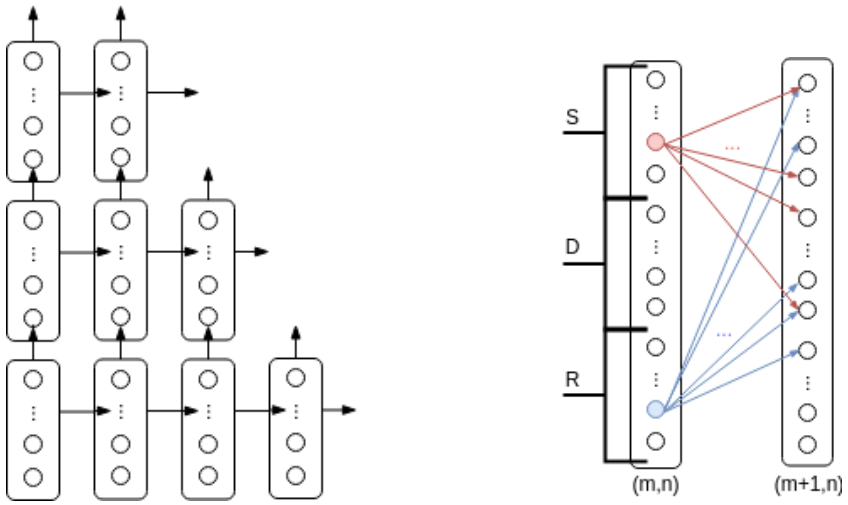


Figure 1: Layers grid structure (left): coordinates in the grid represent (m, n) values. Inter-layer arcs (right): blue arcs refer to a vertex i in either R or D, connected only to vertices $j > i$; red arcs refer to a vertex in S, connected to all vertices.

Second, we observe that finding suitable upper bounds on the maximum number f (resp. s) of jobs that are served in full (resp. partially), can be performed by simply solving a single-machine traditional scheduling problem, or even approximated by solving binary knapsack problems.

Then, we build a directed graph G having one layer for each $m = 0 \dots f$, for each $n = 0 \dots s$. Each layer (m,n) contains one vertex for each job; informally, we indicate a vertex to be in R (resp. D and S) if it is encoding a job in R (resp. D and S); therefore, each job is encoded by $f \cdot s$ vertices in G, one for each layer. Similarly, given two vertices i and j we indicate either $i < j$ or $i > j$ according to the relative position of the corresponding jobs in their total ordering. Each vertex i in S is connected to all vertices in both layer $(m+1,n)$ and layer $(m,n+1)$. Each vertex i in either R or D is instead connected only to vertices j having $j > i$ in both layers $(m+1,n)$ and $(m,n+1)$. Formally, we replace layer $(0, 0)$ with a single dummy vertex 0.

The layers form a grid (Figure 1, left), arcs connect only adjecent layers (Figure 1, right).

1-OTIFF solutions are represented by paths, starting from 0, in such a graph. For finding optimal ones we design a dynamic programming algorithm, working as follows. At each vertex of G we consider *labels* encoding partial solutions, having the form (i, Q, t): t is the overall processing time spent so far, i is the last visited vertex *belonging to* $R \cup D$, Q is the set of jobs in S for which a corresponding vertex has already been visited in the partial solution. We initialize a single label of value $(0, \emptyset, 0)$ in the dummy vertex 0. Then we proceed in phases. At each phase l, we proceed by layers, considering in turn for each $v = 0 \dots l$ the layer $(l - v, v)$. For each layer we proceed by vertex: we perform extension operations, pushing its labels to neighbour vertices.

In particular, when extending a label (i, Q, t) in layer (m, n) from a vertex k to a vertex j

- if $j \notin S$, a new label $(j, Q, t + \pi_j)$ is created in layer $(m + 1, n)$ and a new label $(j, Q, t + p_j)$ is created in layer $(m, n + 1)$
- if $j \in S$, a new label $(i, Q \cup \{j\}, t + \pi_j)$ is created in layer $(m + 1, n)$ and a new label $(i, Q \cup \{j\}, t + p_j)$ is created in layer $(m, n + 1)$

We avoid extensions whenever $j \in Q$, in order to avoid including twice the same job in the partial solution, or $t > P$ in the new label, to avoid exceeding the available time slot. Furthermore

we run dominance checks: taken two labels $\lambda' = (i', Q', t')$ and $\lambda'' = (i'', Q'', t'')$ belonging to the same vertex, we prune λ'' whenever $i' \leq i''$, $Q' \subseteq Q''$ and $t' \leq t''$, and at least one of these conditions is strict.

The algorithm terminates when no new label is created during a particular phase.

We can prove that

Proposition 3. *after termination, all Pareto-optimal solutions are encoded by labels*

and in particular, by labels in the top-right outermost border of non empty layers.

We discuss efficient implementations of our algorithm, and adaptations of speedup techniques from the literature.

Potentially, our algorithm has a high degree of computing parallelism, we therefore focus on effective techniques to enable concurrency.

As a perspective of future research, we remark that our algorithm naturally extends to the case in which prizes are assigned to the selection of jobs, in both full and split options. We expect such an extension to produce effective multiple pricing routines in a column generation setting for the multi-machine multi-time-slot version of the problem.

References

[1] Advanced Cosmetics Manufacturing, project website, https://ad-com.net/ (last access March 2019).

On aircraft deconfliction by bilevel programming

Martina Cerulli[1], Claudia D'Ambrosio[1], and Leo Liberti[1]

[1]CNRS - LIX, Ecole Polytechnique, 91128, Palaiseau, France

Abstract

We present a bilevel programming formulation for the aircraft deconfliction problem with multiple lower-level subproblems. We propose two reformulation based on the KKT conditions and the dual of the lower-level subproblems. Finally, we compare the results obtained implementing these formulations using global optimization solvers.

1 Introduction

We consider the problem of *aircraft deconfliction*, or, in other words, detection and resolution of aircraft conflicts, which is one of the main tasks of Air Traffic Management. Aircraft are said to be potentially *in conflict* if their relative distance is smaller than a given safety threshold. Despite the importance of this kind of control, it is still widely performed manually on the ground by air traffic controllers, who essentially monitor the air traffic of a certain period of time on a radar screen, giving instructions to the pilots. Since the level of automation reached on aircraft is very high, the need for automatic tools to integrate human work on the ground is evident.
There are several ways in which aircraft conflicts can be avoided. The most common is based on the change of the trajectory or the flight level of the involved aircraft. This is the way air traffic controllers usually solve potential conflicts. Another strategy consists in slightly changing the speeds while keeping the trajectories unchanged. The latter is the variant on which we will focus in this paper. We present a Mathematical Programming (MP) formulation for aircraft separation based on speed regulation. For a wider introduction to this problem, see [1].
We will assume that aircraft fly within a fixed altitude layer: they can thus be modeled as points in $\mathbb{R}^2$ (see Figure 1 as an example).

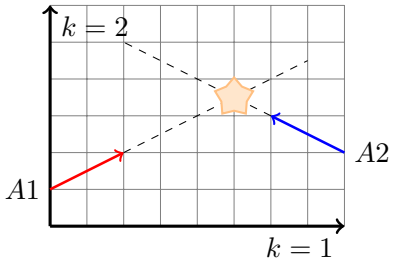


Figure 1: Two conflicting aircraft

Deconfliction is expected to involve minimal deviations from the original aircraft flight plan. Therefore, the objective function to minimize will take into account the percentage of speed change of each aircraft. Of course, for each aircraft pair, we must assure that they do not get closer to each other than a given safety distance at every time t of a fixed interval $[0, T]$ (note that this generates an uncountably infinite set of constraints).

2 Mathematical Formulations

We propose a MP formulation of the speed-change problem variant. The terminology and symbols are taken from [1].

1. **Sets:**
   - $A = \{1, .., n\}$ is the set of aircraft (n aircraft move in the shared airspace)
   - $K = \{1, 2\}$ is the set of directions (the aircraft move in a Euclidean plane)
2. **Parameters:**
   - T is the length of the time horizon taken into account [hours]
   - d is the minimum required safety distance between a pair of aircraft [NauticalMile NM]
   - x_{ik}^0 is the k-th component of the initial position of aircraft i
   - v_i is the initial speed of aircraft i [NM/h]
   - u_{ik} is the k-th component of the direction of aircraft i
   - q_i^{min} and q_i^{max} are the bounds on the ratio of the speed for each aircraft
3. **Variables:**
   - q_i is the is the possible increase or decrease of the original speed of aircraft i: $= 1$ if the speed is unchanged, $q_i > 1$ if it is increased, $q_i < 1$ if it is decreased
   - t_{ij} is the instant of time defined for the aircraft pair i and j; these variables help us compute the relative distance between i and j in time interval $[0, T]$

2.1 Bilevel formulation of the problem

In order to address the issue of uncountably many constraints for each value in $[0, T]$, we propose to formulate the problem as a bilevel MP with multiple second level problems. Each of these subproblems ensures that the minimum distance between each aircraft pair exceeds the safety distance threshold. Thus, each lower-level subproblem involves the lower-level variable t, and is parameterized by the upper-level variables q.

$$\min_{q,t} \quad \sum_{i \in A} (q_i - 1)^2 \tag{1}$$

$$\forall i \in A \quad q_i^{\mathsf{min}} \leq q_i \leq q_i^{\mathsf{max}} \tag{2}$$

$$\forall i < j \in A \quad d^2 \leq \min_{t_{ij} \in [0,T]} \sum_{k \in \{1,2\}} ((x_{ik}^0 - x_{jk}^0) + t_{ij}(q_i v_i u_{ik} - q_j v_j u_{jk}))^2 \tag{3}$$

The upper-level (convex) objective function is the sum of squared aircraft speed changes. This corresponds to finding the feasible solution with the minimum speed change, as mentioned before. It must be minimized w.r.t the time t and the variable q, with each q_i within the given range $[q_i^{\mathsf{min}}, q_i^{\mathsf{max}}]$. The objective of each lower-level subproblem is to minimize over $t_{ij} \in [0, T]$ the relative Euclidean distance between the two aircraft it describes; note that this is also a convex function. This minimum distance, reached at t_{ij}^*, must be at least d^2. This corresponds to imposing the minimum safety distance d between aircraft i and j within $[0, T]$.

2.2 KKT reformulation

We follow standard practice and replace each convex lower-level subproblem by its Karush-Kuhn-Tucker (KKT) conditions. Assuming some regularity condition (e.g. the Slater's condition) holds,

this yields a single-level MP with complementarity constraints. Given the KKT multipliers μ_{ij} and λ_{ij} defined for each lower-level problem, we have:

$$\min_{q,t} \quad \sum_{i\in A}(q_i-1)^2 \tag{4}$$

$$\text{s.t.} \quad \forall i \in A \quad q_i^{min} \le q_i \le q_i^{max} \tag{5}$$

$$\forall i<j \in A \quad \sum_{k\in\{1,2\}} (2t_{ij}(q_i v_i u_{ik} - q_j v_j u_{jk})^2 + \\ + 2(x_{ik}^0 - x_{jk}^0)(q_i v_i u_{ik} - q_j v_j u_{jk}) - \mu_{ij} + \lambda_{ij}) = 0 \tag{6}$$

$$\forall i<j \in A \quad \mu_{ij},\ \lambda_{ij} \ge 0 \tag{7}$$

$$\forall i<j \in A \quad \mu_{ij}\, t_{ij} = 0 \tag{8}$$

$$\forall i<j \in A \quad \lambda_{ij}\, t_{ij} - \lambda_{ij}\, T = 0 \tag{9}$$

$$\forall i<j \in A \quad -t_{ij} \le 0,\ t_{ij} \le T \tag{10}$$

$$\forall i<j \in A \quad \sum_{k\in\{1,2\}} ((x_{ik}^0 - x_{jk}^0) + t_{ij}(q_i v_i u_{ik} - q_j v_j u_{jk}))^2 \ge d^2 \tag{11}$$

The last constraint Eq. (11) is necessary to ensure that each KKT solution t_{ij}^* respects the safety distance.

2.3 Dual reformulation

We propose another closely related reformulation of the bilevel problem (1)-(3), which arises because the lower-level subproblems are convex Quadratic Programs (QP). Specifically, their duals are also QPs which only involve dual variables [2, 3]. In particular, an upper-level constraint such as Eq. (3) has the form $\mathsf{const} \le \min\{\frac{1}{2}\, x^\top Q x + p^\top x \mid Ax \ge b \wedge x \ge 0\}$ with Q positive semidefinite. By strong duality it can be written as follows:

$$\mathsf{const} \le \max\{-\frac{1}{2}\, y^\top Q y + b^\top z \mid A^\top z - Qy \le p \wedge z \ge 0\}, \tag{12}$$

where the maximization QP on right hand side is the dual of the previous minimization one [3].

Proposition 1. *Eq.* (12) *can be replaced by* $\mathsf{const} \le -\frac{1}{2}\, y^\top Q y + b^\top z \wedge A^\top z - Qy \le p \wedge z \ge 0$ $(*)$ *in Eq.* (1)-(3).

Proof. If Eq. (12) is active, then the maximum objective function value of the QP is const. Because of the max operator, the objective function of the QP cannot attain any larger value. This means that $(*)$ can only be feasible when $-\frac{1}{2}\, y^\top Q y + b^\top z$ attains its maximum over $A^\top z - Qy \le p$ and $z \ge 0$. If Eq. (12) is inactive, it has no effect on the optimum. Since $(*)$ is a relaxation of Eq. (12), the same holds. □

Prop. 1 yields the following reformulation of 1-(3).

$$\min_{\substack{q\in[q^{\mathsf{min}},q^{\mathsf{max}}]\\ z\ge 0, y}} \quad \sum_{i\in A}(q_i-1)^2 \tag{13}$$

$$\forall i<j \in A \quad -\sum_{k=1}^{2}(q_i v_i u_{ik} - q_j v_j u_{jk})^2\, y_{ij}^2 + (-T) z_{ij} \ge d^2 - \sum_{k=1}^{2}(x_{ik}^0 - x_{jk}^0)^2 \tag{14}$$

$$\forall i<j \in A \quad -\frac{z_{ij}}{2} - \sum_{k=1}^{2}(q_i v_i u_{ik} - q_j v_j u_{jk})^2\, y_{ij} \le \sum_{k=1}^{2}(x_{ik}^0 - x_{jk}^0)(q_i v_i u_{ik} - q_j v_j u_{jk}) \tag{15}$$

3 Computational results

We considered the set of instances proposed in [1], where n aircraft are placed on a circle of given radius r, with initial speed v_i and a trajectory defined by a heading angle such that aircraft fly toward the center of the circle (or slightly deviating with respect to such direction). Then we also considered instances in which aircraft move along straight trajectories intersecting in n_c conflict points. We set: $T = 2$ hours, $d = 5\,\text{NM}$, $v_i = 400\,\text{NM/h}$ for each $i \in A$. For the "*circle instances*" the heading angles cap_i are randomly generated and parameters x^0_{ik} and u_{ik} are given by $u_{i1} = \cos(\mathsf{cap}_i)$, $u_{i2} = \sin(\mathsf{cap}_i)$, $x^0_{ik} = -r\,u_{ik}$. The bounds q_i^{min} and q_i^{max} are set to 0.94 and 1.03 respectively. We implemented the proposed formulations using the AMPL modeling language [4] and solved them with the global optimization solver Baron [5] (B in the Table 1) or, when Baron was not successful, with a Multistart algorithm (MS in the Table 1) with 1000 iterations in total. The Multistart method for the KKT reformulation uses SNOPT [6] at each iteration, while the one for the Dual reformulation uses IPOPT [7]. Our results are reported in Table 1, and compared with those that are the best among the ones obtained with different methods in [1] and [8].

Instance			Cafieri	KKT reformulation			Dual reformulation		
n	n_c	r	*obj (Best solution)*	*obj*	*time(s)*	solver	*obj*	*time(s)*	solver
Circle instances									
2	-	100	0.002531	0.002524	0.28	B	0.002526	0.41	B
3	-	200	0.001667	0.001664	1.49	B	0.001663	3.70	B
4	-	200	0.004009	0.004025	65.42	B	0.004017	184.4	B
5	-	300	0.003033	0.003052	12511	B	0.003050	13978	B
6	-	300	0.006033	0.006088	31.99	MS	0.006096	7.84	MS
Non-circle instances									
6	5		0.001295	0.001254	53.31	MS	0.001254	14.88	MS
7	4		0.001617	0.001591	238.82	MS	0.001591	31.17	MS
7	6		0.001579	0.001566	86.95	MS	0.001566	33.18	MS
8	4		0.002384	0.002384	1163	MS	0.002384	39.54	MS
10	10		0.001470	0.001469	835.24	MS	0.001397	78.90	MS

Table 1

Looking at the solutions obtained on these instances, it appears that they are comparable. The value of the objective function is always very low, given the nature of the problem (q must be in $[0.94, 1.03]$). For the bigger instances, the reformulation which ensures the minimum solving time is the Dual one.

References

[1] N. Durand S. Cafieri. Aircraft deconfliction with speed regulation: New models from mixed-integer optimization. *J. of Global Optimization*, 58:613–629, 2014.

[2] W. S. Dorn. Self-dual quadratic programs. *Journal of the Society for Industrial and Applied Mathematics*, 9(1):51–54, 1961.

[3] W. S. Dorn. Duality in quadratic programming. *Quarterly of Applied Mathematics*, 18(2):155–162, 1960.

[4] Robert Fourer, David M. Gay, and Brian W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* 2002.

[5] N.V. Sahinidis and M. Tawarmalani. *BARON 7.2.5: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2005.

[6] P.E. Gill. *User's guide for SNOPT version 7.2.* Systems Optimization Laboratory, Stanford University, California, 2006.

[7] A. Wächterm and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 2006.

[8] C. D'Ambrosio S. Cafieri. Feasibility pump for aircraft deconfliction with speed regulation. *Journal of Global Optimization, Springer Verlag*, 71(3):501–515, 2018.

Linear programming for Decision Processes with Partial Information

Victor Cohen[1] and Axel Parmentier[1]

[1]Université Paris-Est, CERMICS (ENPC), Marne-la-Vallée, France

Abstract

Markov Decision Processes are stochastic optimization problems that model situations where a decision maker controls a system based on its state. Partially observed Markov decision processes (POMDPs) are generalizations of Markov Decision Processes where the decision maker has only partial information on the state of the system. Such problems naturally model a wide range of applications such as predictive maintenance. Finding an optimal policy for a POMDP is PSPACE-hard and practically challenging. We introduce a mixed integer linear programming version for POMDPs where decisions are taken based only on the current observation, as well as valid inequalities that are based on a probabilistic interpretation of the dependence between variables. The linear relaxation provides a good bound for the usual POMDPs where the policies depend on the full history of observations and actions. POMDPs suffer from the curse of dimensionality, and systems composed of multiple components evolving independently but linked by a common action, which are relevant in the context of predictive maintenance, are typically intractable. We introduce decomposable POMDPs and a heuristic to solve them to be able to deal with such system. Numerical experiments show the efficiency of our approach.

1 Introduction

Many real-world problems where a decision maker controls a stochastic system evolving over time can be modeled as *Partially Observed Markov Decision Processes* (POMDPs). In such problems, at each timestep, the system is in a *state* s in some finite state space $\mathcal{X}_S$. The decision maker does not observe s, but has access to an *observation* o that belongs to some finite observation space $\mathcal{X}_O$, and is randomly emitted with probability $p(o|s)$. Based on this observation, the decision-maker chooses an action a from some finite action space $\mathcal{X}_A$. The system then transits randomly to a new state s' in $\mathcal{X}_S$ with probability $p(s'|s,a)$ and the decision maker obtains an immediate reward $r(s,a,s')$. The goal of the decision maker in POMDP problem is to find a policy $\delta^t_{a|o}$, which represents a conditional probability of taking action a in $\mathcal{X}_A$ given observation o in $\mathcal{X}_O$ at time t, maximizing the expected total reward over a finite horizon T

$$\max_{\boldsymbol{\delta}\in\Delta} \mathbb{E}_{\boldsymbol{\delta}}\left[\sum_{t=1}^{T} r(S_t, A_t, S_{t+1})\right], \tag{1}$$

where S_t and A_t are random variables representing the state and the action at time t and the expectation over $\boldsymbol{\delta}$ is taken with respect to the distribution $\mathbb{P}_{\boldsymbol{\delta}}$ induced by the policy $\boldsymbol{\delta}$ chosen in the set of policies Δ. $\boldsymbol{\delta}$ and Δ are defined in section 2. In the POMDP literature, we usually assume that there is *perfect recall*, i.e., the decision is taken given all history of observation and actions at each time step. Hence the policy lies in a greater set of policies $\Delta^{PR} \supset \Delta$. POMDPs

are a generalization of the well-known *Markov Decision Processes* (MDPs). Many applications involve systems composed of several components. In such case, the spaces become too large and usual exact methods become computationally intractable (curse of dimensionality). We introduce *decomposable POMDPs* to model such systems. Our contributions are as follows :

1. We propose a mixed-integer linear program (MILP) that exactly solves Problem (1). This formulation generalizes the usual dual linear program for MDP (see [2]).
2. We introduce an extended formulation with new valid inequalities that improve the resolution of our mixed-integer linear program. Such inequalities come from a probabilistic interpretation of the dependence between random variables. Experiments show their efficiency.
3. Leveraging the MILP previously mentioned, we propose a heuristic policy for decomposable POMDPs.

2 Mixed Linear Programming for Partially Observed Markov Decision Processes

In this section, we present an MILP that exactly models Problem (1). Given N in $\mathbb{Z}^+$ we use the notation $[N]$ for $\{1, \dots, N\}$. Our goal is to solve Problem (1), where

$$\Delta = \left\{ \boldsymbol{\delta} \in \mathbb{R}^{T \times |\mathcal{X}_A| \times |\mathcal{X}_O|}, \sum_{a \in \mathcal{X}_A} \delta^t_{a|o} = 1 \text{ and } \delta^t_{a|o} \geq 0, \forall o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] \right\},$$

and $\mathbb{P}_\delta$ and $\mathbb{E}_\delta$ denote the probability distribution and expectation induced by policy δ on the random variables $(S_t, O_t, A_t)_t$, which respectively denote the state, the observation and the action at time t. We define the set of *deterministic* policies $\Delta^0 = \Delta \cap \{0,1\}^{T \times |\mathcal{X}_A| \times |\mathcal{X}_O|}$. Note that Δ is the convex hull of Δ^0. Any policy in $\Delta \backslash \Delta^0$ is a *randomized* policy.

2.1 A mixed-integer linear program

It is well-known that there always exists an optimal *deterministic* policy for MDPs. This result can be extended to POMDPs [3, e.g. Lemma C.1]. We introduce a collection of variables $\boldsymbol{\mu} = \left((\mu^t_s)_s, (\mu^t_{sa})_{s,a}, (\mu^t_{soa})_{s,o,a}\right)_t$ and the following mixed-integer linear program (MILP).

$$\max_{\boldsymbol{\mu}, \boldsymbol{\delta}} \sum_{t=1}^{T} \sum_{\substack{s,s' \in \mathcal{X}_S \\ a \in \mathcal{X}_A}} r(s,a,s') p(s'|s,a) \mu^t_{sa} \tag{2a}$$

$$\text{s.t. } \mu^1_s = p(s) \qquad \forall s \in \mathcal{X}_S \tag{2b}$$

$$\mu^{t+1}_{s'} = \sum_{s \in \mathcal{X}_S, a \in \mathcal{X}_A} p(s'|s,a) \mu^t_{sa} \qquad \forall s' \in \mathcal{X}_S, t \in [T] \tag{2c}$$

$$\mu^t_{sa} = \sum_{o \in \mathcal{X}_O} \mu^t_{soa} \qquad \forall s \in \mathcal{X}_S, a \in \mathcal{X}_A, t \in [T] \tag{2d}$$

$$\mu^t_{soa} \leq p(o|s) \mu^t_s \qquad \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] \tag{2e}$$

$$\mu^t_{soa} \leq \delta^t_{a|o} \qquad \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] \tag{2f}$$

$$\mu^t_{soa} \geq p(o|s)(\mu^t_s + \delta^t_{a|o} - 1) \qquad \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, t \in [T] \tag{2g}$$

$$\boldsymbol{\delta} \in \Delta^o \tag{2h}$$

$$\boldsymbol{\mu} \geq 0 \tag{2i}$$

We show that a feasible solution $\boldsymbol{\mu}$ of Problem (2) can be interpreted as a probability distribution over the random variables, i.e, μ^t_s, μ^t_{sa} and μ^t_{soa} respectively represent the probabilities $\mathbb{P}_{\boldsymbol{\delta}}\big(S_t = s\big)$, $\mathbb{P}_{\boldsymbol{\delta}}\big(S_t = s, A_t = a\big)$ and $\mathbb{P}_{\boldsymbol{\delta}}\big(S_t = s, O_t = o, A_t = a\big)$ for all s in $\mathcal{X}_S$, o in $\mathcal{X}_O$, a in $\mathcal{X}_A$ and t in $[T]$. Let v^* and z^* respectively denote the optimal values of Problem (1) and Problem (2). The following theorem ensures that MILP (2) models Problem (1).

Theorem 1 *Let $(\boldsymbol{\mu}, \boldsymbol{\delta})$ be a feasible solution of Problem* (2)*. Then $\boldsymbol{\mu}$ is equal to the distribution $\mathbb{P}_{\boldsymbol{\delta}}$ induced by $\boldsymbol{\delta}$, and $(\boldsymbol{\mu}, \boldsymbol{\delta})$ is an optimal solution of Problem* (2) *if, and only if $\boldsymbol{\delta}$ is an optimal deterministic policy of Problem* (1)*. Furthermore, $v^* = z^*$.*

Furthermore, it can be shown that the linear relaxation of MILP (2) gives an upper bound on the value of the perfect recall POMDP. In the next section, we provide valid inequalities that improve this linear relaxation.

2.2 Valid inequalities

The difficulty of POMDP comes from the fact that

$$\text{action variable } A_t \text{ is independent from state } S_t \text{ given observation } O_t, \tag{3}$$

which induces non-linearities. A corollary of these independences is that

$$A_t \text{ is independent from } S_t \text{ given } O_t,\ A_{t-1} \text{ and } S_{t-1}. \tag{4}$$

Theorem 1 ensures that these independences are satisfied by the distribution $\boldsymbol{\mu}$ corresponding to an integer solution $(\boldsymbol{\mu}, \boldsymbol{\delta})$ of MILP (2). If the component $\boldsymbol{\mu}$ of a solution $(\boldsymbol{\mu}, \boldsymbol{\delta})$ of the linear relaxation of MILP (2) can still be interpreted as a distribution, independences (3) and (4) are unfortunately no longer satisfied according to this distribution. We now introduce an extended formulation and valid inequalities that enable to restore independences (4).
We introduce new variables $\mu^t_{s'a'soa}$ that can be interpreted as the probabilities

$$\mathbb{P}(S_{t-1} = s', A_{t-1} = a', S_t = s, O_t = o, A_t = a).$$

Consider the following equalities

$$\sum_{s' \in \mathcal{X}_S, a' \in \mathcal{X}_A} \mu^t_{s'a'soa} = \mu^t_{soa}, \qquad \forall s \in \mathcal{X}_S, o \in \mathcal{X}_O, a \in \mathcal{X}_A, \tag{5a}$$

$$\sum_{a \in \mathcal{X}_A} \mu^t_{s'a'soa} = p(o|s)p(s|s', a')\mu^{t-1}_{s'a'}, \qquad \forall s', s \in \mathcal{X}_S, o \in \mathcal{X}_O, a' \in \mathcal{X}_A, \tag{5b}$$

$$\mu^t_{s'a'soa} = p(s|s', a', o) \sum_{\bar{s} \in \mathcal{X}_S} \mu^t_{s'a'\bar{s}oa}, \qquad \forall s', s \in \mathcal{X}_S, o \in \mathcal{X}_O, a', a \in \mathcal{X}_A, \tag{5c}$$

where

$$p(s|s', a', o) = \mathbb{P}(S_t = s | S_{t-1} = s', A_{t-1} = a', O_t = o).$$

Note that $p(s|s', a', o')$ does not depend on the policy $\boldsymbol{\delta}$ and can be easily computed using Bayes rules. Therefore, constraints in (5) are linear.

Proposition 2 *Equalities* (5) *are valid for Problem* (2)*, and there exists solution $\boldsymbol{\mu}$ of the linear relaxation of* (2) *that does not satisfy constraints* (5)*.*

Hence, constraints (5) strengthen the linear relaxation. Numerical experiments in Section 3 show the efficiency of such valid inequalities.

$\lvert\mathcal{X}_S\rvert$	$\lvert\mathcal{X}_O\rvert$	$\lvert\mathcal{X}_A\rvert$	T	Nb. Policies	Prog.	Int. Gap (%)	Final Gap (%)	Time (s)
3	5	5	10	10^{34}	(2)	8.88	Opt	12
					(2) and (5)	2.61	Opt	3.58
			20	10^{69}	(2)	9.06	2.56	TL
					(2) and (5)	2.45	Opt	116.54
4	8	8	10	10^{72}	(2)	15.50	8.64	TL
					(2) and (5)	1.77	Opt	383.32
			20	10^{144}	(2)	15.64	12.57	TL
					(2) and (5)	1.29	0.62	TL

Table 1: POMDP results using MILP (2) with and without (5), with a time limit TL=3600s

3 Numerical experiments

We now provide experiments showing the practical efficiency of our approaches to POMDPs and decomposable POMDPs. All linear programs have been implemented in Julia with JuMP interface and solved using Gurobi 7.5.2. Experiments have been run on a server with 192Gb of RAM and 32 cores at 3.30GHz. Each instance is generated by first choosing $\lvert\mathcal{X}_S\rvert$, $\lvert\mathcal{X}_O\rvert$, $\lvert\mathcal{X}_A\rvert$. We then randomly generate the initial probability $p(s)$, the transition probability $p(s'|s,a)$, the emission probability $p(o|s)$ and the immediate reward function $r(s,a,s')$. We solve Problem (2) with and without valid equalities (5). Table 1 shows the efficiency of MILP (2) to solve (1). The first four columns indicate the size of state space $\lvert\mathcal{X}_S\rvert$, observation space $\lvert\mathcal{X}_O\rvert$, action space $\lvert\mathcal{X}_A\rvert$ and time horizon T. The fifth and the sixth column indicate respectively the number of possible policies and the mathematical program used to solve Problem (2) with or without constraints (5). In the last three columns, we report the integrity gap, the final gap and the computation time for each instance.

4 Decomposable POMDPs

In many applications, the system is composed of several components. For example, in predictive maintenance, a machine is composed of several deteriorating components and each component evolves independently and individually as a POMDP. Having limited maintenance capacities, the decision maker must at each time step choose which components must be maintained urgently. The decision maker does not have access to the component's wear and takes his action while observing output signals from each component. We introduce decomposable POMDPs to model such problems. Since the size of spaces grows exponentially with the number of component, there is no tractable algorithms to solve it. Leveraging the MILP formulation (2), we propose a heuristic policy for such problems (see [1] for further details).

References

[1] V. Cohen and A. Parmentier. Linear programming for decision processes with partial information, 2018.

[2] F. d'Epenoux. A probabilistic production and inventory problem. *Management Science*, 10(1): 98–108, 1963.

[3] Qiang Liu. *Reasoning and Decisions in Probabilistic Graphical Models–A Unified Framework*. University of California, Irvine, 2014.

Lower bounds for a meal pickup-and-delivery scheduling problem

Matteo Cosmi[1], Gaia Nicosia[1], and Andrea Pacifici[2]

[1]Dipartimento di Ingegneria, Università degli Studi "Roma Tre", via della Vasca Navale 79, 00146 Rome, Italy (e-mail: {matteo.cosmi, gaia.nicosia}@uniroma3.it)
[2]Dipartimento di Ingegneria Civile e Ingegneria Informatica, Università degli Studi di Roma "Tor Vergata", Via del Politecnico 1, 00133 Rome, Italy (e-mail: andrea.pacifici@uniroma2.it)

Introduction

We address a single machine scheduling problem arising in a last mile delivery setting for a food company. In the last few years, due to the recent boost of ubiquitous technologies for the e-commerce, meal delivery services viewed a great transformation which is reflected in the dramatic growth of the food delivery sector as shown by the advances of companies like Deliveroo, GrubHub, or Just-Eat. Concurrently, a considerable share of research has dealt with food delivery logistics. See for instance, [2, 4, 5], just to mention some papers submitted/published in the last year.

The problem we are dealing with in this paper is the following: A set of (prepared) food orders are placed by the customers and are to be fulfilled by the company. For each order a delivery point and a desired delivery time are specified. All food is processed in a single production facility (restaurant) and then immediately carried to the customers by a single courier. The latter is allowed to dispatch *at most two* different orders in a single trip. For each order there is an allowed time window for its delivery which lasts δ time units (min.) centered on the ideal delivery time. An order is considered *on time* if and only if it is delivered within that time window and it is *late* otherwise. Since late deliveries correspond to canceled orders and an economic loss, the company tries to schedule orders so that the number of late orders is minimized.

In this paper, we model the resulting decision problem as a special single machine scheduling problem and present dual bounds which can be exploited in an implicit enumeration scheme like, e.g., a branch and bound algorithm. The performance of these bounds is assessed through a computational study on a set of test instances derived by our real-world application.

Problem description

A given set of meal-orders $J = \{1, \ldots, n\}$ are to be allocated to a (single) restaurant, where the food is prepared. Their ideal delivery times are denoted as d'_j and restaurant-destination travel times t_j are also given, for all $j \in J$.

We first consider the case in which the courier is delivering a single order. The restaurant may be regarded as a depot from which the courier starts when he is shipping, say, the j-th order, and where he will go back immediately after the order has been delivered to the customer. If s is the j-th order pick-up time at the restaurant, then the delivery time is given by $s + t_j$. As we already mentioned, an order is considered on time if it is delivered in an interval of δ minutes centered around the ideal time chosen by the customer, i.e., if

$$d'_j - \frac{\delta}{2} \leq s + t_j \leq d'_j + \frac{\delta}{2}.$$

If the courier—considered a single moving resource—is carrying just one order, he is not available for processing any other order $i \neq j$ until he will be back again to the restaurant. In conclusion, we may view the orders as jobs (tasks) and the single courier as a single machine of a scheduling problem where each task $j \in J$ is associated to the following data:

The *due date* d_j is the latest possible time for the courier to complete the j-th delivery on time and get back to the restaurant and it is therefore $d_j = d'_j + \delta/2 + t_j$. The *release date* r_j is the earliest possible time to start from the restaurant while delivering the j-th order on time to the customer. It can be written as $r_j = d'_j - \delta/2 - t_j$. The *processing time* p_j represents the amount of time the courier is busy with order j and, neglecting possible waiting times at the restaurant and at the customer, it is given by the total travel time $p_j = 2t_j$ of the courier from the restaurant to the corresponding customer destination and back to the restaurant. (We assume the restaurant schedules the required preparations so that the food is ready for immediate pick-up by the courier, for any order $j \in J$.) In this framework, an order j is on time if and only if the corresponding task starts (i.e., the courier picks the food at the restaurant) not before r_j and completes (i.e., the courier returns back to the restaurant and he is available for another trip) not later than d_j.

Finding a sequence of orders to minimize the number of late deliveries is a special case of the single machine scheduling problem with (arbitrary) release dates and due dates $1|r_j|\sum U_j$. The latter problem is, in general, strongly $\mathcal{NP}$-hard. In our problem, as a consequence of the above definitions, we have the relation $d_j = r_j + p_j + \delta$ which makes this scheduling problem a special case of $1|r_j|\sum U_j$ since due dates and release dates are interdependent.

Order aggregation. We are now considering the case in which the courier dispatches two orders in a single trip from the restaurant to the two customers' respective locations and back. We look at this composite service as a special task (in the following equivalently referred to as a *twin order* or, simply, a *twin*). We suppose that after the first delivery the courier immediately proceeds to consign the second order of a twin, so we do not allow the introduction of any idle time between the two deliveries (no-wait assumption). Clearly, a courier may fulfill two orders i and j in a single trip only if it is possible to meet the corresponding delivery-time constraints under the no-wait assumption. In this case, the twin composed by the ordered pair of tasks (i, j) (hereafter, for simplicity, indicated by ij) is called a *feasible twin*, so that it may be regarded as a single aggregated task. Naturally extending the concept of on-time order, we say that a twin order $ij \in D$ is on time if so are both its components.

Given two orders $i, j \in J \times J$, we can express the difference between the time taken to travel from customer i to j and the interval separating their ideal delivery times as

$$\tau_{ij} \stackrel{\text{def}}{=} t_{ij} - (d'_j - d'_i) = t_{ij} - (d_j - d_i) + 1/2(p_j - p_i).$$

It is not hard to show [2] that the twin ij is *feasible*, i.e., *it is possible* to deliver both orders i and j on time, if and only if $\delta_{ij} \stackrel{\text{def}}{=} \delta - |\tau_{ij}| \geq 0$. Therefore, we denote the set of feasible twins by $D = \{ij \ : \ (i, j) \in J \times J, \ \delta_{ij} \geq 0\}$. For notational convenience, we include in D, the special symbol jj (for any $j \in J$) in order to represent a single task j as a twin (but not aggregated to another task).

With some algebra (see [2]), it is easy to derive the release date, the processing time and the due date of a twin ij, which are as follows:

$$r_{ij} \stackrel{\text{def}}{=} \begin{cases} d_i - \delta - p_i \ (= r_i) & \text{if } \tau_{ij} > 0 \\ d_j - \delta - \frac{1}{2}(p_j + p_i) - t_{ij} & \text{if } \tau_{ij} \leq 0 \end{cases} \tag{1}$$

$$p_{ij} \stackrel{\text{def}}{=} t_{ij} + t_i + t_j = t_{ij} + \frac{1}{2}(p_i + p_j) \tag{2}$$

$$d_{ij} \stackrel{\text{def}}{=} r_{ij} + \delta_{ij} + p_{ij}. \tag{3}$$

Observe that, if $\tau_{ij} > 0$, then $d_{ij} = d_j$.

Lower Bounds

Both bounds we are presenting in this section are based on partitioning the set of jobs (order) in a number, say k, of classes $J^{(1)}, \ldots, J^{(k)}$, then summing the contributions of each class to eventually obtain a lower bound which is valid for the whole set of jobs. This is an idea similar to that described in [1] where this concept is exploited to obtain assorted bounds through different methodologies. We devised and tested several techniques to design partitions that return the best possible bounds. In this regard, one of the best procedure works as follows: The planning interval (between the minimum release time and maximum due date) $[r_{\min}, d_{\max}]$ is divided in k time slots. Two jobs are in the same class if and only if their release times are in the same time slot. We then have a partition of the jobs in k classes.

Note that, when computing a lower bound, the twins are not known *a priori*. Hence, we need an estimate from below of a job contribution to the processing time of any possible twin containing that job. Recalling Equation 2 for a twin ij, we may use for job i the reduced duration $p_i/2$.

"Flow based" lower bound. The first bound we describe presents some similarities to the so-called "Flow based" lower bound of [1]. We compute a bound for the ℓ-th class as follows. All jobs $i \in J^{(\ell)}$ must be processed in the time window $[r^{(\ell)}, d^{(\ell)}]$, where $r^{(\ell)} = \min_{i \in J^{(\ell)}}\{r_1\}$ and $d^{(\ell)} = \max_{i \in J^{(\ell)}}\{d_i\}$. It is straightforward to see that if the overall durations of the jobs in $J^{(\ell)}$ exceeds the time window length $d^{(\ell)} - r^{(\ell)}$, then at least one job will be a tardy job. More precisely, if we denote by $i(h, \ell)$ the h-th shortest job in $J^{(\ell)}$ and we have

$$\sum_{h=1}^{|J^{(\ell)}|-m} \frac{1}{2} p_{i(h,\ell)} > d^{(\ell)} - r^{(\ell)},$$

then at least $m + 1$ jobs of $J^{(\ell)}$ are tardy. Clearly, the overall bound is given by summing up the contributions from all $J^{(\ell)}$, for $\ell = 1, \ldots, k$.

The LHS of the above inequality may actually be increased by a factor that takes into account travel times among clients (which contribute in the overall twin processing times). A lower estimate of this factor can be computed as the minimum cost of a perfect matching on the undirected graph[1] where vertices correspond to J, edges to D, and the costs of edge ij is t_{ij}.

Kise-Ibaraki-Mine lower bound. Analogous to the previous bound, we recur to the job set partition $J^{(1)}, \ldots, J^{(k)}$. We compute a bound for the ℓ-th class by using a relaxation of the sub-problem that makes the jobs of $J^{(\ell)}$ *agreeable* which implies that for any two jobs i and j such that $r_i < r_j$ then $d_i \leq d_j$. This way, we are allowed to use the Kise-Ibaraki-Mine algorithm [3] to optimally solve the relaxation.

In order to determine the above relaxation, we basically enlarge the allotted intervals $[r_j, d_j]$. Finding an effective way to do so (i.e., determine the minimum total enlargement so that the jobs of $J^{(\ell)}$ become agreeable) is an interesting problem itself that can be solved in linear time.

Preliminary Computational Results

Hereafter we sketch some preliminary results of a set of experiments aimed at comparing the quality of the two proposed lower bounds. The tests were run on 177 real-life instances (derived from the test set used [2]) in which the number of orders n varies from 5 to 31, the optimal solution value is

[1] Depending on whether $|J|$ is odd or even, the vertex set may include the restaurant or not.

known and strictly positive for 143 of those instances. A state-of-the-art solver (namely, Gurobi) is not able to optimally solve (natural MIP models for) the remaining 34 cases. Algorithms to compute lower bounds were built using the Julia programming language and all tests were performed on a computer equipped with an Intel Xeon E5-2643v3 3.40 GhZ CPU and 32 GB RAM.

A first comparison against the lower bounds provided by Gurobi shows that our bounds, although not tight in most of the instances, are computed much faster and, in $91,18\%$ of the cases, they outperform those bestowed by the solver even after 5 minutes of running time.

Table 1 reports the lower bounds values for the 143 optimally solved instances. Each of the last three rows refers to a set of instances having the same optimal solution value, which is reported in the first column. The second column records the number of instances with those solution value. Columns 3–5 give the number of instances in which the "flow based" lower bound value is equal to 1, 2, or 3. (For example, among the 8 instances with optimal solution value equal to 3, three instances have LB = 2 and five instances have LB = 1). Columns 6–8 report the same data for the Kise-Ibaraki-Mine lower bound, and the last three columns the best bounds between the two.

Table 1: Lower Bounds and Optimal values

OPT val.	# Inst	LB_{FB}			LB_{KIM}			$BestLB$		
		1	2	3	1	2	3	1	2	3
1	104	7	-	-	5	-	-	9	-	-
2	31	20	0	-	10	2	-	18	2	-
3	8	5	3	0	3	3	0	5	3	0

References

[1] P. Baptiste, L. Peridy, E. Pinson. A branch and bound to minimize the number of late jobs on a single machine with release time constraints, European Journal of Operational Research 144(1), 1–11 (2003).

[2] M. Cosmi, G. Nicosia, A. Pacifici. Scheduling for last-mile meal-delivery processes, 9th IFAC conference MIM 2019 on Manufacturing Modelling, Management, and Control, Berlin, Germany, August 2019.

[3] S. Li, Z. Chen, T. Guochun. Optimality proof of the Kise–Ibaraki–Mine algorithm, Journal of Scheduling 15(3), 289–294 (2012).

[4] D. Reyes, A. Erera, M. Savelsbergh, S. Sahasrabudhe, R. O'Neil. The Meal Delivery Routing Problem, Optimization-Online, 2018.

[5] B. Yıldız, M. Savelsbergh. Provably high-quality solutions for the meal delivery routing problem, Transportation Science, Optimization-Online, 2018.

From Hotelling to Load Balancing: Approximation and the Principle of Minimum Differentiation

Matthias Feldotto[1], Pascal Lenzner[2], Louise Molitor[3], and Alexander Skopalik[4]

[1]Paderborn University
[2,3]Hasso Plattner Institute
[4]University of Twente

Abstract

Competing firms tend to select similar locations for their stores. This phenomenon, called the principle of minimum differentiation, was captured by Hotelling with a landmark model of spatial competition but is still the object of an ongoing scientific debate. Although consistently observed in practice, many more realistic variants of Hotelling's model fail to support minimum differentiation or do not have pure equilibria at all. In particular, it was recently proven for a generalized model which incorporates negative network externalities and which contains Hotelling's model and classical selfish load balancing as special cases, that the unique equilibria do not adhere to minimum differentiation. Furthermore, it was shown that for a significant parameter range pure equilibria do not exist.

We derive a sharp contrast to these previous results by investigating Hotelling's model with negative network externalities from an entirely new angle: approximate pure subgame perfect equilibria. This approach allows us to prove analytically and via agent-based simulations that approximate equilibria having good approximation guarantees and that adhere to minimum differentiation exist for the full parameter range of the model. Moreover, we show that the obtained approximate equilibria have high social welfare.

Introduction The choice of a profitable facility location is one of the core strategic decisions for firms competing in a spatial market. Finding the right location is a classical object of research and has kindled the rich and interdisciplinary research area called Location Analysis [32, 14, 6]. In this paper we investigate one of the landmark models of spatial competition and strategic product differentiation where facilities offering the same service for the same price compete in a linear spatial marked. Originally introduced by Hotelling [24] and later extended by Downs [10] to model political competition, the model is usually referred to as the *Hotelling-Downs model*. It assumes a market of infinitely many clients which are distributed evenly on a line and finitely many firms which want to open a facility and which strategically select a specific facility location in the market to sell their service. Every client wants to obtain the offered service and selects the nearest facility to get it. The utility of the firms is proportional to the number of clients visiting their facility. Thus, the location decision of a firm depends on the facility locations of all its competitors as well as on the anticipated behavior of the clients. This two-stage setting is challenging to analyze but at the same time yields a plausible prediction of real-world phenomena.

One such phenomenon is known as the *principle of minimum differentiation* [5, 13] and it states that competing firms selling the same service tend to co-locate their facilities instead of spreading them evenly along the market. This can be readily observed in practice, e.g., stores of different fast-food chains or consumer goods shops are often located right next to each other. For the original version where clients simply select the nearest facility, Eaton and Lipsey [13] proved in a seminal paper that for $n \neq 3$ competing firms the Hotelling-Downs model has pure subgame perfect equilibria which respect the principle of minimum differentiation.

However, the original Hotelling-Downs model is overly simple. A more realistic variant, where the cost function of a client is a linear combination of distance and waiting time, was proposed by Kohlberg [25] and will be the focus of our attention. Kohlberg's model is especially interesting, since it can be interpreted as an interpolation of two extreme models: the Hotelling-Downs model, where clients select the nearest facility and classical *Selfish Load Balancing* [39], where clients select the least congested facility.

For Kohlberg's model it is known that no pure subgame perfect equilibria exist where the facility locations are pairwise different. Furthermore, in a recent paper Peters et al. [31] show for up to six facilities that pure

equilibria exist if and only if there is an even number of facilities and the clients' cost function is tilted heavily towards preferring less congested facilities. Moreover, in sharp contrast to the principle of minimum differentiation, they show that in these unique equilibria only two facilities are co-located.

In this paper we re-establish the principle of minimum differentiation for Kohlberg's model by considering *approximate pure subgame perfect equilibria*. We believe that in contrast to studying exact subgame perfect equilibria, investigating approximate subgame perfect equilibria yields more reliable predictions since the study of exact equilibria assumes actors who radically change their current strategy even if they can improve only by a tiny margin. In the real world this is not true, as many actors only move out of their "comfort zone" if a significant improvement can be realized. This threshold behavior can naturally be modeled via a suitably chosen approximation factor. Furthermore, approximate equilibria are the only hope for a plausible prediction for many variants of the Hotelling-Downs model where exact equilibria do not exist. To the best of our knowledge, approximate equilibria have not been studied before in the realm of Location Analysis.

Related Work The Hotelling-Downs model was also analyzed for non-linear markets, e.g., on graphs [30, 20, 19], fixed locations on a circle [34], finite sets of locations [27, 28], and optimal interval division [36]. Moreover, many facility location games are variants of the Hotelling-Downs model and there is a rich body of work analyzing them [38, 7, 33, 11, 21, 18, 15] and Vornoi games [2, 12, 3].

In our model facilities offer their service for the same price. Models where facilities can also strategically set the price have been considered [9, 26, 22, 8, 29, 23, 1]. Other recent work investigates different client attraction functions instead of simply using the distance to the facilities [4, 16, 35].

Using agent-based simulations for variants of the Hotelling-Downs model seems to be a quite novel approach. We could find only the recent work of van Leeuwen & Lijesen [37] in which the authors claim to present the first such approach. They study a multi-stage variant with pricing which is different from our setting.

Our Contribution We study approximate pure subgame perfect equilibria in Kohlberg's model of spatial competition with negative network externalities in which n facility players strategically select a location in a linear market. Our slightly reformulated model has a parameter $0 \leq \alpha \leq 1$, where $\alpha = 0$ yields the original Hotelling-Downs model, i.e., clients select the nearest facility, and where $\alpha = 1$ yields classical selfish load balancing, i.e., clients select the least congested facility.

First, we study the case $n = 3$, which for $\alpha = 0$ is the famous unique case of the Hotelling-Downs model where exact equilibria do not exist. We show that for all α an approximate subgame perfect equilibrium exists with approximation factor $\rho \leq 1.2808$. Moreover, for $\alpha = 0$ we show that this bound is tight.

Next, we consider the facility placement which is socially optimal for the clients and analyze its approximation factor, i.e., we answer the question how tolerant the facility players have to be to accept the social optimum placement for the clients. For this placement, in which the facilities are uniformly distributed along the linear market, we derive exact analytical results for $4 \leq n \leq 10$. Building on this and on a conjecture specifying the facility which has the best improving deviation, we generalize our results to $n \geq 4$. We find that the obtained approximation factor ρ approaches 1.5 for low α which implies that in these cases facility players must be very tolerant to support these client optimal placements (cf. Fig. 1).

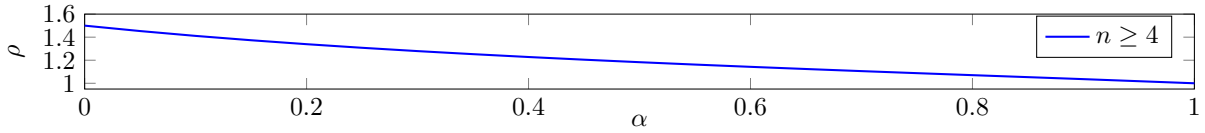


Figure 1: Results for client-optimal facility placement.

We contrast this by our main contribution, which is a study of a facility placement proposed by Eaton & Lipsey [13] (cf. Fig. 2) from an approximation perspective. This placement supports the principle of minimum differentiation since all but at most one facility are co-located with another facility and at the same time it is an exact equilibrium for both extreme cases of the model, i.e., for $\alpha = 0$ and $\alpha = 1$. We provide analytical proofs that for these placements $\rho \leq 1.0866$ holds for $4 \leq n \leq 10$. Also, based on another conjecture, we show that for arbitrary even $n \geq 10$ we get $\rho \approx 1.08$ (cf. Fig. 3).
Our conjectures used for proving the general results are based on the analytical results for $n \leq 10$ and on extensive agent-based simulations of a discretized variant of the model. It turns out that these simulations

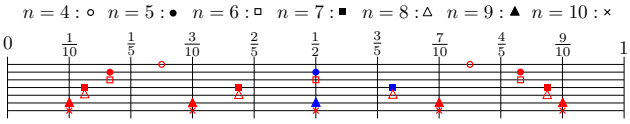


Figure 2: Facility placements from [13] for $4 \leq n \leq 10$. Co-located facilities are red, single facilities are colored blue.

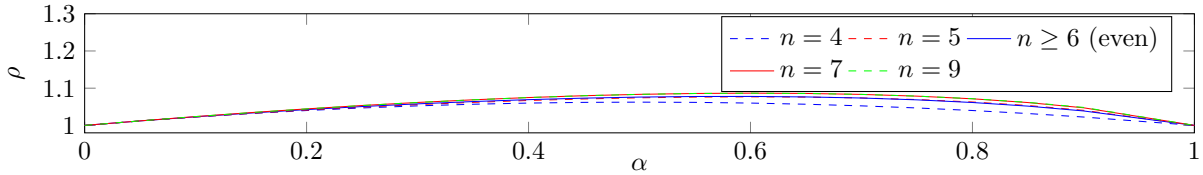


Figure 3: Results for facility placements from [13].

yield reliable predictions for the original model and we also use them for providing promising results for the general case with odd n. In particular, we demonstrate that empirically we have $\rho \approx 1.08$ for arbitrary $n \geq 10$.

Last but not least, we show that the facility placements proposed by Eaton & Lipsey [13] are also socially good for the clients. We compare their social cost with the cost of the social optimum placement and prove a low ratio for all α.

All omitted details can be found in the full version [17].

Conclusion We demonstrate that for Kohlberg's model facility placements exist which adhere to the principle of minimum differentiation, are close to stability in the sense that facilities can only improve their utility by at most 8% by deviating and these placements are also socially beneficial for all clients. This remarkble contrast to the results of Peters et al. [31] indicates that studying approximate equilibria may yield more realistic results than solely focusing on exact equilibria. Moreover, investigating approximate equilibria may also lead to new insights for other models in the realm of Location Analysis.

References

[1] C. Ahlin and P. D. Ahlin. Product differentiation under congestion: Hotelling was right. *Economic Inquiry*, 51(3):1750–1763, 2013.

[2] H. Ahn, S. Cheng, O. Cheong, M. J. Golin, and R. van Oostrum. Competitive facility location: the voronoi game. *Theor. Comput. Sci.*, 310(1-3):457–467, 2004.

[3] S. Bandyapadhyay, A. Banik, S. Das, and H. Sarkar. Voronoi game on graphs. *Theor. Comput. Sci.*, 562:270–282, 2015.

[4] O. Ben-Porat and M. Tennenholtz. Shapley facility location games. In N. R. Devanur and P. Lu, editors, *WINE'17*, volume 10660 of *LNCS*, pages 58–73. Springer, 2017.

[5] K. E. Boulding. *Economic analysis*. Harper and brothers Publishers, London, 1941.

[6] S. Brenner. *Location (Hotelling) Games and Applications*. American Cancer Society, 2011.

[7] J. Cardinal and M. Hoefer. Non-cooperative facility location and covering games. *Theor. Comput. Sci.*, 411(16-18):1855–1876, 2010.

[8] C. d'Aspremont, J. J. Gabszewicz, and J.-F. Thisse. On hotelling's "stability in competition". *Econometrica*, 47(5):1145–1150, 1979.

[9] A. de Palma and L. Leruth. Congestion and game in capacity: A duopoly analysis in the presence of network externalities. *Annales d'Economie et de Statistique*, (15/16):389–407, 1989.

[10] A. Downs. An economic theory of political action in a democracy. *Journal of Political Economy*, 65(2):135–150, 1957.

[11] M. Drees, B. Feldkord, and A. Skopalik. Strategic online facility location. In *International Conference on Combinatorial Optimization and Applications*, pages 593–607. Springer, 2016.

[12] C. Dürr and N. K. Thang. Nash equilibria in voronoi games on graphs. In L. Arge, M. Hoffmann, and E. Welzl, editors, *Algorithms - ESA 2007, 15th Annual European Symposium, October 8-10, 2007, Proceedings*, volume 4698 of *Lecture Notes in Computer Science*, pages 17–28. Springer, 2007.

[13] B. Eaton and R. Lipsey. The principle of minimum differentiation reconsidered: Some new developments in the theory of spatial competition. 42:27–49, 02 1975.
[14] H. A. Eiselt, G. Laporte, and J. Thisse. Competitive location models: A framework and bibliography. *Transportation Science*, 27(1):44–54, 1993.
[15] M. Feldman, A. Fiat, and I. Golomb. On voting and facility location. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 269–286. ACM, 2016.
[16] M. Feldman, A. Fiat, and S. Obraztsova. Variations on the hotelling-downs model. In D. Schuurmans and M. P. Wellman, editors, *AAAI'16*, pages 496–501. AAAI Press, 2016.
[17] M. Feldotto, P. Lenzner, L. Molitor, and A. Skopalik. From hotelling to load balancing: Approximation and the principle of minimum differentiation. *CoRR*, abs/1903.04265, 2019.
[18] D. Fotakis and C. Tzamos. On the power of deterministic mechanisms for facility location games. *ACM Transactions on Economics and Computation*, 2(4):15, 2014.
[19] G. Fournier. General distribution of consumers in pure hotelling games. *CoRR*, abs/1602.04851, 2016.
[20] G. Fournier and M. Scarsini. Location games on networks: Existence and efficiency of equilibria. *CoRR*, abs/1601.07414, 2016.
[21] M. X. Goemans and M. Skutella. Cooperative facility location games. *J. Algorithms*, 50(2):194–214, 2004.
[22] I. Grilo, O. Shy, and J.-F. Thisse. Price competition when consumer behavior is characterized by conformity or vanity. *Journal of Public Economics*, 80(3):385 – 408, 2001.
[23] T. Heikkinen. A spatial economic model under network externalities: symmetric equilibrium and efficiency. *Operational Research*, 14(1):89–111, 2014.
[24] H. Hotelling. Stability in competition. *The Economic Journal*, 39(153):41–57, 1929.
[25] E. Kohlberg. Equilibrium store locations when consumers minimize travel time plus waiting time. *Economics Letters*, 11(3):211 – 216, 1983.
[26] A. Navon, O. Shy, J.-F. Thisse, et al. *Product differentiation in the presence of positive and negative network effects*. Centre for Economic Policy Research, 1995.
[27] M. Núñez and M. Scarsini. Competing over a finite number of locations. *Economic Theory Bulletin*, 4(2):125–136, Oct. 2016.
[28] M. Núñez and M. Scarsini. *Large Spatial Competition*, pages 225–246. Springer International Publishing, Cham, 2017.
[29] M. J. Osborne and C. Pitchik. The nature of equilibrium in a location model. *International Economic Review*, pages 223–237, 1986.
[30] D. Pálvölgyi. Hotelling on graphs. In *URL http://media. coauthors. net/konferencia/conferences/5/palvolgyi. pdf. Mimeo*, 2011.
[31] H. Peters, M. Schröder, and D. Vermeulen. Hotelling's location model with negative network externalities. *International Journal of Game Theory*, Feb. 2018.
[32] C. S. Revelle and H. A. Eiselt. Location analysis: A synthesis and survey. *European Journal of Operational Research*, 165(1):1–19, 2005.
[33] D. Sabán and N. S. Moses. The competitive facility location problem in a duopoly: Connections to the 1-median problem. In P. W. Goldberg, editor, *Internet and Network Economics - 8th International Workshop, WINE 2012, Liverpool, UK, December 10-12, 2012. Proceedings*, volume 7695 of *Lecture Notes in Computer Science*, pages 539–545. Springer, 2012.
[34] S. C. Salop. Monopolistic competition with outside goods. *The Bell Journal of Economics*, 10(1):141–156, 1979.
[35] W. Shen and Z. Wang. Hotelling-downs model with limited attraction. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 660–668. International Foundation for Autonomous Agents and Multiagent Systems, 2017.
[36] J. Tian. Optimal interval division. 2015.
[37] E. van Leeuwen and M. Lijesen. Agents playing hotellings game: an agent-based approach to a game theoretic model. *The Annals of Regional Science*, 57(2-3):393–411, 2016.
[38] A. Vetta. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 416. IEEE Computer Society, 2002.
[39] B. Vöcking. Selfish load balancing. *Algorithmic game theory*, 20:517–542, 2007.

Lower Bound Computations for the Nonnegative Rank

Samuel Fiorini[1], Krystal Guo[1], Marco Macchia[1], and Matthias Walter[2]

[1]Université libre de Bruxelles, Brussels, Belgium
[2]University of Twente, Enschede, Netherlands

Abstract

The nonnegative rank of a matrix is the smallest inner dimension when writing it as a product of two nonnegative matrices. Such nonnegative matrix factorizations have numerous applications in machine learning and data mining, where one usually allows inexact factorizations. The exact counterpart is related to the so-called extension complexity of a polytope, an important parameter in combinatorial optimization.

We implemented different algorithms for computing lower bounds on the nonnegative rank of a matrix. In this extended abstract we focus on results that relate our best algorithms' performance to the size of the matrix.

1 Introduction

Let $M \in \mathbb{R}^{m\times n}_{\geq 0}$ be a nonnegative matrix. A *nonnegative factorization* of M is a product $M = X \cdot Y$ with $X \in \mathbb{R}^{m\times r}_{\geq 0}$ and $Y \in \mathbb{R}^{r\times n}_{\geq 0}$, where we call r the *rank of the factorization*. Expressing a given data matrix as $M \approx X \cdot Y$ with a small r is the (approximate) nonnegative matrix factorization problem. The exact version of the problem is to compute the *nonnegative rank* $\mathrm{rk}_+(M)$ of M, defined as the smallest such r with $M = X \cdot Y$. This quantity has an important interpretation in polyhedral combinatorics [1], which we briefly outline.

The *extension complexity* of a polytope P is the smallest number of facets of any (typically higher-dimensional) polytope Q that affinely projects to P. The number of facets is equal to the (minimum) number of inequalities that are necessary to describe Q in a linear programming (LP) formulation. Suppose we have an inner description of $P \subseteq \mathbb{R}^d$ in terms of its vertices $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ as well as an outer description in terms of linear inequalities $Ax \leq b$ with $A \in \mathbb{R}^{m\times d}$. Then the corresponding slack matrix $M \in \mathbb{R}^{m\times n}_{\geq 0}$ has entries $M_{i,j} \coloneqq b_i - A_{i,*}x_j$. Yannakakis [8] proved that $\mathrm{rk}_+(M)$ is equal to the extension complexity of P. Since its computation is notoriously hard, in our paper we deal with the computation of lower bounds.

2 Lower bounds

Combinatorial bounds. Let $S \coloneqq \mathrm{supp}(M) \coloneqq \{(i,j) \in [m] \times [n] : M_{i,j} > 0\}$ denote the support of M. We call an index set $R \subseteq [m] \times [n]$ a *rectangle* if $R \subseteq S$ and if it is of the form $R = I \times J$ for $I \subseteq [m]$ and $J \subseteq [n]$. We denote by $\mathcal{R}$ the set of all rectangles. Every nonnegative factorization with inner dimension r induces a covering of S with r rectangles (see [8, 2]), which establishes the *rectangle covering bound* $\mathrm{rc}(M)$. It can be computed using the integer program (IP)

$$\min_{\lambda \in \mathbb{Z}^{\mathcal{R}}_{\geq 0}} \left\{ \sum_{R\in\mathcal{R}} \lambda_R : \sum_{R\in\mathcal{R}:\, s\in R} \lambda_R \geq 1 \ \forall s \in S \right\}. \tag{rc}$$

Since this is a minimization problem, the optimum value of its LP relaxation, called the *fractional rectangle covering bound* and denoted by $\mathrm{frc}(M)$, is also a lower bound on $\mathrm{rk}_+(M)$.

Both bounds can be refined slightly by considering entry pairs $s_1, s_2 \in S$ with $s_1 = (i_1, j_1)$ and $s_2 = (i_2, j_2)$. We call $\{s_1, s_2\}$ a *fooling pair* if $M_{i_1,j_1} \cdot M_{i_2,j_2} > M_{i_1,j_2} \cdot M_{i_2,j_1} > 0$ holds and denote by $\mathcal{P}$ the set of all fooling pairs. Although $\{i_1, i_2\} \times \{j_1, j_2\}$ is a rectangle, its induced submatrix of M has rank 2. Thus, at least one of the entries must be covered by two rectangles in any rectangle covering induced by a nonnegative factorization of M. This observation was made in [6] and establishes the *refined rectangle covering bound* $\mathrm{rrc}(M)$, defined via the IP

$$\min_{\lambda \in \mathbb{Z}^{\mathcal{R}}_{\geq 0}} \left\{ \sum_{R \in \mathcal{R}} \lambda_R : \sum_{R \in \mathcal{R}:\, s \in R} \lambda_R \geq 1 \ \forall s \in S, \sum_{R \in \mathcal{R}:\, s_1 \in R \vee s_2 \in R} \lambda_R \geq 2 \ \forall \{s_1, s_2\} \in \mathcal{P} \right\}. \tag{rrc}$$

Once again we define the bound based on the value of the its LP relaxation $\mathrm{frrc}(M)$, called the *fractional refined rectangle covering bound*.

Another combinatorial bound is the *fooling set bound* $\mathrm{fs}(M)$ that is derived from the packing counterpart of the rectangle covering problem It is defined as

$$\max \{|F| : F \subseteq S \text{ such that } |F \cap R| \leq 1 \text{ for all } R \in \mathcal{R}\}, \tag{fs}$$

and we call the feasible sets $F \subseteq S$ *fooling sets*. It is equal to the maximum size of a clique in an auxiliary graph in which nodes correspond to entries of S and two nodes are connected by an edge if and only if their entries are not contained in a rectangle.

Hyperplane separation bounds. In contrast to the combinatorial bounds the *hyperplane separation bound* also exploits the actual entries of M. It was suggested by Fiorini and used to establish an exponential lower bound on the extension complexity of the perfect matching polytope [7]. For its correctness we refer to Lemma 1 therein. It reads

$$\mathrm{hsb}(M) \coloneqq \max \left\{ \frac{\langle W, M \rangle}{||M||_\infty} : W \in \mathbb{R}^{m \times n}, \ \langle W, \chi(R) \rangle \leq 1 \text{ for all } R \in \mathcal{R} \right\}, \tag{hsb}$$

where $\chi(R) \in \{0,1\}^{m \times n}$ is the characteristic vector of R. Note that the entries $W_{i,j}$ with $(i,j) \notin S$ do not play a role in the model. By restricting $W_{i,j}$ to be nonnegative for all $(i,j) \in S$, we obtain the weaker version, called the *nonnegative hyperplane separation bound* and denoted by $\mathrm{nhsb}(M)$.

3 Computational study

In our software tool `nonnegrank` [3] we implemented all lower bounds from Section 2 (see Table 1). Our code is written in `C++` and relies on different external libraries. Several bound computations can be made more efficient in two ways. First, if the input matrix M has some symmetry, then the computational effort can often be reduced by reducing the dimension of the problem. Second, if the set $\mathcal{R}^{\max} \subseteq \mathcal{R}$ of inclusion-wise maximal rectangles is available explicitly, then several subproblems can be solved more quickly. Our software can compute $\mathcal{R}^{\max}$ in time polynomial in $|S|$ and $|\mathcal{R}^{\max}|$ using Ganter's Next Closure Algorithm [4]. Finally, in case this enumeration shall be avoided, we provide an implementation of (rc) and (rrc) using branch-and-price.

Table 1: Implemented bounds with their used external libraries, depending on availability of $\mathcal{R}^{\max}$ (column **Enum**) and information whether symmetry can be exploited (column **Symmetry**).

Bound	**Enum**	**Implementation [libraries]**	**Symmetry**
fs	No	Graph model for (fs) [`Cliquer`] or IP model [`Gurobi` or `SCIP`]	No
rc	Yes	Model (rc) over $\mathcal{R}^{\max}$ [`Gurobi` or `SCIP`]	No
	No	Branch-and-price for Model (rc) [`SCIP`]	No
rrc	Yes	Model (rrc) over $\mathcal{R}^{\max}$ [`Gurobi` or `SCIP`]	No
	No	Branch-and-price for Model (rrc) [`SCIP`]	No
frc	Yes	LP of Model (rc) over $\mathcal{R}^{\max}$ [`Gurobi` or `SoPlex`]	Yes
	No	LP of Model (rc) [`Gurobi` or `SCIP`]	Yes
frrc	Yes	LP of Model (rrc) over $\mathcal{R}^{\max}$ [`Gurobi` or `SoPlex`]	Yes
	No	LP of Model (rrc) [`Gurobi` or `SCIP`]	Yes
hsb	Yes/No	Model (hsb) [`Gurobi` or `SCIP`]	Yes
nhsb	Yes	Model (hsb) with $W_s \geq 0\ \forall s \in S$ over $\mathcal{R}^{\max}$ [`Gurobi` or `SoPlex`]	Yes
	No	Model (hsb) with $W_s \geq 0\ \forall s \in S$ [`Gurobi` or `SCIP`]	Yes

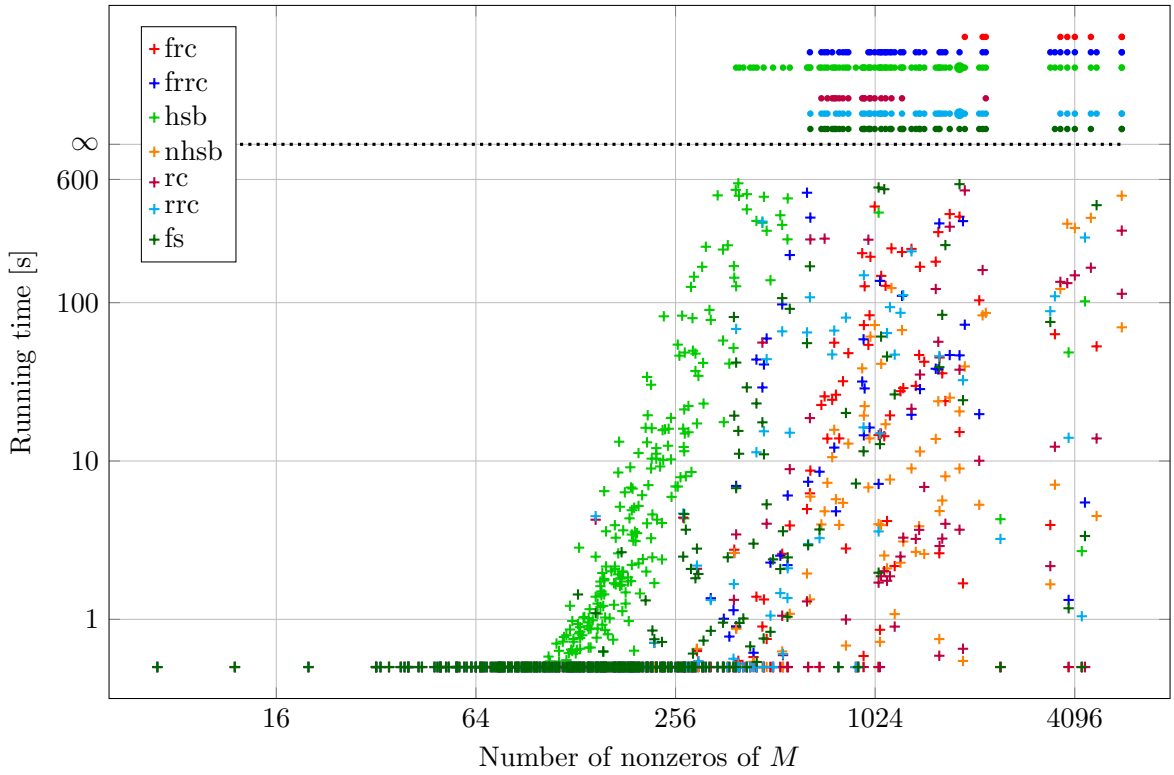


Figure 1: Running times for various bound computations. Time limit: 600 s. Balls above dotted line indicate number of timeouts. IP/LP solver was `Gurobi`, rectangle enumeration turned on, symmetry detection turned off.

Setup. In this extended abstract we report results only for two types of instances, both derived from slack matrices of polytopes with dimensions 6, 7 or 8. The first are random 0/1-polytopes with prescribed ambient dimension and expected number of vertices. The second set is derived from 2-level polytopes randomly chosen from the 2-level polytope database from [5]. Our test were run single threaded on a machine with 2.97 GHz and 8 GB memory.

Experimental results. For the sake of this extended abstract, we restrict ourselves to considering computational results corresponding to the aggregated dataset. We set a time limit of 600 s and use `Gurobi` as the undelying solver, since `SCIP` turned out to be generally slower on our instances. Moreover, we do not report detailed results for symmetry breaking algorithms since they were generally faster in case symmetry was present. On the one hand, we notice that slack matrices of 2-level polytopes exhibit considerable symmetries. On the other hand slack matrices of 0/1-polytopes did not present symmetries. When possible, we use the complete enumeration of rectangles in $\mathcal{R}^{\max}$ and provide the corresponding reduced running times. Figure 1 shows the resulting computation times as a function of the size of the support of the input nonnegative matrix. Moreover, it is easy to see that the computation of hsb is the most time-consuming, the reason being that a cutting plane method has to be adopted (even when $\mathcal{R}^{\max}$ is computed). It is interesting to notice that rc can be computed within the time limit of 600 s for many instances with support of size greater than 1000, even though it requires to solve an IP. The slightly more complicated bound rrc is harder to compute, involving contraints corresponding to fooling pairs.

Future work. In the full version of this abstract, we plan to have a detailed comparison of the lower bounds in Section 2, in order to provide a complete evaluation of the quality of the bounds, and compare it with the running times.

Acknowledgements. The first, the second and the third authors are supported by ERC grant *FOREFRONT* (grant agreement no. 615640) funded by the European Research Council under the EU's 7th Framework Programme (FP7/2007-2013).

References

[1] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Extended formulations in combinatorial optimization. *Annals of Operations Research*, 204(1):97–143, 2013.

[2] Samuel Fiorini, Volker Kaibel, Kanstantsin Pashkovich, and Dirk O. Theis. Combinatorial bounds on nonnegative rank and extended formulations. *Discrete Mathematics*, 313(1):67–83, 2013.

[3] Samuel Fiorini, Marco Macchia, Krystal Guo, and Matthias Walter. nonnegrank – a tool to compute lower bounds on the nonnegative rank, 2019. Software page: `https://bitbucket.org/matthias-walter/nonnegrank/`.

[4] Bernhard Ganter and Klaus Reuter. Finding all closed sets: A general approach. *Order*, 8(3):283–290, 1991.

[5] Marco Macchia. *Two level polytopes: geometry and optimization.* PhD thesis, Université libre de Bruxelles, 2018.

[6] Michael Oelze, Arnaud Vandaele, and Stefan Weltge. Computing the extension complexities of all 4-dimensional 0/1-polytopes. arXiv:1406.4895, June 2014.

[7] Thomas Rothvoss. The matching polytope has exponential extension complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 263–272, New York, NY, USA, 2014. ACM.

[8] Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. *Journal of Computer and System Sciences*, 43(3):441–466, 1991.

The Graceful Game*

Luisa Frickes[1], Simone Dantas[1], and Atílio G. Luiz[2]

[1]IME, Universidade Federal Fluminense, Niterói, Rio de Janeiro, Brazil
[2]Universidade Federal do Ceará, Quixadá, Ceará, Brazil

Abstract

A graceful labeling of a graph G with m edges consists of labeling the vertices of G with distinct integers from 0 to m such that, when each edge is assigned as induced label the absolute difference of the labels of its endpoints, all induced edge labels are distinct. The Graceful game was introduced by Tuza in 2017 as a two-players game on a connected graph in which Alice and Bob take turns labeling the vertices with distinct integers from 0 to m. Alice's goal is to gracefully label the graph as Bob's goal is to prevent it from happening. In this work, we study winning strategies for Alice and Bob in complete graphs, paths, cycles, complete bipartite graphs, caterpillars, prisms, wheels and hypercubes.

1 Introduction

Graph labelling is an area of graph theory that has been gaining a particular importance since the 1960's. The main concern in this area consists in determining the feasibility of assign labels (usually numbers) to the elements of a graph satisfying certain conditions. However, the idea of assigning symbols other than numbers to the elements of a graph is not recent. For example, an old and very studied problem in graph theory is the vertex coloring problem, which consists in determining the least number of colors needed to color the vertices of a given graph such that any two adjacent vertices receive distinct colors. Vertex colorings arose in connection with the well known Four Color Conjecture, which remained open for more than 150 years until its solution in 1976 [1].

In the last decades, many contexts have emerged where it is required to label the vertices or the edges of a given graph with numbers. Most of these problems, such as harmonious labelings [10] and L(2,1)-labelings [11], emerged naturally from modeling of optimization problems on networks. Formally, given a graph G and a set $L \subset \mathbb{R}$, a *labeling* of G is a vertex labeling $f\colon V(G) \to L$ that induces an edge labeling $g\colon E(G) \to \mathbb{R}$ in the following way: $g(uv)$ is a function of $f(u)$ and $f(v)$, for all $uv \in E(G)$, and g respects some specified restrictions.

One of the most studied graph labelings is the *graceful labeling*, so named by S. W. Golomb [9] and initially introduced by A. Rosa [15] in 1996. A *graceful labeling* of a graph G with m edges is an injective function $f\colon V(G) \to \{0, 1, \ldots, m\}$ such that, when each edge $uv \in E(G)$ is assigned the *(induced) label* $g(uv) = |f(u) - f(v)|$, all induced edge labels are distinct, that is, the (induced) edge labeling g is also injective. Graph G is called *graceful* when it has a graceful labeling. The most studied open problem related to graceful labelings is the *Graceful Tree Conjecture*, which states that all trees are graceful [15]. For a comprehensive list of old and recent results on graceful labelings, the reader is referred to Gallian's dynamic survey [7].

*This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq and FAPERJ.

From the vast literature of Graph Labeling (more than two thousand papers [7]), it is notorious that labeling problems are usually studied from the perspective of determining whether a given graph has a required labeling or not. An alternative perspective is to analyze labeling problems from the point of view of combinatorial games. The study of combinatorial games is a classical area in both discrete mathematics and game theory [3]. In most combinatorial games, two players — traditionally called Alice and Bob — alternately select and label vertices or edges (typically one vertex or edge in each step) in a graph G which is completely known for both players.

Only a few papers have been published on labeling games so far [2, 4, 5, 8, 12, 16]. In a recent survey, Z. Tuza [16] survey the area and proposes new labeling games such as the graceful game studied in this work. While the number of articles published in labeling games has been scarce, in the related area of graph colorings, there is a track of research concerning 'game chromatic number' that has more than fifty papers published in it (see Tuza and Zhu's survey [17].)

In this work, we investigate the graceful game, proposed by Tuza [17], and study winning strategies for Alice and Bob in complete graphs, paths, cycles, complete bipartite graphs, caterpillars, prisms, wheels and hypercubes. This paper is organized as follows: Section 2 presents auxiliary results and definitions. Section 3 presents our main results on the graceful game and, in Section 4, we set our conclusion and perspectives.

2 Basic notation and auxiliary lemmas

Before presenting our main results, some definitions are needed. Let G be a simple graph with vertex set $V(G)$ and edge set $E(G)$. We denote an edge $e \in E(G)$ by uv where u and v are its endpoints. Moreover, we also say that the edge e *connects* u and v and that u and v are *adjacent*. An *element* of G is a vertex or an edge of G.

The *Graceful Game* is defined in the following way: Alice and Bob alternately assign a previously unused label $\phi(v) \in \{0, \ldots, m\}$ to a previously unlabeled vertex v of a given graph G. If both endpoints of an edge $uv \in E(G)$ are labeled, the *label* of uv is defined as $|\phi(u) - \phi(v)|$. A move (label assignment) is said to be *legal* if, after it, all edge labels are distinct. In the Graceful Game, Alice *wins* if the whole graph G is gracefully labeled, and Bob *wins* if he can prevent this.

It is well known that not every graph is graceful [9]. For non-graceful graphs, it is immediate that Bob is the winner and, therefore, the game is completely determined for such graphs. In this work, we investigate classes of graphs for which it is possible to obtain a graceful labeling. In the following, we state two auxiliary lemmas that are used in our proofs.

Lemma 1. *Let G be a simple graph. Alice can only use the label 0 (resp. m) to label a vertex $v \in V(G)$ if v is adjacent to every remaining vertex not yet labeled or v is adjacent to a vertex already labeled by Bob with m (resp. 0).*

Proof. The only way to obtain an edge with label m is by assigning labels 0 and m to two adjacent vertices. Thus, suppose Alice assigns 0 to a vertex $v \in V(G)$, without Bob having yet labeled any vertex with m and there is an unlabeled vertex not adjacent to v in the graph. On Bob's next move, he uses m to label the vertex that is not adjacent to v, making it impossible for Alice to gracefully label the graph. The complementary case is analogous. □

Lemma 2. *Let G be a simple graph. If Bob assigns 0 (resp. m) to a vertex $v \in V(G)$, where v has two non-adjacent vertices or only one adjacent vertex, then Alice is forced to label a vertex adjacent to v with m (resp. 0).* □

3 Results

In this section, we state our main results. A *path graph* P_n is a connected graph on n vertices whose vertices can be arranged in a linear sequence $(v_0, v_1, \ldots, v_{n-1})$ in such a way that two vertices are adjacent if and only if they are consecutive in the linear sequence.

Theorem 3. *Bob has a winning strategy for any P_n, $n \geq 4$. For $n = 3$ the winner is the player who starts the game, and Alice has a winning strategy for $n = 1, 2$.*

Sketch of the proof. Alice always wins on P_1 and P_2 since there is only one way of labeling P_1 and P_2, and they are both graceful. For P_3, if Bob starts, it is sufficient that he labels v_1 with 1. In contrast, if Alice starts, she labels v_1 with 0 or 2. Now, independently of Bob's choice, the graph is graceful. Paths P_4 and P_5 are treated separately and, using Lemmas 1 and 2, it is shown that Bob can exhaust Alice's possibilities of creating the edge label $m-1$. For P_n with $n \geq 6$, let us say Alice starts by labeling a vertex $v_i \in V(P_n)$, $i \in \{0, \ldots, n-1\}$, with label j, $j \in \{0, \ldots, m = n-1\}$. We can establish that if $i \in \{0, 1, 2\}$, then Bob labels v_{n-1} for the second vertex and, if $i \geq 3$, he chooses to label v_0. Without loss of generality, we consider $i \geq 3$. As agreed, Bob now labels v_0 and his chosen label must be 0. This forces Alice to assign m to v_1 for the third vertex and then, Bob assigns $m-1$ to v_2 making it impossible for this labeling to be graceful. However, there exists a problem that comes with this strategy: if Alice uses some label, Bob cannot use it again so, if $j = m-1$ the strategy is impracticable. In this case, instead of labeling v_2 with $m-1$, Bob labels it with 2, guaranteeing his victory. The cases where $i \in \{0, 1, 2\}$ are analogous. □

A graph is *bipartite* if its vertex set can be partitioned into two subsets A and B so that every edge has an endpoint in A and the other in B. A simple bipartite graph is *complete*, denoted $K_{p,q}$, if each vertex of A is joined to each vertex of B, where $p = |A|$ and $q = |B|$.

Theorem 4. *Bob has a winning strategy for all $K_{p,q}$, $p, q \geq 2$. Alice wins the Graceful game in any bipartite graph $K_{1,n-1}$ if she is the first player.* □

A *complete graph* K_n is a graph in which every pair of distinct vertices is connected by a unique edge. Golomb [9] proved that a complete graph K_n is graceful if and only if $n \leq 4$. The cases $n = 1$ and $n = 2$ are trivial and also follow by Theorem 3. Thus, it remains to analyze K_3 and K_4.

Theorem 5. *Alice wins on K_3 and Bob on K_4, no matter who starts.* □

The next class of graphs considered in this work are the cycles. A *cycle graph* C_n, with $n \geq 3$, is a connected simple graph such that all of its vertices can be arranged in a cyclic sequence $(v_0, v_1, \ldots, v_{n-1})$ such that two vertices are adjacent if and only if they are consecutive in the sequence. It is known [15] that C_n is graceful if and only if $n \equiv 0, 3 \pmod 4$. Since C_n is not graceful for any other values of n, it is immediate that Bob is the winner in those cases.

Theorem 6. *Bob has a winning strategy for C_n, $n \geq 4$, and Alice can only win on C_3.* □

A *caterpillar* $cat(k_1, k_2, \ldots, k_s)$ is a tree obtained from a path graph $(v_1, v_2, \ldots, v_s)$, called *spine*, by joining k_i leaves to v_i, for each $i \in \{1, \ldots, s\}$. It is known that all caterpillars are graceful [15]. In this work, we prove the following result for caterpillars.

Theorem 7. *Bob has a winning strategy for all caterpillars.* □

The next class of graphs considered in this work are the wheel graphs. A *wheel* W_n is a graph formed by connecting a single vertex v_0 to all vertices $v_1, v_2, \ldots, v_n$ of a cycle C_n, $n \geq 3$. It is known that all wheels are graceful [6] and, in this work, we prove the following result about wheels.

Theorem 8. *When Bob is the first player, he has a winning strategy for all wheel graphs.* □

The *prism* $P_{r,s}$, $s \geq 2$, is defined as the cartesian product $C_r \square P_s$ of a cycle of length r and a path with s vertices. Many families of prisms are known to be graceful [7, 13].

Theorem 9. *Bob has a winning strategy for all prisms $P_{r,s}$ with $r \geq 8$ and $s \geq 1$.* □

A *n-dimensional hypercube graph* Q_k, or just *hypercube*, is defined recursively in terms of the cartesian product of two graphs as follows: (a) $Q_1 = K_2$; and (b) $Q_k = K_2 \square Q_{k-1}$. It is known that all hypercubes are graceful [14]. In this work, we prove the following result.

Theorem 10. *Bob has a winning strategy for all hypercubes Q_k, $k \geq 2$.* □

4 Concluding Remarks

In this work, we investigate the graceful game for many classic families of graphs and determine that Bob has winning strategies for most of them as, for example, for all paths P_n with $n \geq 4$; all cycles C_n with $n \geq 4$; all hypercubes Q_k with $k \geq 2$; all caterpillars; all complete bipartite graphs $K_{p,q}$, $p, q \geq 2$; and all prisms $P_{r,s}$ with $r \geq 8$ and $s \geq 1$. For the investigated classes, Alice has winning strategies for a few cases such as for paths P_1 and P_2, for $K_{1,n-1}$ when she is the first player, and on K_3 and C_3.

As future work, we intend to completely determine the result for wheels when Alice is the first player and also the cases of prisms that remained open. Some other classes of graphs that are interesting to consider, due to their structure, are powers of cycles and paths.

References

[1] K. Appel and W. Haken. Every planar map is four-colorable, Part I: Discharging. *Illinois Journal of Mathematics*, 21(3):429–490, 1977.

[2] O. Baudon, J. Przybyo, M. Senhaji, E. Sidorowicz, E. Sopena, and M. Woźniak. The neighbour-sum-distinguishing edge-colouring game. *Discrete Mathematics*, 340:1564–1572, 2017.

[3] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning ways for your mathematical plays*. Academic Press, 1982.

[4] E. Boudreau, B. Hartnell, K. Schmeisser, and J. Whiteley. A game based on vertex-magic total labelings. *Australasian Journal of Combinatorics. 29 (2004), 6773.*, 29:67–73, 2004.

[5] M.-L. Chia, H.-N. Hsu, D. Kuo, Sh.-Ch. Liaw, and Z.-T. Xu. The game L(d,1)-labeling problem of graphs. *Discrete Mathematics*, 312:3037–3045, 2012.

[6] R. Frucht. Graceful numbering of wheels and related graphs. *Annals of the New York Academy of Sciences*, 319(1):219–229, 1979.

[7] J. A. Gallian. A dynamic survey of graph labeling. *The Electronic Journal of Combinatorics*, DS6:1–502, 2018.

[8] A. Giambrone and E. L. C. King. Vertex-magic edge labeling games on graphs with cycles. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 78:75–96, 2011.

[9] S. W. Golomb. *Graph Theory and Computing*, chapter How to number a graph, pages 23–37. Academic Press, New York, 1972.

[10] R. L. Graham and N. J. A. Sloane. On additive bases and harmonious graphs. *SIAM Journal of Algebraic Discrete Methods*, 1(4):382–404, 1980.

[11] J. R. Griggs and R. K. Yeh. Labeling graphs with a condition at distance 2. *SIAM Journal of Discrete Mathematics*, 5(4):586–595, 1992.

[12] B. Hartnell and D. Rall. A vertex-magic edge labeling game. *Congressus Numerantium*, 161:163–167, 2003.

[13] D. Jungreis and M. Reid. Labeling grids. *Ars Combinatoria*, 34:167–182, 1992.

[14] A. Kotzig. Decompositions of complete graphs into isomorphic cubes. *Journal of Combinatorial Theory*, B 31:292–2496, 1981.

[15] A. Rosa. On certain valuations of the vertices of a graph. *Theory of Graphs, International Symposium, Rome, July 1966*, pages 349–355, 1967.

[16] Z. Tuza. Graph labeling games. *Electronic Notes in Discrete Mathematics*, 60:61–68, 2017.

[17] Z. Tuza and X. Zhu. *Topics in Chromatic Graph Theory*, volume 156 of *Encyclopedia of Mathematics and Its Applications*, chapter Colouring games, pages 304–326. Cambridge University Press, 2015.

On the palette index of graphs with a small cyclomatic number

Aghasi Ghazaryan[1]

[1]Department of Informatics and Applied Mathematics, Yerevan State University, 0025, Armenia

Abstract

Given a proper edge coloring ϕ of a graph G, we define the palette $S_G(v,\phi)$ of a vertex $v \in V(G)$ as the set of all colors appearing on edges incident with v. The palette index $\check{s}(G)$ of G is the minimum number of distinct palettes occurring in a proper edge coloring of G. In this paper we give an upper bound on the palette index of a graph G, in terms of cyclomatic number $cyc(G)$ of G and maximum degree $\Delta(G)$ of G. We also give an upper bound for the palette index of unicycle and bicycle graphs. Moreover, we show that these upper bounds are sharp.

1 Introduction

All graphs considered in this paper are finite, undirected, and have no loops or multiple edges. Let $V(G)$ and $E(G)$ denote the sets of vertices and edges of a graph G, respectively. The degree of a vertex v in G is denoted by $d(v)$, and the maximum degree of vertices in G by $\Delta(G)$. The terms and concepts that we do not define can be found in [1].

The problem of coloring the vertices or edges of a graph has always attracted researcher's attention. These problems become interesting and challenging when one requires that the coloring satisfies certain conditions. For instance, the vertices or edges of the graph have to be properly colored, that is, adjacent vertices or adjacent edges have to receive distinct colors. The existence of a proper vertex-coloring or edge-coloring in a graph G can be formulated in terms of the chromatic parameters $\chi(G)$ or $\chi'(G)$ that are known as the chromatic number or the chromatic index of G, respectively. Probably, these two parameters are the most popular chromatic parameters, but there are many others. For instance, the circular chromatic index [2], the list chromatic number [3], etc. The chromatic parameters can often be used to know something about the structure of a graph. As an example, the chromatic index tells us if a regular graph is decomposable into perfect matchings.

In this paper we consider a chromatic parameter called the palette index of a simple graph G. As far as we know, this parameter was introduced quite recently in [4] and denoted by $\check{s}(G)$. It can be defined as follows. Let ϕ be a proper edge-coloring of G and let v be a vertex of G. The set of colors assigned by ϕ to the edges incident to v is called the palette of ϕ (with respect to ϕ) and is denoted by $S_G(v,\phi)$. For every proper edge-coloring of G we can consider the set $S_G(\phi)$ of distinct palettes of ϕ, namely, $S_G(\phi) = \{S_G(v,\phi) \mid v \in V(G)\}$. The cardinality of $S_G(\phi)$ is at most $|V(G)|$. The palette index of G is the minimum number of distinct palettes taken over all proper edge-colorings of G, namely, $\check{s}(G) = min\{|S_G(\phi)| \mid \phi \text{ proper edge coloring of } G\}$.

As shown in [4], the palette index of a regular graph is 1 if and only if the graph is Class 1 (see [5] for the definition of graphs of Class 1 and 2 according to Vizings Theorem). Moreover, it is different from 2. As mentioned in [6], the palette index of a d-regular graph of Class 2 satisfies the inequality $3 \leq \check{s}(G) \leq d+1$. Hence, $\check{s}(G) = 3$ if G is a 2-regular graph of Class 2.

In [4] it was studied the palette index of cubic graphs. More specifically,

$$\check{s}(G) = \begin{cases} 1 & \text{if } G \text{ is Class 1} \\ 3 & \text{if } G \text{ is Class 2 and has a perfect matching} \\ 4 & \text{if } G \text{ is Class 2 and has no perfect matching} \end{cases}$$

In [4] the palette index of the complete graph K_n $n \geq 4$ is also given. Namely,

$$\check{s}(K_n) = \begin{cases} 1 & \text{if } n \equiv 0 \pmod 2 \\ 3 & \text{if } n \equiv 3 \pmod 4 \\ 4 & \text{if } n \equiv 1 \pmod 4 \end{cases}$$

The paper [6] investigated 4-regular graphs; they proved that $\check{s}(G) \in \{1, 3, 4\}$ if G is 4-regular and Class 2, and that all these values are in fact attained.

Since the computing of the chromatic index of cubic graph is NP-complete [7], determining the palette index of a given graph is also NP-complete, even for cubic graphs [4]. This in fact means that even determining if a given graph has palette index 1 is an NP-complete problem.

Vizing's edge coloring theorem yields an upper bound on the palette index of a general graph G with maximum degree Δ, namely that $\check{s}(G) \leq 2^{\Delta+1} - 2$. It is not hard to construct graphs which palette index is quite small than $2^{\Delta+1} - 2$. Indeed, it was described in [8] an infinite family of multigraphs, whose palette index grows asymptotically as Δ^2. It is an open question whether there are such examples without multiple edges. Furthermore, they suggested to prove that there is a polynomial $p(\Delta)$ so that for any graph with maximum degree Δ, $\check{s}(G) \leq p(\Delta)$.

There are few results about the palette index of non-regular graphs. In [9] it is completely determined the palette index of the complete bipartite graphs $K_{a,b}$ with $a < 5$.

It was studied in [11] the palette index of bipartite graphs. In particular, they determined the exact value of the palette index of grids.

[10] studies the palette index of trees, proving that $\check{s}(T) \leq \sum_{i=1}^{\Delta} \left[\frac{\Delta}{i}\right]$. The paper also proves the sharpness of this bound constructing T^{Δ} graphs for which palette index is $\sum_{i=1}^{\Delta} \left[\frac{\Delta}{i}\right]$.

In this paper we focus on non-regular graphs. In particular, we give tight upper bound for palette index of unicycle graphs. We give tight upper bound for palette index of bicycle graphs. Moreover, we derive a general upper bound on palette index of graphs in terms of cyclomatic number and maximum degree.

2 Main results

Definition 1. *We will say the unicycle graph G with the cycle C satisfies condition * if*

- *The cycle C has an odd length.*
- *There is a vertex with a degree greater than 2, which belongs to $V(C)$.*
- *There are no 2 neighbor vertices from $V(C)$ with a degree greater than 2.*
- *The maximum degree of G is even.*

Theorem 2. *For any unicycle graph G the following inequality is true:*

$$\check{s}(G) \leq \begin{cases} \sum_{i=1}^{\Delta} \left[\frac{\Delta}{i}\right] + 1 & \text{if } G \text{ satisfies } \textbf{\textit{condition *}} \\ \sum_{i=1}^{\Delta} \left[\frac{\Delta}{i}\right] & \text{otherwise} \end{cases}.$$

Moreover, this upper bound is sharp.

Theorem 3. *For any bicycle graph G the following inequality is true :*

$$\check{s}(G) \leq \sum_{i=1}^{\Delta} \left\lceil \frac{\Delta}{i} \right\rceil + 2$$

and this bound is sharp.

Theorem 4. *For any graph G, we have*

$$\check{s}(G) \leq \sum_{i=1}^{\Delta'} \left\lceil \frac{\Delta'}{i} \right\rceil + 2cyc(G),$$

where $\Delta' = min\{\Delta(H) \mid H \text{ is a spanning tree of } G\}$.

Corollary 5. *For any graph G, we have*

$$\check{s}(G) \leq \sum_{i=1}^{\Delta} \left\lceil \frac{\Delta}{i} \right\rceil + 2cyc(G).$$

References

[1] D.B. West, Introduction to Graph Theory, Prentice-Hall, New Jersey, 2001.

[2] X. Zhu: Circular chromatic number: a survey, Discrete Math., 229(2001), 371-410.

[3] V.G. Vizing, Vertex colorings with given colors, Metody Diskret. Analiz., 29(1976), 3-10 (in Russian).

[4] M. Hornak, R. Kalinowski, M. Meszka, M. Wozniak, Minimum number of palettes in edge colorings, Graphs Combin. 30 (2014), 619-626.

[5] S. Fiorini, R. J. Wilson, Edge-colorings of graphs, Research Notes in Mathematics, 16, Pitman, London, 1977.

[6] S. Bonvicini, G. Mazzuoccolo, Edge-colorings of 4-regular graphs with the minimum number of palettes, Graphs Combin. 32 (2016), 1293-1311.

[7] I. Holyer, The NP-completeness of edge-coloring, SIAM J. COMPUT, 10 (1981), 718-720.

[8] M. Avesani, A. Bonisoli, G. Mazzuoccolo, A family of multigraphs with large palette index, preprint available on Arxiv.

[9] M. Hornak, J. Hudak, On the palette index of complete bipartite graphs, Discussiones Mathematicae Graph Theory 38 (2018), 463476.

[10] A. Bonisoli, S. Bonvicini, G. Mazzuoccolo, On the palette index of a graph: the case of trees, Lecture Notes of Seminario Interdisciplinare di Matematica 14 (2017), 49-55.

[11] C.J. Casselgren, P.A. Petrosyan, Some results on the palette index of graphs, Discrete Mathematics and Theoretical Computer Science, 2019, to appear.

A colorful generalization for the Poison Game

Damián-Emilio Gibaja-Romero[1] and Vanessa Cruz-Molina[2]

[1]Universidad Popular Autónoma del Estado de Puebla
[2]Instituto Tecnológico de Toluca

Abstract

The Poison Game is a two-player combinatorial game, on a finite directed graph, where players, a robber and a cop, sequentially choose a successor of the vertex previously chosen by the other player. In this game, the cop poisons each vertex that he chooses, which means that the robber cannot choose a vertex previously chosen by the cop. If the robber moves first, Duchet and Meyniel (1993) show that he wins the game if and only if the directed graph has a kernel. In this paper, we study a generalization of the 'Poison game' in which n cop-robbers pairs, each of them identified by a color, simultaneously play a poison game on a finite $n-$edge colored directed graph D. We show that D has a kernel by monochromatic paths when all the players who move first win the game, but the converse is not true.

1 Introduction

The 'Cops and Robbers Game' is a combinatorial game in which a cop tries to capture a robber when they sequentially choose vertices along a finite graph; the cop wins the game when he occupies the same vertex where the robber stands (Fromme, 1984). The 'Poison Game' is a variation of the cops and robbers game, introduced by (Duchet and Meyniel, 1993), where the cop and the robber sequentially choose a vertex on a finite directed graph. It is a game of complete knowledge since the graph is common knowledge and each player observes the vertex chosen by the other player. The 'Poison Game' proceeds as follows: the robber chooses first, and the cop chooses second by selecting a successor vertex of the vertex chosen by the robber. In this 'Cops and Robbers' variation, the cop has the ability to poison every vertex that he chooses, which means that the robber cannot choose a vertex previously chosen by the cop. The cop wins the game when he captures the robber, i.e., in a situation where the robber cannot choose a non-poisoned vertex. Duchet and Meyniel (1993) demonstrate that the robber (or the player who moves first) wins the game if and only if the digraph where they move has a kernel, introduced in Von Neumann and Morgenstern (2007) as a solution concept.

This paper presents a generalization for the 'Poison Game,' the Colorful Poison Game,' considering that n cop-robber pairs (R_i, C_i) play a poison game on a directed graph whose set of arrows is partitioned into n disjoint sets, and each set is identified with a color $i \in \{1, 2, \ldots, n\}$. Hence, we say that an arrow is i-colored when it belongs to the set of arrows identified with color i. Also, we consider that each pair (R_i, C_i) is identified by color i. In this generalization, cops-robbers pairs simultaneously play a poison game in the $n-$ colored digraph, with the restriction that they only can move on arrows identified with their color. In other words, a player only can choose a vertex if it is the final vertex of an arrow identified with his color. Cops poison the vertices that they choose with his color, i.e., the robber with the same color cannot choose such vertices in later stages. We assume that robbers are immune to the poison of cops with a different color, and many players of

different colors can stand in the same vertex. The cops win the game if no robber can choose a non-poisoned vertex. We investigate if results from the classical 'Poison Game' hold for this game.

In the classical 'Poison Game,' whenever the digraph has a kernel, choosing a vertex in the kernel represents a winning strategy for the robber since the kernel is a set of independent and absorbent vertices. In other words, there are no arrows between vertices in the kernel, and there is a directed path from all vertices outside the kernel to some vertex in this set Duchet and Meyniel (1993). So, it is necessary to establish what is a "kernel" for n-colored digraphs since we consider that n cop-robber pairs simultaneously interact in a n colored digraph. To generalize the notion of a kernel for colored graphs, we start by introducing the meaning of a monochromatic path. A *monochromatic path* is a path whose arrows are colored by the same color. Hence, the *kernel by monochromatic paths* generalizes the notion of a kernel for colored digraphs Galeana-Sánchez (1996); Sands et al. (1982) since it is a set of vertices that are independent and absorbent by monochromatic paths. We demonstrate that a kernel by monochromatic paths exists if the agents who move first (the robbers) win the game. However, the existence of a kernel by monochromatic paths does not guarantee the victory of the robbers by choosing a vertex in this set.

2 Basic elements

Let $\mathcal{P}$ be the set of players with cardinality $2n$. We consider a partition $\{R, C\}$ of P, where $R = \{R_1, R_2, \ldots, R_n\}$ and $C = \{C_1, C_2, \ldots, C_n\}$. We say that C_i is the opponent of R_i, and viceversa, for all $i = 1, 2, \ldots, n$.

Consider $D = (V(D), F(D))$ a finite digraph, where $V(D)$ is the set of vertices and $F(D)$ is the set of arrows. Form now on, D refers to a digraph without multiple arrows (different arrows with the same extremes) and loops (arrows that start and finish in the same vertex). A generic vertex in $V(D)$ is denoted by x, and a generic arrow is a pair (x, y) in $V(D) \times V(D)$. Since D is a directed graph, we recall that $(x, y) \neq (y, x)$. The set of all successors of a vertex x is $\Delta_D^+(x) = \{y \in V(D) | xy \in F(D)\}$, and the set of all predecessors of x is $\Delta_D^-(x) = \{y \in V(D) | yx \in F(D)\}$.

Let $\mathcal{N} = \{1, 2, \ldots, n\}$ be a set of n different colors; and we assume that each pair (R_i, C_i) is associated to color i. We say that D is an n-edge colored directed graph if there exists a partition $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ of the arrows' set $F(D)$, where each set F_i is identified with color i. We say that an arrow that belongs to F_i is i-colored.

For vertices $x, y \in V(D)$, we say that y is an i-successor of x if and only if $(x, y) \in F_i$. A directed monochromatic path, or simply a monochromatic path, of color i is a succession of vertices $\{x_0, x_1, \ldots, x_k\}$ such that $(x_i, x_{i+1}) \in F_i$ for all $i = 0, 1, \ldots k-1$. We say that a monochromatic path $\{x_0, x_1, \ldots, x_k\}$ has length k. A subset $\mathcal{K} \subseteq V(D)$ is a **kernel by monochromatic paths** (KMP) if and only if i) (Independence property) there is no monochromatic path from x to y, for all $x, y \in \mathcal{K}$; and ii) (Absorbency property) there exists a monochromatic path from z to some element of $\mathcal{K}$, for all $z \in V(D) \setminus \mathcal{K}$.

3 The colorful Poison Game

In our Poison game generalization, we consider that players in $\mathcal{P}$ choose a vertex on a finite n-edge colored directed graph D. The colorful Poison game proceeds as follows.

During the first step, all robbers in R simultaneously choose a vertex of D. In the next step, all cops in C simultaneously choose and poison an i-successor of the vertex chosen by his opponent. We allow many cops and robbers to choose the same vertex, which means that different cops can poison a vertex.

The game continues iteratively. As in the classical 'Poison game,' R_i cannot choose a vertex previously chosen by its opponent C_i, but we assume that R_i is immune to the poison of C_j, for all $j \neq i$. In other words, R_i can choose the same vertex that C_j chooses or poisons, for all $j \neq i$. Each robber R_i chooses a non-poisoned i-successor of the vertex chosen by C_i. Each cop C_i chooses and poisons an i-successor of the vertex chosen by his opponent R_i. Cops can choose any vertex even if it is a poisoned vertex.

A player wins the game when his opponent cannot choose a vertex successor because all successors are poisoned or there are no successors. The colorful poison game "finishes" when no player can choose a successor. However, as in the classical 'Poison Game,' it is possible that players infinitely choose a successor of the vertex previously chosen by his opponent. If this happen for a pair (R_i, C_i), we say that the R_i wins the game.

4 Results

Duchet and Meyniel (1993) demonstrate that a robber wins the game if and only if the game is played on a directed graph with a kernel. In the Colorful Poison game, the existence of a KMP does not guarantee that robbers win the game, even though such concept is the natural generalization of the kernel concept for colored graphs, as the following example illustrates.

Example 1. *Consider $R = \{R_1, R_2\}$ and $C = \{C_1, C_2\}$. We identify (R_1, C_1) with red, and (R_2, C_2) with blue. We consider that the colorful Poison Game is played on the directed graph D, that we show in Figure 1. Note that $K = \{1, 3\}$ is a KMP of the directed graph D.*

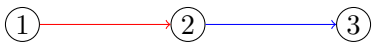


Figure 1: Poison game with 2 pairs of players.

Consider that the Colorful Poison game unfolds as follows. During the first step, R_1 chooses the vertex 1, while R_2 chooses the vertex 3; i.e., both robbers choose a vertex that belongs to K. In the second step, we observe that C_2 cannot choose a successor vertex of 3, which means that R_2 wins. However, vertex 2 is a red-successor of 1, and C_1 can choose it. Since vertex 2 has no red-successors, the game finishes and R_1 losses the game.

Remark 2. *In monochromatic directed graphs, the robber can escape from the cop since any poisoned vertex has a successor in the kernel that the robber can choose. By Example 1, vertices in a KMP do not guarantee winning strategies for all robbers since an **alternating path** can connect its vertices.*[1]

Remark 2 establishes a necessary and sufficient condition to guarantee that vertices in a KMP induce a winning strategy for all robbers.

Theorem 3. *Let D be an $n-$edge colored directed graph D and let $\mathcal{K}$ be a KMP of D. All robbers win the game by choosing any vertex in $\mathcal{K}$ if and only if $\Delta_D(x)^+ = \varnothing$ for all $x \in \mathcal{K}$.*

Proof. Consider that robbers win the game by choosing any $y \in \mathcal{K}$. We proceed by contradiction, i.e., we assume that $\Delta_D^+(x) \neq \varnothing$ for some $x \in \mathcal{K}$. This means that x has an i-successor $y \notin \mathcal{K}$ that is absorbed by some $x' \in \mathcal{K} \setminus \{x\}$ through a monochromatic path $\{y = x_0, x_1, \ldots, x_k = x'\}$ of color $j \neq i$. Note that R_i looses the game by choosing x because $\{x, y, x_1\}$ generates an alternating

[1] An alternating path is a path where any two adjacent edges have a different color.

path, which contradicts the fact that all robbers win the game by choosing a vertex in $\mathcal{K}$. Finally, choosing a vertex $x \in \mathcal{K}$ induces a winning strategy for each robber in R since no cop can choose a successor of x. Therefore, all robbers win the game when they choose a vertex in $\mathcal{K}$. □

Although a KMP does not guarantee a winning strategy for robbers, the colored directed graph has a KMP when all robbers win the game. Note that a set of vertices that generate a winning strategy for all robbers is not necessarily a kernel by monochromatic paths. For example, in a graph that is the union of monochromatic directed cycles of length four induce a winning strategy for all robbers. Thus, the proof of our main result basis on building a KMP from the set of vertices that guarantees a winning strategy for any robber.

Theorem 4. *Consider a colorful Poison game played on an n-edge colored directed graph D. If robbers win the game, then D has a kernel by monochromatic paths.*

Proof. If all robbers win the game, there exists a non-empty subset $\overline{V}$ of $V(D)$ such that cops do not capture robbers when they choose vertices in $\overline{V}$.

We know that vertices with no successors guarantee a winning strategy for all robbers; even more, such vertices belong to a KMP, if it does exist (see Theorem 3). Let $\overline{V}^0 = \{x \in \overline{V} | \Delta_D^+(x) = \varnothing\}$ the set of all vertices in $\overline{V}$ that have no successors, which can be empty or not. In this document, we discuss the case where $\overline{V}^0 \neq \varnothing$.

Now, consider $\overline{V}^{0+} = \left\{x \in \overline{V} \setminus \overline{V}^0 | \text{there exists a monochromatic path from } x \text{ to } y \in \overline{V}^0\right\}$. In words, $\overline{V}^{0+}$ is the set of all vertices absorbed by any vertex in $\overline{V}^{0+}$.

Claim 5. *Let $\overline{V}^1 = \overline{V} \setminus \left(\overline{V}^0 \cup \overline{V}^{0+}\right)$. The set $\overline{V}^0 \cup \overline{V}^1$ is KMP of D.*

By definition, there are no paths from any vertex $x \in \overline{V}^1$ to some vertex $y \in \overline{V}^0$. Moreover, vertices in $\overline{V}^0$ have no successors. Therefore, $\overline{V}^0 \cup \overline{V}^1$ satisfies the property of independence.

Also, by construction, there exists a monochromatic path from any vertex in $\overline{V}$ to some vertex in $\overline{V}^0 \cup \overline{V}^1$. We need to prove the existence of a monochromatic path from any vertex in $V(D) \setminus \overline{V}$ to some vertex in $\overline{V}^0 \cup \overline{V}^1$. Consider $x_1 \in V(D) \setminus \overline{V}$, i.e., x does not induce a winning strategy for some $R_i \in R$. Then, C_i chooses and poisons an i successor x_2 of x_1, and the game finishes.

Note that vertex x_2 can belong to $\overline{V}^0$ or not. In the last case, we can generate an infinite path which contradicts the fact that D is finite. Therefore, $\overline{V}^0 \cup \overline{V}^1$ is a KMP.

□

References

Duchet, P. and Meyniel, H. (1993). Kernels in directed graphs: a poison game. *Discrete mathematics*, 115(1-3):273–276.

Fromme, M. A.-M. (1984). A game of cops and robbers. *Discrete Appl. Math*, 8:1–12.

Galeana-Sánchez, H. (1996). On monochromatic paths and monochromatic cycles in edge coloured tournaments. *Discrete Mathematics*, 156(1-3):103–112.

Sands, B., Sauer, N., and Woodrow, R. (1982). On monochromatic paths in edge-coloured digraphs. *Journal of Combinatorial Theory, Series B*, 33(3):271–275.

Von Neumann, J. and Morgenstern, O. (2007). *Theory of games and economic behavior (commemorative edition)*. Princeton university press.

Enumeration of Minimal Connected Dominating Sets in chordal bipartite graphs*

Benjamin Gras[1] and Mathieu Liedloff[1]

[1]Université d'Orléans, INSA Centre Val de Loire, LIFO EA 4022, FR-45067 Orléans, France

Abstract

The enumeration of subsets of vertices satisfying a given property in a graph can have two objectives, finding an exact algorithm to solve an NP-hard problem and also getting a combinatorial bound on the number of such objects in a graph. The Connected Dominating Set problem has been extensively studied, in the enumeration setting we ask to output all inclusion-wise minimal connected dominating sets. In this paper, we consider the class of chordal bipartite graphs and prove that any such graph has at most $\mathcal{O}^*(1.7990^n)$ different minimal connected dominating sets, and that they can be enumerated in such time.

1 Introduction and definitions

The design of exact exponential algorithms in order to solve NP-hard problems gave rise to a deep study of enumeration algorithms [3]. It has also the perk of giving combinatorial bounds on the number of objects. Minimal Connected Dominating Sets Enumeration have recently seen some improvements in general graphs [5], and also in various graph classes [4], but there is still a gap between lower bounds and upper bounds, most notably in the general case where the best known algorithm runs in time $\mathcal{O}^*\left(2^{(1-\varepsilon)n}\right)$ with $\varepsilon = 10^{-50}$ while the best published lower bound is $\Omega(1.4422^n)$. Bipartite graphs have no better algorithm than the one for general graphs, and it might be interesting to improve the enumeration for this class of graphs. With this goal in mind, trying to improve the result on a class with more structure could be a good starting point, so we describe an algorithm for the class of chordal bipartite graphs, which are not chordal in general, so the best previously known algorithm was the one for general graphs [5], and not the algorithm for chordal graphs running in time $\mathcal{O}^*(1.7159^n)$ [4].

We start with some definitions and notations. Let $G = (X, Y, E)$ be a bipartite graph and $W \subseteq V$ a subset of the vertices of G. We will denote by $G[W]$ the subgraph of G *induced* by W. Given vertex $v \in V$, we will denote by $N(x)$ the open neighbourhood of x and $N[x]$ its closed neighbourhood. Moreover, we denote by $N_W(x)$ the open neighbourhood of x in $G[W]$. By extension, we will use $N(W)$ as the union of the neighbourhoods of vertices of W, $N(W) = \bigcup_{w \in W} N(w)$, we will also use $N[W]$ defined analogously. For two vertices $u, v \in W$, we will denote by $u \sim_W v$ the fact that u, v are *connected* in $G[W]$. An edge $xy \in E$ is called *bisimplicial* if $G[N(x) \cup N(y)]$ is a complete bipartite graph. A graph $G = (V, E)$ is a *chordal bipartite* graph if it is bipartite and weakly-chordal. This means that every induced cycle of a chordal bipartite graph is exactly of length 4. The name chordal bipartite could be misleading as it is not the intersection between chordal and bipartite graphs, as such a class would contain only forests. A set of vertices

*This work is supported by the French National Research Agency, ANR project GraphEn (ANR-15-CE40-0009).

$S \subseteq V$ is a *dominating set* if $N[S] = V$. It is a *connected set* if $G[S]$ is a connected graph. A set S of elements is called *minimal* or *inclusion-wise minimal* for a property Π if S satisfies Π and no strict subset of S satisfies Π.

Proposition 1 (See [2] for example)**.** *A bipartite graph $G = (V, E)$ is chordal bipartite if and only if it has a perfect edge-without-vertex elimination ordering, meaning there is an order on the edges $e_1, \cdots, e_{|E|}$ such that $\forall i \in \{1, \cdots, |E|\}, e_i$ is bisimplicial for the graph $G = (V, E - \{e_1, \cdots, e_{i-1}\})$.*

In particular, any chordal bipartite graph has a bisimplicial edge if it has an edge, and such a bisimplicial edge can be found in polynomial time [1]. The point of this paper is to describe an algorithm that, given a chordal bipartite graph, enumerates every set of vertices that is minimal for the property of being both a connected set and a dominating set.

2 Algorithm

2.1 The algorithm

The algorithm consists in enumerating every subset of the smaller of both independent sets of the chordal bipartite graph. And then complete it with vertices of the other independent set using a branching algorithm based on the property that a chordal bipartite graph has a bisimplicial edge.

Algorithm 1: MCDS

Input : $G = (X, Y, E)$ assuming without loss of generality that $|Y| \leq |X|$.
Output: Every Minimal Connected Dominating Set of G.
Enumerate every subset Y' of Y, discard each Y' such that $N(Y') \neq X$. For each remaining Y', launch the subroutine MCDSsub$(G, X \cup Y, Y')$.

Algorithm 2: MCDSsub

Input : G, O, S where $G = (X, Y, E)$ is a chordal bipartite graph, $O, S \in X \cup Y$.
Output: Some Minimal Connected Dominating Sets S' of G such that $S' - S \subseteq O \cap X$.
If $G[O]$ is edgeless, return S iff it is a minimal connected dominating set of G.
Now, $G[O]$ is chordal bipartite and it has an edge, so it has a bisimplicial edge $xy \in X \times Y$.
The algorithm is then a set of cases, and we always apply the first case possible.

1. If $d_{G[O]}(y) \geq 2$ then return MCDSsub$(G, O - N_O(y), S \cup \{x\})$ and MCDSsub$(G, O - \{x\}, S)$
2. If $y \notin N(S)$ then return MCDSsub$(G, O - \{x\}, S \cup \{x\})$
3. If $y \notin S$ then return MCDSsub$(G, O - \{y\}, S)$
4. If $\exists y' \in S \cap N_O(x) - \{y\}$ such that $y \sim_S y'$, then return MCDSsub$(G, O - \{y\}, S)$
5. If $S \cap N_O(x) - \{y\} = \emptyset$ then return MCDSsub$(G, O - \{y\}, S)$
6. Otherwise return MCDSsub$(G, O - \{x\}, S \cup \{x\})$

2.2 Correctness proof

We are going to prove that if the input is G, O, S where $G = (X, Y, E)$ is a chordal bipartite graph, O and S are two sets of vertices satisfying the property:

$$\forall x \in X \cap O, \left(\exists y \in N(x) \cap S - O \Rightarrow \exists y' \in N(x) \cap O \cap S \text{ s.t. } y \sim_S y'\right) \\ \vee \left(N_O(x) \cap S = \emptyset \wedge \forall y, y' \in N(x) \cap S - O, y \sim_S y'\right), \qquad (+)$$

then the output of MCDSsub is every Minimal Connected Dominating Set S' of G such that $S' - S \subseteq O \cap X$ and $S' - S$ is an inclusion-wise minimal set for the property:

$$O \cap Y \subseteq N(S') \wedge \forall y, y' \in O \cap Y \cap S, y \sim_{S'} y'. \qquad (*)$$

First, we give some intuition behind the previous statement. The set O corresponds to the set of vertices that we still consider, so in X the vertices for which we still have to decide if we put it in the connected dominating set or not, in Y the vertices that are not in O are vertices we voluntarily took out of it, after making sure they can not change the decision. To this extent, the property $(+)$ describes the fact that for each vertex x in $X \cap O$, either each of its neighbours that was taken out of O was useless because a different neighbour of x is already in the same connected component induced by S and still in O, or x can not connect two vertices of different connected components induced by S. We ask for this property for the input to ensure that if a vertex x in O can be taken in S' to connect two components of $G[S]$, then x sees both those components through vertices in O, so that we consider them.

The property $(*)$ only corresponds to the fact that we only consider O, and so that the extension of S into S' is a set that is minimal for both domination and connectivity only for the vertices in O, without considering the vertices not in O.

To show this statement, it is quite straightforward to see that property $(+)$ is maintained in all the cases, since it could only be violated in the cases 3,4,5, and the definition of the property $(+)$ fits the different cases. So we have to prove that the output is the one claimed.

If $G[O]$ has no edges, then the output is correct, since the only one possible is $S' = S$, otherwise it can not be minimal for $(*)$.

To prove the case 1, we observe that any set of vertices containing x and an other vertex x' of $N_O(y)$ can not be minimal for property $(*)$ since $N_O(x) \subseteq N_O(x')$ and property $(+)$ which ensures that considering vertices of $O \cap Y$ is sufficient for the connectivity of vertices in O. For the case 2, x is the only possible vertex in O so that $y \in N(S')$, so there is no choice but to put x in S'. In the case 3, y is in $N(S)$ but not in S, so this vertex satisfies every condition of property $(*)$ already. The case 4 is also straightforward: every extension in $O \cap X$ of S connecting y' to every other vertex of $Y \cap O \cap S$ also connects y to every vertex of $Y \cap O \cap S$. For the case 5, y is already in $N(S)$ and the only neighbour of y in O, x, can not connect y to any vertex of $O \cap Y \cap S$, so the fact that y belongs to O does not change the set of sets of vertices satisfying the property $(*)$.

Lemma 2. *For every call to MCDSsub , for every $y \in G$, if $d_{G[O]}(y) \geq 2$ then $y \in O$.*

Proof. Every vertex is in O in the first call, then a vertex $y \in Y$ can be taken out of O only in cases 3, 4, 5, in which $d_{G[O]}(y) = 1$, and we never add any vertex to O. □

In order to prove case 6, we prove the following Lemma:

Lemma 3. *If yx is an edge such that $N_O(y) = \{x\}$, $y \in S$ and for every $y' \in S \cap N_O(x) - \{y\}$, y' and y do not belong to the same connected component in $G[S]$ and $S \cap N_O(x) - \{y\} \neq \emptyset$ then every extension of S into S' using vertices of $O \cap X$ such that S' is a minimal connected dominating set of G contains the vertex x.*

Proof. Let yx be an edge such that $N_O(y) = x$, $y \in S$ and for every $y' \in S \cap N_O(x) - \{y\}$, y' and y do not belong to the same connected component in $G[S]$ and $S \cap N_O(x) - \{y\} \neq \emptyset$.

Let's assume that there exists an extension S' of S into a minimal connected dominating set such that $S' - S \subseteq X \cap O - \{x\}$.

Let $y' \in S \cap N_O(x) - \{y\}$. We know there exists a path in S' from y' to y, let's call $y_1 x_2 y_3 x_4 \cdots y_p$ an induced path in S' such that $y_1 = y$ and $y_p = y'$. Since y and y' are not in the same connected

component in $G[S]$ and $N_O(y) = x$, $p \geq 5$. So $xy_1x_2y_3 \cdots y_p$ is a cycle of length at least 6 in G, so it is not an induced cycle. Since G is chordal bipartite and there is no edge between any x_i and y_j unless $|i - j| = 1$, x is adjacent to every y_i. $S' \cap Y = S \cap Y$, so for every i, $y_i \in S \cap N(x)$.

Let z be the last vertex on this path such that u_k and y are in the same connected component in $G[S]$, $z \in Y$ because $S' \cap Y = S \cap Y$, thus $z = y_k$ for some k. We have then $x_{k+1} \notin S$, so $x_{k+1} \in O$, so $|N_O(y_k)| \geq 2$. This gives us both the facts that $y_k \in O$ by Lemma 2 and $y_k \neq y$. We can conclude with $y_k \in S \cap N_O(x) - \{y\}$, which contradicts the fact that for every $y' \in S \cap N_O(x) - \{y\}$, y' and y do not belong to the same connected component in $G[S]$. □

One can then easily see that MCDS returns every Minimal Connected Dominating Set of a chordal bipartite graph G.

2.3 Running-time Analysis

MCDSsub is a branching algorithm, we will use the measure $|X \cap O|$ to analyse its complexity. There is only one case where there is branching, where in one branch the measure decreases by 1 and in the other one it decreases by at least 2. All other cases are reduction rules where the measure is non-increasing, and the cases where the measure is constant are used a polynomially bounded number of time as the size of O decreases by at least 1 each time and is at most n. So MCDSsub runs in time $\mathcal{O}^*\left(1.6181^{|X|}\right)$.

MCDS launches MCDSsub at most $2^{|Y|}$ times. So it runs in time $\mathcal{O}^*\left(2^{|Y|}1.6181^{|X|}\right)$. Since $|Y| \leq \frac{n}{2}$, the running time of MCDS is $\mathcal{O}^*\left(1.7990^n\right)$ which is then also a combinatorial upper bound on the number of minimal connected dominating sets in a chordal bipartite graph.

3 Conclusion

The best known lower bound on the time complexity of such algorithm is $\Omega\left(1.4422^n\right)$ [6], an interesting question would be to tighten the gap between this lower bound and our upper bound.

References

[1] M. Bomhoff and B. Manthey. Bisimplicial edges in bipartite graphs. *Discrete Applied Mathematics*, 161(12):1699–1706, 2013.

[2] A. Brandstädt, J. P. Spinrad, et al. *Graph classes: a survey*, volume 3. Siam, 1999.

[3] F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.

[4] P. A. Golovach, P. Heggernes, and D. Kratsch. Enumerating minimal connected dominating sets in graphs of bounded chordality. *Theoretical Computer Science*, 630:63 – 75, 2016.

[5] D. Lokshtanov, M. Pilipczuk, and S. Saurabh. Below all subsets for minimal connected dominating set. *CoRR*, abs/1611.00840, 2016.

[6] M. Y. Sayadi. Private Communications, 2018.

Dispersing obnoxious facilities on a graph

Alexander Grigoriev[1], Tim A. Hartmann[2], Stefan Lendl[3], and Gerhard J. Woeginger[2]

[1]Department of Quantitative Economics, Maastricht University, The Netherlands
[2]Department of Computer Science, RWTH Aachen, Germany
[3]Institute of Discrete Mathematics, TU Graz, Austria

Abstract

We study a continuous facility location problem on a graph where all edges have unit length and where the facilities may also be positioned in the interior of the edges. The goal is to position as many facilities as possible subject to the condition that any two facilities have at least distance δ from each other.

We investigate the complexity of this problem in terms of the rational parameter δ. The problem is polynomially solvable, if the numerator of δ is 1 or 2, while all other cases turn out to be NP-hard.

1 Introduction

A large part of the facility location literature deals with *desirable* facilities that people like to have nearby, such as service centers, police departments, fire stations, and warehouses. However, there also are facilities that are *undesirable* and *obnoxious*, such as nuclear reactors, garbage dumps, chemical plants, military installations, and high security penal institutions. A standard goal in location theory is to spread out such obnoxious facilities and to avoid their accumulation and concentration in a small region; see for instance Erkut & Neuman [5] and Cappanera [2] for comprehensive surveys on this topic.

In this paper, we investigate the location of obnoxious facilities on a graph. Formally, let $G = (V, E)$ be an undirected connected graph, where every edge is rectifiable and has unit length. Let $P(G)$ denote the continuum set of points on all the edges and vertices. For two points $p, q \in P(G)$, we denote by $d(p, q)$ the length of a shortest path connecting p and q in the graph. A subset $S \subset P(G)$ is said to be *δ-dispersed* for some positive real number δ, if any two points $p, q \in S$ with $p \neq q$ are at distance $d(p, q) \geq \delta$ from each other. Our goal is to compute for a given graph $G = (V, E)$ and a given positive real number δ a maximum cardinality subset $S \subset P(G)$ that is δ-dispersed. Such a set S is called an *optimal* δ-dispersed set, and $|S|$ is called the *δ-dispersion number* δ-Disp(G).

Our results.

We provide a complete picture of the complexity of computing the δ-dispersion number for connected graphs $G = (V, E)$ and positive rational numbers δ.

- If $\delta = 1/b$ for some integer b, then the δ-dispersion number of G can be written down without really looking at the structure of the graph: If G is a tree then δ-Disp$(G) = b|E| + 1$, and if G is not a tree then δ-Disp$(G) = b|E|$.

- If $\delta = 2/b$ for some integer b, then δ-Disp(G) can be computed in polynomial time. The algorithm uses the Edmonds-Gallai decomposition of G and reformulates the problem as a submodular optimization problem.
- If $\delta = a/b$ for integers a and b with $a \geq 3$ and $\gcd(a, b) = 1$, then the computation of δ-Disp(G) is an NP-hard problem.

2 Notation and technical preliminaries

All graphs in this paper are undirected and connected, and all edges have unit length. Throughout the paper we use the word *vertex* in the graph-theoretic sense, and we use the word *point* to denote the elements of the geometric structure $P(G)$. For a graph $G = (V, E)$ and a subset $V' \subseteq V$, we denote by $G[V']$ the subgraph induced by V'. For an integer $c \geq 1$, the *c-subdivision* of G is the graph that results from G by subdividing every edge in E by $c - 1$ new vertices into c new edges.

For an edge $e = \{u, v\}$ and a real number λ with $0 \leq \lambda \leq 1$, we denote by $p(u, v, \lambda)$ the point on e that has distance λ from vertex u. Note that $p(u, v, 0) = u$ and $p(u, v, 1) = v$, and note that point $p(u, v, \lambda)$ coincides with point $p(v, u, 1 - \lambda)$; hence we will sometimes assume without loss of generality that $\lambda \leq 1/2$.

3 Results

Lemma 1. *Let G be a graph, let $c \geq 1$ be an integer, and let G' be the c-subdivision of G. Then for every $\delta > 0$, the δ-dispersed sets in G are in one-to-one correspondence with the $(c \cdot \delta)$-dispersed sets in G'. In particular, δ-Disp(G) = $(c \cdot \delta)$-Disp(G').*

Lemma 1 has many useful consequences, as for instance the following:

Lemma 2. *Let $\delta > 0$ and let $c \geq 1$ be an integer.*

- *If the problem of computing the δ-dispersion number is NP-hard, then also the problem of computing the $(c \cdot \delta)$-dispersion number is NP-hard.*
- *If the problem of computing the $(c \cdot \delta)$-dispersion number is polynomially solvable, then also the problem of computing the δ-dispersion number is polynomially solvable.*

For integers ℓ and k, the rational number ℓ/k is called *k-simple*. A set $S \subseteq P(G)$ is k-simple, if for every point $p(u, v, \lambda)$ in S the number λ is k-simple.

Lemma 3. *Let $\delta = a/b$ with integers a and b, and let $G = (V, E)$ be a graph. Then there exists an optimal δ-dispersed set S^* that is $2b$-simple.*

From Lemma 3, we obtain an NP-certificate for computing the δ-dispersion number of a given graph.

By a reduction from Independent Set in Cubic Graphs, we show that computing the δ-dispersion number is NP-hard for $\delta = a/b$, $\gcd(a, b) = 1$ and $a \geq 3$. Our arguments do not work for the cases with $a = 1$ and $a = 2$, as our gadgets and our arguments break down at various places.

Theorem 4. *Let a and b be positive integers with $\gcd(a, b) = 1$ and odd $a \geq 3$. Then it is NP-complete to compute the (a/b)-dispersion number of a graph G.*

The polynomial time result for $\delta = 2$

The following theorem goes back to Edmonds [4] and Gallai [6, 7]; see also Lovász & Plummer [9].

Theorem 5. *(Edmonds-Gallai structure theorem) Let $G = (V, E)$ be a graph. The following decomposition of V into three sets X, Y, Z can be computed in polynomial time.*

$$\begin{aligned} X &= \{v \in V \mid \text{there exists a maximum matching that misses } v\} \\ Y &= \{v \in V \mid v \notin X \text{ and } v \text{ is adjacent to some vertex in } X\} \\ Z &= V - (X \cup Y) \end{aligned}$$

The Edmonds-Gallai decomposition has the following properties:

- *Set X is the union of the odd-sized components of $G - Y$; every such odd-sized component is factor-critical. Set Z is the union of the even-sized components of $G - Y$.*
- *Every maximum matching in G induces a perfect matching on every (even-sized) component of Z and a near-perfect matching on every (odd-sized) component of X. Furthermore, the matching matches the vertices in Y to vertices that belong to $|Y|$ different components of X.* □

We further subdivide the set X in the Edmonds-Gallai decomposition into two parts: Set X_1 contains the vertices of X that belong to components of size 1, and set $X_{\geq 3}$ contains the vertices that belong to (odd-sized) components of size at least 3. The *vicinity* $\mathrm{vic}(v)$ of a vertex $v \in V$ consists of vertex v itself and of the midpoints of all edges incident to v.

Lemma 6. *There exists an optimal 2-dispersed set S^* in canonical form (with underlying edge set E^*) that additionally satisfies the following three properties.*

P1. In every component of $X_{\geq 3}$, the set E^* induces a near-perfect matching.

P2. For every vertex $y \in Y$, the set $\mathrm{vic}(y) \cap S^*$ is either empty or consists of the midpoint of some edge between X and Y.

P3. In every component of Z, the set E^* induces a perfect matching.

Theorem 7. *The 2-dispersion number of a graph G can be computed in polynomial time.* □

The polynomially solvable cases

Theorem 7 and Lemma 2 together imply that for every rational number $\delta = a/b$ with numerator $a \leq 2$, the δ-dispersion number of a graph can be computed in polynomial time. We now present some results that provide additional structural insights into these cases. The cases where the numerator is $a = 1$ are structurally trivial, and the value of the corresponding δ-dispersion number can be written down with the sole knowledge of $|V|$ and $|E|$.

Lemma 8. *Let $\delta = 1/b$ for some integer b, and let $G = (V, E)$ be a connected graph.*

- *If G is a tree then δ-$\mathrm{Disp}(G) = b|E| + 1$.*
- *If G is not a tree then δ-$\mathrm{Disp}(G) = b|E|$.*

The following lemma derives an explicit (and very simple) connection between the 2-dispersion number and the $(2/b)$-dispersion number (with odd denominator b) of a graph. The lemma also implies directly that for every odd b, the computation of $(2/b)$-dispersion numbers is polynomial time equivalent to the computation of 2-dispersion numbers.

Lemma 9. *Let $G = (V, E)$ be a graph, let $z \geq 1$ be an integer, and let $\delta = 2/(2z + 1)$. Then the dispersion numbers satisfy* $\delta\text{-Disp}(G) = 2\text{-Disp}(G) + z|E|$.

References

[1] S. Abravaya and M. Segal. Maximizing the number of obnoxious facilities to locate within a bounded region. *Computers and Operations Research*, 37:163–171, 2010.

[2] P. Cappanera. A survey on obnoxious facility location problems. Technical report, Dipartimento di Informatica, Universitá di Pisa, Italy, 2010.

[3] R.L. Church and R.S. Garfinkel. Locating an obnoxious facility on a network. *Transportation Science*, 12:107–118, 1978.

[4] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[5] E. Erkut and S. Neuman. Analytical models for locating undesirable facilities. *European Journal of Operational Research*, 40:275–291, 1989.

[6] T. Gallai. Kritische Graphen II. *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei*, 8:373–395, 1963.

[7] T. Gallai. Maximale Systeme unabhängiger Kanten. *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei*, 9:401–413, 1964.

[8] A.J. Goldman and P.M. Dearing. Concepts of optimal location for partially noxious facilities. *ORSA Bulletin*, 23:B–31, 1975.

[9] L. Lovász and M.D. Plummer. *Matching Theory.* Annals of Discrete Mathematics 29, North-Holland, Amsterdam, 1986.

[10] N. Megiddo and A. Tamir. New results on the complexity of p-center problems. *SIAM Journal on Computing*, 12:751–758, 1983.

[11] M. Segal. Placing an obnoxious facility in geometric networks. *Nordic Journal of Computing*, 10:224–237, 2003.

[12] A. Tamir. Obnoxious facility location on graphs. *SIAM Journal on Discrete Mathematics*, 4:550–567, 1991.

Compatible spanning circuits in edge-colored Fan-type graphs

Zhiwei Guo[1,2,3], Hajo Broersma[2], Binlong Li[1,3], and Shenggui Zhang[1,3]

[1]Department of Applied Mathematics, Northwestern Polytechnical University, Xi'an, 710072, P.R. China
[2]Faculty of EEMCS, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
[3]Xi'an-Budapest Joint Research Center for Combinatorics, Xi'an, Shaanxi 710129, P.R. China

Abstract

A spanning circuit in a graph G is defined as a closed trail visiting each vertex of G. A compatible spanning circuit in an edge-colored graph refers to a spanning circuit in which each pair of edges traversed consecutively along the spanning circuit have distinct colors. As two extreme cases, the existence of compatible Hamilton cycles and compatible Eulerian circuits in edge-colored graphs has been studied extensively. In recent results the existence of compatible spanning circuits visiting each vertex v at least $\lfloor (d(v)-1)/2 \rfloor$ times in edge-colored graphs satisfying Ore-type conditions has been proved. In this presentation, we show several results on the existence of compatible spanning circuits visiting each vertex at least a specified number of times in edge-colored Fan-type graphs. We will also present a sufficient condition for the existence of such compatible spanning circuits in edge-colored $2(k+1)$-edge-connected graphs, as well as some sufficient conditions for the asymptotical existence of compatible spanning circuits in edge-colored random graphs.

1 Introduction

In this presentation we consider only finite undirected graphs without loops or multiple edges. For terminology and notations not defined here, we refer the reader to Bondy and Murty [4].

Let G be a graph. We use $V(G)$ and $E(G)$ to denote the set of vertices and edges of G, respectively. For a vertex v of G, denote by $E(v)$ the set of edges of G incident to v, and $d(v) = |E(v)|$ the *degree* of v in G. In particular, $\delta(G)$ denotes the *minimum degree* of G. For two vertices u, v of G, a (u,v)*-path* of G refers to a path of G connecting u and v, and the *distance* between u and v in G, denoted by $dist(u,v)$, is defined as the length of a shortest (u,v)-path of G.

A *spanning circuit* in a graph G is defined as a closed trail that visits each vertex of G. A *Hamilton cycle* of G can be regarded as a spanning circuit that visits each vertex of G exactly once; and an *Eulerian circuit* of G can be regarded as a spanning circuit that traverses each edge of G. It is not difficult to see that a spanning circuit can be considered as a relaxation between a Hamilton cycle and an Eulerian circuit. A graph is said to be *Hamiltonian* if it contains a Hamilton cycle, and *Eulerian* if it admits an Eulerian circuit. It is well-known that determining whether a graph is Hamiltonian is NP-complete, and many sufficient conditions for the existence of Hamilton cycles have been found. In particular, Fan [6] proved that if a 2-connected graph G on $n \geq 3$ vertices satisfies $\max\{d(u), d(v)\} \geq n/2$ for every pair of vertices u, v of G with $dist(u,v) = 2$, then G is Hamiltonian. A graph G on n vertices is called a *Fan-type graph* if $\max\{d(u), d(v)\} \geq n/2 + c$ for every pair of vertices u, v of G with $dist(u,v) = 2$, where c is some fixed constant. There is also a well-known characterization of Eulerian graphs, i.e. a connected graph G is Eulerian if and only if the degree of each vertex of G is even (see [4]). Fleury further obtained a polynomial-time algorithm to find an Eulerian circuit in an arbitrary Eulerian graph (see [4]). A *spanning Eulerian*

subgraph of a graph G refers to an Eulerian spanning subgraph of G. Clearly, each spanning circuit of a graph G corresponds to a spanning Eulerian subgraph of G. A graph is said to be *supereulerian* if it contains a spanning Eulerian subgraph. Pulleyblank [13] proved that determining whether a graph is supereulerian is NP-complete (even for planar graphs).

An *edge-coloring* of a graph G is defined as a mapping $c : E(G) \to \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. An *edge-colored graph* refers to a graph with a fixed edge-coloring. A *compatible spanning circuit* in an edge-colored graph is defined as a spanning circuit in which each pair of edges traversed consecutively along the spanning circuit have distinct colors. An edge-colored graph is said to be *properly colored* if each pair of adjacent edges of the graph have distinct colors. Thus, a compatible Hamilton cycle is also properly colored. However, a compatible spanning circuit is not necessarily properly colored. Compatible spanning circuits are of interest in graph theory applications, for example, in genetic and molecular biology [12], in the design of printed circuit and wiring boards [14], and in channel assignment in wireless networks [1].

Let G be an edge-colored graph. Denote by $C(G)$ the set of colors appearing on the edges of G, and $d^i(v)$ the cardinality of the set $\{e \in E(v) : c(e) = i\}$ for a vertex $v \in V(G)$ and a color $i \in C(G)$. For a vertex v of G, we define the *maximum monochromatic degree* of v as $\Delta^{mon}(v) = \max\{d^i(v) : i \in C(G)\}$.

As two extreme cases of compatible spanning circuits, the existence of compatible Hamilton cycles and compatible Eulerian circuits in edge-colored graphs has been studied extensively. For more details on the existence of compatible Hamilton cycles, we refer the reader to [10] and some related references cited in [10]. For more details on the existence of compatible Eulerian circuits, we refer the reader to [2, 9].

Recently, Guo et al. [8] considered the existence of more general compatible spanning circuits in edge-colored graphs for the first time, and proved some sufficient conditions for the existence of compatible spanning circuits visiting each vertex v at least $\lfloor (d(v) - 1)/2 \rfloor$ times in edge-colored graphs satisfying Ore-type conditions. The following problem was also presented in [8].

Problem 1 (Guo, Li, Li and Zhang [8])**.** *Let G be an edge-colored 2-connected graph on n vertices satisfying Fan's condition (see [6]), i.e., $\max\{d(u), d(v)\} \geq n/2$ for every pair of vertices u, v of G with $dist(u, v) = 2$. Under what conditions does G contain a compatible spanning circuit visiting each vertex v at least $\lfloor (d(v) - 1)/2 \rfloor$ times?*

In this presentation, we first demonstrate a sufficient condition for the existence of compatible spanning circuits visiting each vertex v at least $\lfloor (d(v)-3)/2 \rfloor$ times in edge-colored Fan-type graphs, as follows.

Theorem 1. *Let G be an edge-colored 2-connected graph on n vertices with $\delta(G) \geq 3$ such that $\max\{d(u), d(v)\} \geq n/2 + 1$ for every pair of vertices u, v of G with $dist(u, v) = 2$. If $\Delta^{mon}(v) \leq (d(v) - 3)/2$ for each vertex v of G with $d(v) \geq 5$, and $\Delta^{mon}(v) = 1$ otherwise, then G contains a compatible spanning circuit visiting each vertex v at least $\lfloor (d(v) - 3)/2 \rfloor$ times.*

We further prove a sufficient condition for the existence of compatible spanning circuits visiting each vertex v at least $\lfloor (d(v) - 1)/2 \rfloor$ times in edge-colored Fan-type graphs, as follows.

Theorem 2. *Let G be an edge-colored 4-connected graph on n vertices such that $\max\{d(u), d(v)\} \geq n/2 + 2$ for every pair of vertices u, v of G with $dist(u, v) = 2$. If $\Delta^{mon}(v) \leq (d(v) - 1)/2$ for each vertex v of G, then G contains a compatible spanning circuit visiting each vertex v at least $\lfloor (d(v) - 1)/2 \rfloor$ times.*

Using the proof technique of Theorem 2, we prove a sufficient condition for the existence of compatible spanning circuits visiting each vertex v at least specified number of times in edge-colored $2(k+1)$-edge-connected graphs, as well as some sufficient conditions for the asymptotical existence of compatible spanning circuits visiting each vertex v at least $\lfloor (d(v)-1)/2 \rfloor$ times in some edge-colored random graphs, as follows.

Theorem 3. *Let G be an edge-colored $2(k+1)$-edge-connected graph. If for each vertex v of G, $\Delta^{mon}(v) \leq r/2$, where $r = \lfloor (k-1)(d(v)-1)/k \rfloor$, then G contains a compatible spanning circuit visiting each vertex v at least $\lfloor r/2 \rfloor$ times.*

A *random graph process* $\widetilde{\mathcal{G}}(n) = (G_0, G_1, \ldots, G_m, \ldots)$ on the vertex set $V_n = \{v_1, v_2, \ldots, v_n\}$, introduced by Bollobás and Frieze [3], is defined as a Markov process (a random process in which the future is independent of the past and is only dependent on the present) in which G_m is a graph with $V(G_m) = V_n$ and $|E(G_m)| = m$. The initial graph G_0 is an empty graph. For $m \geq 1$, the graph G_m is obtained from G_{m-1} by choosing an edge $e \in \binom{V_n}{2} \setminus E(G_{m-1})$ uniformly at random (u.a.r. for short), where $\binom{V_n}{2}$ denotes the set of all pairs of vertices of V_n, and putting $E(G_m) = E(G_{m-1}) \cup \{e\}$. We use $G \sim \widetilde{\mathcal{G}}(n)$ to denote a random graph constructed by the random graph process $\widetilde{\mathcal{G}}(n)$. In fact, the random graph $G_m \sim \widetilde{\mathcal{G}}(n)$ is distributed exactly as the graph $G_{n,m}$ (a random graph chosen uniformly from the set of all graphs with the vertex set $V_n = \{v_1, v_2, \ldots, v_n\}$ and exactly m edges) constructed by the original random graph process introduced by Erdős and Rényi [5].

We say that a property $\mathcal{P}$ of a graph G on n vertices holds *asymptotically almost surely* (a.a.s. for short) if the probability that $\mathcal{P}$ holds tends to 1 as $n \to \infty$.

Theorem 4. *Let G be an edge-colored random graph constructed by the random graph process $\widetilde{\mathcal{G}}(n)$ with $\delta(G) \geq 4$. If $\Delta^{mon}(v) \leq (d(v)-1)/2$ for each vertex v of G, then G contains a.a.s. a compatible spanning circuit visiting each vertex v at least $\lfloor (d(v)-1)/2 \rfloor$ times.*

Let d be a positive integer. A *random d-dimensional geometric graph process* $\widetilde{\mathcal{G}}^d(n,r) = (G^d(n,r))_{0 \leq r < \infty}$ on the vertex set $V_n = \{v_1, v_2, \ldots, v_n\}$, generalized from Gilbert's definition [7], is defined as a continuous process as r ranges from 0 to ∞, in which $G^d(n,r)$ is a random d-dimensional geometric graph defined by placing u.a.r. all vertices of V_n in the d-dimensional unit square $[0,1]^d$, and joining two vertices v_i and v_j whenever $\|v_i - v_j\|_p \leq r$ for a given real number $0 \leq r < \infty$, where $\|\cdot\|_p$ denotes the standard ℓ_p norm for some fixed $1 < p \leq \infty$ (see [11]). We use $G \sim \widetilde{\mathcal{G}}^d(n,r)$ to denote a random d-dimensional geometric graph constructed by the random d-dimensional geometric graph process $\widetilde{\mathcal{G}}^d(n,r)$. For more details on random geometric graph processes and related applications, we refer the reader to [15].

Theorem 5. *Let d be a positive integer, and G be an edge-colored 4-connected random d-dimensional geometric graph constructed by the random d-dimensional geometric graph process $\widetilde{\mathcal{G}}^d(n,r)$. If $\Delta^{mon}(v) \leq (d(v)-1)/2$ for each vertex v of G, then G contains a.a.s. a compatible spanning circuit visiting each vertex v at least $\lfloor (d(v)-1)/2 \rfloor$ times.*

References

[1] S.K. Ahuja, Algorithms for routing and channel assignment in wireless infrastructure networks, Ph.D. thesis, *Univ. Arizona*, 2010.

[2] A. Benkouar, Y. Manoussakis, V.Th. Paschos and R. Saad, Hamiltonian problems in edge-colored complete graphs and Eulerian cycles in edge-colored graphs: Some complexity results, *RAIRO Oper. Res.* **30** (1996) 417–438.

[3] B. Bollobás and A. Frieze, On matchings and Hamiltonian cycles in random graphs, In: Random Graphs'83 (M. Karoński and A. Ruciński, Eds.), North-Holland, Amsterdam, (1985) 23–46.

[4] J.A. Bondy and U.S.R. Murty, Graph Theory, Springer, New York, 2008.

[5] P. Erdős and A. Rényi, On the evolution of random graphs, *Publ. Math. Inst. Hung. Acad. Sci.* **5** (1960) 17–61.

[6] G.H. Fan, New sufficient conditions for cycles in graphs, *J. Combin. Theory Ser. B* **37** (1984) 221–227.

[7] E.N. Gilbert. Random plane networks. *J. Soc. Indust. Appl. Math.* **9** (1961) 533–543.

[8] Z. Guo, B. Li, X. Li and S. Zhang, Compatible spanning circuits in edge-colored graphs, submitted.

[9] A. Kotzig, Moves without forbidden transitions in a graph, Mat. Časopis Sloven, *Akad. Vied* **18** (1968) 76–80.

[10] A. Lo, Properly coloured Hamiltonian cycles in edge-coloured complete graphs, *Combinatorica* **36** (2016) 471–492.

[11] T. Müller, X. Pérez-Giménez and N. Wormald. Disjoint Hamilton cycles in the random geometric graph, *J. Graph Theory* **68** (2011) 299–322.

[12] P.A. Pevzner, Computational molecular biology: an algorithmic approach, MIT Press, Cambridge, MA, 2000.

[13] W.R. Pulleyblank, A note on graphs spanned by Eulerian graphs, *J. Graph Theory* **3** (1979) 309–310.

[14] I.L. Tseng, H.W. Chen and C.I. Lee, Obstacle-aware longest path routing with parallel MILP solvers, *in: Proc WCECS-ICCS* **2** (2010) 827–831.

[15] M. Walters. Random geometric graphs. In surveys in combinatorics 2011, *London Math. Soc. Lecture Note Ser.* **392** (2011) 365–401.

Clustered Feasibility by Breaking

Nili Guttmann-Beck[1] and Stern Michal[1,2]

[1]Academic College of Tel-Aviv Yaffo, Yaffo, Israel
[2]Caesarea Rothchild Institute, university of Haifa, Haifa, Israel

1 Introduction

Let $G = (V, E)$ be a complete undirected graph with a vertex set $V = \{v_1, \ldots, v_n\}$ and the complete edge set E. Let $H =< G, \mathcal{S} >$ be a hypergraph, where $\mathcal{S}$ is a set of clusters $S_1, \ldots, S_m$, $S_i \subseteq V$ for $i \in \{1, \ldots, m\}$, such that the clusters in $\mathcal{S}$ are not necessarily disjoint. The Clustered Spanning Tree by Trees problem, denoted by *CSTT*, is to decide whether there exists a spanning tree of G, such that each cluster induces a subtree. The Clustered Spanning Tree by Paths problem, denoted by *CSTP*, is to decide whether there exists a spanning tree of G, such that each cluster induces a path.

In the *CSTT* problem, verifying whether a hypergraph has a feasible solution tree can be performed in two manners. According to ([1], [2], [4] and summarized in [3]), a hypergraph $H =< G, \mathcal{S} >$ has a feasible solution tree if and only if it satisfies the Helly property and its intersection graph is chordal. A second approach is to use Algorithm *ES*, presented and proved in [5], which finds a maximum spanning tree in a weighted graph which represents the hypergraph. In the *CSTP* problem [6] introduced a polynomial time algorithm which constructs a tree where each cluster spans a path, if one exists.

We use the intersection graph of H to see how an instance for either one of the problems may be divided into smaller instances, when the intersection graph contains a cut-node, a cut-edge or a separating-path, whose removal from the intersection graph breaks the connectivity of the graph. We prove how the feasibility question of every connected component, created after the removal of the cut-node, cut-edge or separating-path, may be used to decide whether the original hypergraph has a feasible solution. This approach may be of great significance regarding the complexity of the decision problems.

For the case when H do no have a feasible solution, we introduce the idea of a feasible removal list and a feasible insertion list. A feasible removal (insertion) list contains a list of vertices and clusters such that removing (inserting) those vertices from (into) the appropriate clusters create a hypergraph with a feasible solution. The structure of the intersection graph may be further used to construct feasible removal (insertion) list based on the corresponding feasible removal (insertion) lists created for each one of the sub-problems corresponding to the different connected components, created after removing a cut-node, cut-edge or separating-path from the intersection graph.

Throughout this paper, we assume that the intersection graph of H is connected. Otherwise, a feasible solution tree of H can be constructed by properly adding edges between the feasible solutions of each connected component, if they exist. When no feasible solution tree exists for this case, the union of feasible removal (insertion) lists of the different connected components creates a feasible removal (inserted) list for the given hypergraph.

A possible motivation for the *CSTT* and *CSTP* problems is presented in [7], from the area of communication networks. Given a complete graph where each vertex represents a customer, each

edge represents a link between two customers, and there is a collection of not necessarily disjoint clusters of vertices where each cluster represents a group of customers. The problem is to construct a communication tree network in such a way that each cluster of vertices from the given collection induces a subtree or a path in the solution tree. In this way, the network has the *group broadcast* property and the *group fault tolerance* property.

2 Induced Graphs

2.1 Definitions

Definition 1. *Given a graph $G = (V, E)$ and a set of not necessarily disjoint clusters $\{S_{i_1}, \ldots, S_{i_p}\}$. The* **intersection graph of $\{S_{i_1}, \ldots, S_{i_p}\}$**, *denoted by $G_{int}(\{S_{i_1}, \ldots, S_{i_p}\})$, is defined to be a graph whose set of nodes is $\{s_{i_1}, \ldots, s_{i_p}\}$, where s_{i_j} corresponds to S_{i_j}, and an edge (s_{i_j}, s_{i_k}) exists if $S_{i_j} \cap S_{i_k} \neq \phi$.*

Definition 2. *Let $H =< G, \mathcal{S} >$ be a hypergraph and let $\mathcal{S}' \subset \mathcal{S}$ be a set of clusters. We define the induced hypergraph $H[\mathcal{S}']$ to be the hypergraph which is the complete graph on the vertex set $V(\mathcal{S}') = \bigcup_{S_i \in \mathcal{S}'} S_i$, and its clusters set is $\mathcal{S}'$.*

Claim 3. *Let $H =< G, \mathcal{S} >$ be a hypergraph, if $G_{int}(\mathcal{S})$ is the intersection graph of H, then the induced graph $G_{int}(\mathcal{S})[\bigcup_{S_i \in \mathcal{S}'} S_i]$ is the intersection graph of $H[\mathcal{S}']$ and therefore can be denoted as $G_{int}(\mathcal{S}')$.*

Definition 4. *Let $H =< G, \mathcal{S} >$ be a hypergraph. RL is a* **removal list of H** *if RL is a list of pairs: $RL = \{(v_1, S_{i_1}), \ldots, (v_k, S_{i_k})\}$ with $v_j \in S_{i_j}$, such that the removal of every vertex v_j from cluster S_{i_j} creates a new instance of the hypergraph. If the new hypergraph has a feasible solution tree (for $CSTT$ or $CSTP$ problem) we say that RL is a* **feasible removal list of H**.

Definition 5. *Let $H =< G, \mathcal{S} >$ be a hypergraph. IL is an* **insertion list of H** *if IL is a list of pairs: $IL = \{(v_1, S_{i_1}), \ldots, (v_k, S_{i_k})\}$ with $v_j \notin S_{i_j}$, such that adding every vertex v_j to cluster S_{i_j} creates a new instance of the hypergraph. If the new hypergraph has a feasible solution tree we say that IL is a* **feasible insertion list of H**.

Definition 6. *Let $H =< G, \mathcal{S} >$ be a hypergraph, if $L = \{(v_1, S_{i_1}), \ldots, (v_k, S_{i_k})\}$ is a removal (insertion) list and $\mathcal{S}' \subset \mathcal{S}$ a set of clusters, we define the* **induced removal (insertion) list $L[\mathcal{S}']$** *to be $\{(v, S_i) | (v, S_i) \in L, v \in V(\mathcal{S}'), S_i \in \mathcal{S}'\}$.*

2.2 Basic Claims

Lemma 7. *Let $H =< G, \mathcal{S} >$ be a hypergraph, if T is a feasible solution tree for $CSTT$ ($CSTP$) problem and $X = \bigcap_{S_i \in \mathcal{S}'} S_i$ for $\mathcal{S}' \subset \mathcal{S}$, then $T[X]$ is a connected subtree (path).*

Theorem 8. *([5]) Let $H =< G, \mathcal{S} >$ be a hypergraph with a connected intersection graph $G_{int}(\mathcal{S})$ and a feasible solution tree T. If $G_{int}(\mathcal{S}')$ is connected for $\mathcal{S}' \subset \mathcal{S}$, then $T[V(\mathcal{S}')]$ is a feasible solution tree of $H[\mathcal{S}']$.*

Lemma 9. *Let $H =< G, \mathcal{S} >$ be a hypergraph, if L is a feasible removal (insertion) list of H, then $L[\mathcal{S}']$ is a feasible removal (insertion) list of $H[\mathcal{S}']$ for every $\mathcal{S}' \subset \mathcal{S}$.*

3 Breaking Results

Definition 10. *A node $v \in V$ is a* **cut-node** *of a connected graph $G = (V, E)$ if the induced graph of G on $V \backslash \{v\}$ is not connected.*

Remark 11. *Consider a cut-node s^* of the intersection graph $G_{int}(\mathcal{S})$, and let $\mathcal{S}_1, \ldots, \mathcal{S}_k$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S}) \backslash \{s^*\}$. In this case $\mathcal{S}_1, \ldots, \mathcal{S}_k, \{S^*\}$ are pairwise disjoint, $V(\mathcal{S}_1), \ldots, V(\mathcal{S}_k)$ are pairwise disjoint, but $(\bigcup_{i=1}^{k} \mathcal{S}_i) \cup \{S^*\} = \mathcal{S}$ and $(\bigcup_{i=1}^{k} V(\mathcal{S}_i)) \cup S^* = V$.*

Lemma 12. *Given a hypergraph $H =< G, \mathcal{S} >$, whose intersection graph contains a cut-node s^*, and let $\mathcal{S}_1, \ldots, \mathcal{S}_k$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S}) \backslash \{s^*\}$. If every $H[(\mathcal{S}_i \cup \{S^*\}]$, $i \in \{1, \ldots, k\}$, has a feasible solution tree, for the CSTT (CSTP) problem, then H has a feasible solution tree for the CSTT (CSTP) problem.*

Corollary 13. *Given a hypergraph $H =< G, \mathcal{S} >$, whose intersection graph contains a cut-node s^*, and let $\mathcal{S}_1, \ldots, \mathcal{S}_k$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S}) \backslash \{s^*\}$. If L_i is a feasible removal (insertion) list of $H[\mathcal{S}_i \cup \{S^*\}]$, $i \in \{1, \ldots, k\}$, then $\bigcup_{i=1}^{k} L_i$ is a feasible removal (insertion) list of H, for the CSTT (CSTP) problem.*

Definition 14. *An edge (v_1, v_2) is a* **cut-edge** *of a connected graph $G = (V, E)$ if removing the edge (v_1, v_2) from G disconnects G into two connected components.*

Remark 15. *Consider a cut-edge (s_1, s_2) of the intersection graph $G_{int}(\mathcal{S})$, and let $\mathcal{S}_1, \mathcal{S}_2$ be the clusters sets which correspond to the two connected components of $G_{int}(\mathcal{S}) \backslash \{(s_1, s_2)\}$, with $s_1 \in \mathcal{S}_1$ and $s_2 \in \mathcal{S}_2$. In this case, $\mathcal{S}_1, \mathcal{S}_2$ are pairwise disjoint, but $\mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}$, $V(\mathcal{S}_1) \cap V(\mathcal{S}_2) = S_1 \cap S_2$ and $V(\mathcal{S}_1) \cup V(\mathcal{S}_2) = V$.*

Lemma 16. *Given a hypergraph $H =< G, \mathcal{S} >$, whose intersection graph contains a cut-edge (s_1, s_2), and let $\mathcal{S}_1, \ldots, \mathcal{S}_k$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S}) \backslash \{(s_1, s_2)\}$. If $H[\mathcal{S}_1]$ and $H[\mathcal{S}_2]$ have feasible solution trees for the CSTT (CSTP) problem, then H has a feasible solution tree for the CSTT (CSTP) problem.*

Corollary 17. *Given a hypergraph $H =< G, \mathcal{S} >$, whose intersection graph contains a cut-edge (s_1, s_2), and let $\mathcal{S}_1, \mathcal{S}_2$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S}) \backslash \{(s_1, s_2)\}$. If L_1 is a feasible removal (insertion) list of $H[(\mathcal{S}_1 \backslash \{S_1\}) \cup \{S_1 \cap V(\mathcal{S}_1)\}]$ and L_2 is a feasible removal (insertion) list of $H[(\mathcal{S}_2 \backslash \{S_2\}) \cup \{S_2 \cap V(\mathcal{S}_2)\}]$, then $L_1 \cup L_2$ is a feasible removal (insertion) list of H, for the CSTT (CSTP) problem.*

Definition 18. *A path $P = (v_1 - \cdots - v_k)$ is a* **separating-path** *of a connected graph $G = (V, E)$ if removing vertices $v_1, \ldots, v_k$ from G disconnects G into two connected components. However, G remains connected if we do not remove one of the vertices v_1 or v_k.*

Remark 19. *Consider a separating-path $P = (s_1 - \cdots - s_k)$ of the intersection graph $G_{int}(\mathcal{S})$, and let $\mathcal{S}_a, \mathcal{S}_b$ be the clusters sets which correspond to the two connected components of $G_{int}(\mathcal{S}) \backslash \{s_1, \ldots, s_k\}$. Since $G_{int}(\mathcal{S})$ remains connected if we do not remove one of the vertices s_1 or s_k, there is at least one edge connecting s_1 to $\mathcal{S}_a$, at least one edge connecting s_1 to $\mathcal{S}_b$ and similarly for s_k. Note that $\mathcal{S}_a, \mathcal{S}_b$ are disjoint and $\mathcal{S}_a \cup \mathcal{S}_b \cup \{S_1, \ldots, S_k\} = \mathcal{S}$.*

Lemma 20. *Given a hypergraph $H =< G, \mathcal{S} >$, whose intersection graph contains a separating-path $P = (s_1 - \cdots - s_k)$ and let $\mathcal{S}_a, \mathcal{S}_b$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S}) \backslash \{s_1, \ldots, s_k\}$. If $H[\mathcal{S}_a \cup \{S_1, \ldots, S_k\}]$ and $H[\mathcal{S}_b \cup \{S_1, \ldots, S_k\}]$ have feasible solution trees for the CSTT problem, then $IL = \{(v^*, S_i) | 3 \leq i \leq k\}$ for a vertex $v^* \in S_1 \cap S_2$, is a feasible insertion list of H for the CSTT problem.*

Corollary 21. *Given a hypergraph $H =< G, \mathcal{S} >$, whose intersection graph contains a separating-path $P = (v_1 - \cdots - v_k)$ and let $\mathcal{S}_a, \mathcal{S}_b$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S})\backslash\{s_1, \ldots, s_k\}$. If IL_a is a feasible insertion list of $H[\mathcal{S}_a \cup \{S_1, \ldots, S_k\}]$ and IL_b is a feasible insertion list of $H[\mathcal{S}_b \cup \{S_1, \ldots, S_k\}]$ for the CSTT problem, then $IL_a \cup IL_b \cup IL_p$, where $IL_p = \{(v^*, S_i) | 3 \leq i \leq k\}$ for a vertex $v^* \in S_1 \cap S_2$, is a feasible insertion list of H for the CSTT problem.*

Lemma 22. *Given a hypergraph $H =< G, \mathcal{S} >$, whose intersection graph contains a separating-path with $k = 2$ ($P = (s_1 - s_2)$) and let $\mathcal{S}_a, \mathcal{S}_b$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S})\backslash\{s_1, s_2\}$. If L_a is a feasible removal (insertion) list of $H[\mathcal{S}_a \cup \{S_1, S_2\}]$ and L_b is a feasible removal list of $H[\mathcal{S}_b \cup \{S_1, S_2\}]$ for the CSTT problem, then $L_a \cup L_b$ is a feasible removal (insertion) list of H for the CSTT problem.*

Remark 23. *Consider a hypergraph $H =< G, \mathcal{S} >$, whose intersection graph contains a separating-path with $k = 2$ and let $\mathcal{S}_a, \mathcal{S}_b$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S})\backslash\{s_1, s_2\}$. There exists such hypergraph where $H[\mathcal{S}_a \cup \{S_1, S_2\}]$ and $H[\mathcal{S}_b \cup \{S_1, S_2\}]$ have feasible solution trees for the CSTP problem, but H has no feasible solution tree for the CSTP problem.*

Lemma 24. *Given a hypergraph $H =< G, \mathcal{S} >$, whose intersection graph contains a separating-path with $k = 2$ and let $\mathcal{S}_a, \mathcal{S}_b$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S})\backslash\{s_1, s_2\}$. If $H[\mathcal{S}_a \cup \{S_1, S_2\}]$ and $H[\mathcal{S}_b \cup \{S_1, S_2\}]$ have feasible solution trees for the CSTP problem, and at least two of the next sets: $V(\mathcal{S}_a) \cap (S_1 \cap S_2)$ or $V(\mathcal{S}_b) \cap (S_1 \cap S_2)$ or $(S_1 \cap S_2)\backslash(V(\mathcal{S}_a) \cup V(\mathcal{S}_b))$ are empty, then H has a feasible solution tree for the CSTP problem.*

Corollary 25. *Given a hypergraph $H =< G, \mathcal{S} >$, whose intersection graph contains a separating-path with $k = 2$ and let $\mathcal{S}_a, \mathcal{S}_b$ be the clusters sets which correspond to the connected components of $G_{int}(\mathcal{S})\backslash\{s_1, s_2\}$. If RL_a is a feasible removal list of $H[\mathcal{S}_a \cup \{S_1, S_2\}]$ and RL_b is a feasible removal list of $H[\mathcal{S}_b \cup \{S_1, S_2\})]$ for the CSTP problem, then $RL_a \cup RL_b \cup RL$, where $RL = \{(v, S_i) | v \in S_1 \cap S_2, S_i \in (\mathcal{S}_a \cup \mathcal{S}_b)\}$, is a feasible removal list of H for the CSTP problem.*

References

[1] P. Duchet. Propriété de helly et problémes de représentation. *Colloqu. Internat. CNRS, Problemes Conbinatoires et Theorie du Graphs, Orsay, France*, 260:117–118, 1976.

[2] C. Flament. Hypergraphes arborés. *Discrete Math.*, 21(3):223–227, 1978.

[3] T. A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory.* SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.

[4] P. J. Slater. A characterization of soft hypergraphs. *Canad. Math. Bull.*, 21(3):335–337, 1978.

[5] Z. Sorek. Nodes insertion for the intersecting clusters problem. Technical report, The Academic College of Tel Aviv Yaffo, 2015.

[6] R. Swaminathan and D. K. Wagner. On the consecutive-retrieval problem. *SIAM J. Comput.*, 23(2):398–414, 1994.

[7] A. S. Tanenbaum and D. J. Wetherall. *Computer Networks.* Prentice Hall, 5 edition, 2011.

Improved Dynamic Kernels for Hitting-Set

Zacharias Heinrich[1] and Rüdiger Reischuk[1]

[1]Institut für Theoretische Informatik, Universität zu Lübeck, 23562 Lübeck, Germany,
{zacharias.heinrich,reischuk}@tcs.uni-luebeck.de

Abstract

A dynamic kernelisation for d-hitting set with parameter k named *lazy sunflower* is presented that achieves an update time $O(d^2\ d!\ k^d)$ for insertion and $O(d^3\ d!\ k^d)$ for deletions which significantly improves the best bounds known so far.

1 Introduction

A graph algorithm $\mathcal{A}$ takes a graph G as instance and computes an output $\pi(G)$ for a given task like determining the (number of) connected components, coloring vertices with ℓ colors, or finding a vertex cover of size at most k. In a dynamic environment graphs may change as new edge connections between vertices show up or existing ones are lost. Then we have to handle a sequence $G_0, G_1, \ldots$ of graphs over a fixed set of vertices V and would like to generate the outputs $\pi(G_i)$ efficiently [HK01, HdLT01].

Instead of computing the result each time from scratch one is interested in a data structure to represent the versions G_i in such a way that $\pi(G_i)$ can be obtained much faster. Now the update time for a single edge insertion or deletion becomes important, too, to measure the overall speedup obtained. If $T_\pi(m)$ denotes the time to solve the problem for a static graph G with m edges then starting with the edge-free graph and performing m insertions to generate G a dynamic graph algorithm $\mathcal{A}$ could be applied to compute $\pi(G)$. Thus a single update of $\mathcal{A}$ cannot be cheaper than $T_\pi(m)/m$ and the goal is to come to this bound as close as possible. Several basic graph problems in $\mathcal{P}$ have been investigated in this respect and appropriate algorithms have been presented.

For $\mathcal{NP}$-hard graph problems like vertex cover fixed parameter algorithms have been designed to obtain fast solutions even for large graphs if the additional parameter is bounded (for more details see [Ca15]). The main technique is kernelisation. It reduces a large graph to a much smaller subgraph that is *equivalent* with respect to the problem considered.

Can these ideas be combined to solve parameterized problems in a dynamic setting more efficiently than computing a new kernel after each update? Few papers have considered this question so far. [IO14] has investigated k-vertex cover and achieved an update time $O(k^2)$ under the constraint that each G_i actually possesses a vertex cover of size k. [AMW17] develops a dynamization of the Buss kernel and achieves an update time $O(k)$ in the worst case and $O(1)$ amortized and does not need the condition on the maximal size of a vertex cover.

***d*-hitting set**, given a hypergraph with edge size bounded by some $d \in \mathbb{N}$, does there exist a set V' of at most k vertices such that each edge has a nonempty intersection with V', is a natural generalization of vertex cover. [AMW17] constructs a dynamic kernel for d-hitting set based on a complicated notion of *good sets* that achieves an update time $(d!)^d\ k^{O(d^2)}$ and kernel size $d!\ (k+1)^d$. This paper exploits the sunflower property [ER60] more directly and constructs a dynamic kernelisation of size $d!\ k^d$ with $O(d^2\ d!\ k^d)$ time for edge insertion and $O(d^3\ d!\ k^d)$ time for edge deletion.

2 Definitions

Vertex cover and hitting set are well known $\mathcal{NP}$-hard graph problems that belong to the complexity class FPT (fixed-parameter tractable).

Definition 1 (FPT). *A parameterized language $L \subseteq \Sigma^* \times \mathbb{N}$ is in* FPT *if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and an algorithm $\mathcal{A}$ that decides membership in L for inputs (I, k) in time bounded by $f(k) \cdot poly(|(I)|)$.*

Vertex cover is probably the most prominent example for a problem in FPT. It can be considered as the special case of d-hitting set for $d = 2$.

Definition 2 (Parameterized d-Hitting Set). *For $d \in \mathbb{N}$ instances of d-*hitting set *are tupels (G, k), where $G = (V, E)$ is a hypergraph with edge size bounded by d and $k \in \mathbb{N}$ is an additional parameter. A hitting set is a subset $V' \subseteq V$ of vertices such that $V' \cap e \neq \emptyset$ for all $e \in E$.*
One has to decide whether G possesses a hitting set of size k.

A standard technique to put a language into FPT is kernelisation,

Definition 3 (Kernelisation). *A kernelisation for a parameterized language L is a mapping, where instances (I, k) are reduced to smaller instances (I', k') called kernel with $|I'| \leq f(k)$ for some computable function f and $k' \leq k$ such that $(I, k) \in L \iff (I', k') \in L$. In addition, this mapping should be efficiently computable with respect to the size of I.*

Kernelisation can be used in particular for graph problems. We want to extend this technique to a dynamic environment.

Definition 4 (Dynamic Graph Algorithm). *A dynamic graph algorithm $\mathcal{A}$ receives as input a sequence $X = G, x_1, x_2, \ldots$, where $G = (V, E)$ is a hypergraph with vertex set V of size n and edge set E, and the x_i are operations on edges, either* `insert`(e) *or* `delete`(e) *for some $e \subseteq V$. The input sequence defines a sequence of hypergraphs $G_0, G_1, G_2, \ldots$ where $G_0 = G$ and G_{i+1} is derived from G_i by applying operation x_{i+1} if the relevant edge e is not present, resp. is present in G_i.*

$\mathcal{A}$ solves a graph problem π dynamically if it maintains a data structure $Y = Y_0, Y_1, Y_2, \ldots$, where Y_i is supposed to relate to G_i such that $\pi(G_i)$ can be determined using Y_i.

Let T_{in} denote the (worst case) time of $\mathcal{A}$ to update Y_i from Y_{i-1} if x_i is an `insert`*, and T_{del} for a* `delete` *assuming that one starts with an edge free graph. Furthermore, T_{out} denotes the time to compute $\pi(G_i)$ from $\pi(G_{i-1})$, Y_{i-1} and Y_i.*

A dynamic kernelisation uses as data structure Y a sequence of kernels that have to be updated according to the edge operations x_i. From a kernel Y_i for G_i a solution $\pi(G_i)$ can then be derived using a fast static FPT-algorithm for the given problem. Since the size of the kernel matters one would like to keep it small.

The notion of a sunflower has shown to be helpful for solving the hitting set problem.

Definition 5 (Sunflower). *A subset $\{e_1, \ldots, e_p\}$ of edges of a hypergraph G is called a sunflower S if there exists a subset $C \subseteq V$ of vertices called the* **core** *of S such that $e_i \cap e_j = C$ for all e_i, e_j with $i \neq j$. The sets e_i are called the* **petals** *of S and their number p is the size of S.*

Note that the core can be empty and thus a sunflower may consist of p pairwise disjoint edges. When looking for a hitting set of size k in a graph containing a sunflower of size $k + 1$ one has to select at least one vertex of the core. Thus one can reduce large sunflowers with nonempty core to its core without destroying any minimal (with respect to inclusion) hitting set of size at most k. If the graph contains a sunflower of size $k + 1$ with empty core then a hitting set of size k cannot exist. Erdös and Rado have proven a bound on the existence of sunflowers.

Lemma 1 ([ER60]). *Let $s(d,p)$ denote the minimal number m of edges such that every hypergraph with m edges where each edge is of size at most d has a sunflower with more than p petals. Then $s(d,p) \leq d!\, p^d + 1$.*

3 Constructing Dynamic Kernels

For ordinary graphs ($d = 2$) a sunflower of size larger than k with nonempty core is a vertex v of degree at least $k+1$ together with its incident edges. Thus for a vertex cover of size k such a vertex has to be part of the cover. In a static situation one fixes v for the vertex cover, removes all incident edges and then can select up to $k-1$ additional vertices to cover the remaining graph. If all large degree vertices are handled this way the remaining graph is either obviously too large to allow a small vertex cover or has a kernel of size bounded by $O(k^2)$.

In a dynamic environment where edges may be added or deleted things become more complicated since removing large degree vertices with their edges may lead to high update times. The nodes of a graph are split into three subsets which are COVER: nodes that have to be selected, KERNEL: the nodes in the kernel, and NOC: nodes that are useless for a cover. A dynamic data structure has to represent these relations and should allow a fast implementation of changes.

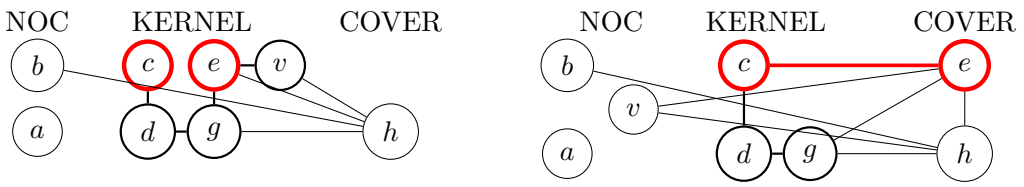


Figure 1: an update of the kernel for vertex cover after inserting edge $\{c, e\}$

In [AMW17] it has been shown that one can achieve a worst case update time $O(k)$ keeping the kernel size bounded by $O(k^2)$. Amortized the update time can even be kept constant. In the same paper a generalisation to hitting sets has been considered.

Theorem 1 ([AMW17]). *There exists a dynamic kernelisation for d-HITTING SET with size bound k that achieves update time $(d!)^d\, k^{O(d^2)}$ and kernel size $d!\,(k+1)^d$.*

This result can be improved as follows:

Theorem 2. *d-HITTING SET with size bound k has a dynamic kernelisation with $T_{in} \leq O(d^2\, d!\, k^d)$ and $T_{del} \leq O(d^3\, d!\, k^d)$ and kernel size $g(d,k) \leq d!\, k^d + 1$.*

We give a short illustration of the main ideas. Given a sequence of hypergraphs G_i that is generated by inserting and deleting edges starting from an edge free graph the data structure stores all current edges until the sunflower bound $g(d,k)$ is reached. Every time the data structure has that many edges we search for a sunflower of size (at least) $k+1$, which can be done in time $O(d^2\ \ d!\, k^d)$. The sunflower is reduced to its core C, that means all petals are removed and a new (virtual) hyperedge C is added. If C is empty we increase a special counter by 1 to keep track of the number of sunflowers with empty core. Over time several new such hyperedges may be generated that have to be carefully distinguished. This complicates the data structure, but we have to skip details in this extended abstract.

Deleting edges that are physically in the data structure can be done directly. If an edge e is part of a sunflower S that has been shrunk to its core and deleting e reduces the size of S

below $k+1$ all remaining petals are added back and the core is removed (see Fig. 2). However, such updates may lead to a recursive movement of cores of sunflowers. Since the size of the cores in such a recursion decreases at least by 1 in each step the depth is bounded by d. This gives an extra factor d in the update time for deletion.

4 Conclusion

An obvious question is whether the update time for hitting set can further be reduced. For vertex cover we have also considered more sophisticated kernels like the one in [CKJ01] of linear size. Our investigations have shown that a good update time for such a kernel is quite unlikely. For the Buss kernel, however, it is open whether constant worst case update time can be achieved.

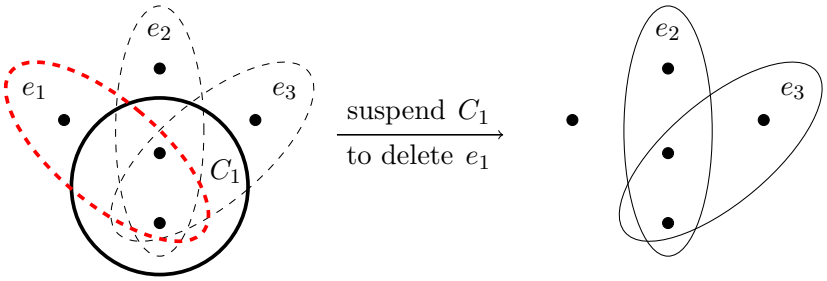


Figure 2: for $k = 2$ deletion of an edge e_1 destroys the 3-petal sunflower generated by e_1, e_2, e_3, which is represented by its core C_1 after a sunflower reduction. To delete the sunflower first the core is removed, then the edge e_1, and finally the two remaining edges are inserted back

References

[AMW17] J. Alman, M. Mnich, and V. Williams. Dynamic parameterized problems and algorithms. In *Proc. 44. ICALP*, pages 41:1–41:16, 2017.

[Ca15] M. Cygan and al. *Parameterized Algorithms*. Springer, 2015.

[CKJ01] J. Chen, I. Kanj, and W. Jia. Vertex cover: Further observations and further improvements. *J. Algorithms*, 41(2):280–301, 2001.

[ER60] P. Erdös and R. Rado. Intersection theorems for systems of sets. *J. London Mathematical Society*, s1-35(1):85–90, 1960.

[HdLT01] J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001.

[HK01] M. Henzinger and V. King. Maintaining minimum spanning forests in dynamic graphs. *SIAM J. Comput.*, 31(2):364–374, 2001.

[IO14] Y. Iwata and K. Oka. Fast dynamic graph algorithms for parameterized problems. In *Proc. 14. SWAT*, pages 241–252, 2014.

Algorithm and Hardness Result for Semipaired Domination in Graphs

Michael A. Henning[1], Arti Pandey[2], and Vikash Tripathi[2]

[1]Department of Pure and Applied Mathematics, University of Johannesburg, Auckland Park, 2006 South Africa

[2]Department of Mathematics, Indian Institute of Technology Ropar, Rupnagar, Punjab, INDIA

mahenning@uj.ac.za, {arti,2017maz0005}@iitrpr.ac.in

Abstract

For a graph G with no isolated vertices, a dominating set D of G is called a semipaired dominating set of G if D can be partitioned into 2-element subsets such that the vertices in each 2-element set are at distance at most two. The minimum cardinality of a semipaired dominating set of G is called the semipaired domination number of G, and is denoted by $\gamma_{pr2}(G)$. The Minimum Semipaired Domination problem is to find a semipaired dominating set of G of cardinality $\gamma_{pr2}(G)$. Given a graph G and a positive integer k, the Semipaired Domination Decision problem is to decide whether G has a semipaired dominating set of cardinality at most k. In this paper, we show that the Semipaired Domination Decision problem is NP-complete even for split graphs, an important subclass of chordal graphs. On the positive side, we propose a linear-time algorithm to solve the Minimum Semipaired Domination problem in trees.

keywords: *Domination, Semipaired domination, Chordal graphs, NP-completeness, Graph algorithms.*

1 Introduction

Let $G = (V, E)$ be a graph. For a vertex $v \in V$, let $N_G(v) = \{u \in V | uv \in E\}$ and $N_G[v] = N_G(v) \cup \{v\}$ denote the *open neighborhood* and the *closed neighborhood* of v, respectively. For two distinct vertices $u, v \in V$, the distance $dist_G(u, v)$ between u and v is the length of a shortest path between u and v. A vertex u *dominates* v if either $u = v$ or u is adjacent to v. A set $D \subseteq V$ is called a *dominating set* of $G = (V, E)$ if each $v \in V$ is dominated by a vertex in D, that is, $|N_G[v] \cap D| \geq 1$ for all $v \in V$. The *domination number* of a graph G, denoted by $\gamma(G)$, is the minimum cardinality of a dominating set of G. For a graph G, the Minimum Domination problem is to find a dominating set of cardinality $\gamma(G)$. The notion of domination and its variations in graphs has been studied a great deal both from a theoretical as well as algorithmic point of view, see [5, 6]; a rough estimate says that it occurs in more than 6000 papers to date. A dominating set D is called a *paired dominating set* if $G[D]$ contains a perfect matching. For a graph G with no isolated vertices, the Minimum paired domination problem is to find a paired dominating set of G of minimum cardinality. The concept of paired domination was introduced by Haynes and Slater in [4].

A relaxed form of paired domination called semipaired domination was introduced by Haynes and Henning in [1] and studied further in [7, 2, 3]. A set S of vertices in a graph G with no isolated

vertices is a *semipaired dominating set*, abbreviated a semi-PD-set, of G if S is a dominating set of G and S can be partitioned into 2-element subsets such that the vertices in each 2-element set are at distance at most 2. In other words, the vertices in the dominating set S can be partitioned into 2-element subsets such that if $\{u, v\}$ is a 2-set, then the distance between u and v is either 1 or 2. We say that u and v are *semipaired*. The *semipaired domination number* of G, denoted by $\gamma_{pr2}(G)$, is the minimum cardinality of a semi-PD-set of G. Since every paired dominating set is a semi-PD-set, and since every semi-PD-set is a dominating set, we have the following observation.

Observation 1.1. *([1]) For every isolate-free graph G, $\gamma(G) \leq \gamma_{pr2}(G) \leq \gamma_{pr}(G)$.*

By Observation 1.1, the semipaired domination number is squeezed between two fundamental domination parameters, namely the domination number and the paired domination number. For a graph G with no isolated vertices, the Minimum Semipaired Domination problem is to find a semipaired dominating set of cardinality $\gamma_{pr2}(G)$, and the Semipaired Domination Decision problem is the decision version of the Minimum Semipaired Domination problem. In this paper, we observe that there are graph classes where paired domination and semipaired domination problem differs in complexity. We also show that the Semipaired Domination Decision problem is NP-complete even for split graphs, a subclass of chordal graphs. On the positive side, we propose a linear-time algorithm to compute a minimum cardinality semipared dominating set of trees.

2 Preliminaries

2.1 Notations

Let $G = (V, E)$ be a graph. A set $S \subseteq V$ is called an *independent set* of G if $uv \notin E$ for all $u, v \in S$. A set $K \subseteq V$ is called a *clique* of G if $uv \in E$ for all $u, v \in K$. A graph G is said to be a *chordal graph* if every cycle in G of length at least four has a *chord*, that is, an edge joining two non-consecutive vertices of the cycle. A chordal graph $G = (V, E)$ is a *split graph* if V can be partitioned into two sets I and C such that C is a clique and I is an independent set. Let n and m denote the number of vertices and number of edges of G, respectively. In this paper, we only consider connected graphs with at least two vertices.

2.2 Complexity difference between paired domination and semipaired domination

In this section, we make an observation on complexity difference between paired domination and semipaired domination. We show that the decision version of the Minimum paired domination problem is NP-complete for GP4 graphs, but the Minimum Semipaired Domination problem is easily solvable for GP4 graphs. The class of GP4 graphs was introduced by Henning and Pandey in [8]. Below we recall the definition of GP4 graphs.

Definition 1 (GP4-graph)**.** *A graph $G = (V, E)$ is called a* GP4-graph *if it can be obtained from a general connected graph $H = (V_H, E_H)$ where $V_H = \{v_1, v_2, \ldots, v_{n_H}\}$, by adding a path of length 3 to every vertex of H. Formally, $V = V_H \cup \{w_i, x_i, y_i, z_i \mid 1 \leq i \leq n_H\}$ and $E = E_H \cup \{v_i w_i, w_i x_i, x_i y_i, y_i z_i \mid 1 \leq i \leq n_H\}$.*

Theorem 2. *If G is a* GP4-graph*, then $\gamma_{pr2}(G) = \frac{2}{5}|V(G)|$.*

Lemma 3. *If G is a GP4-graph constructed from a graph H as in Definition 1, then H has a paired dominating set of cardinality k, $k \leq n_H$ if and only if G has a semi-PD-set of cardinality $2n_H + k$.*

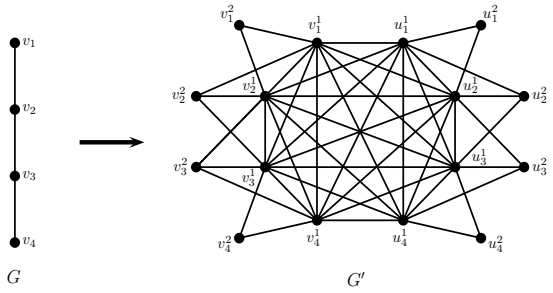


Figure 1: An illustration to the construction of G' from G in the proof of Theorem 5.

Since the decision version of the Minimum Paired Domination problem is known to be NP-complete for general graphs [4], the following theorem follows directly from Lemma 3.

Theorem 4. *The decision version of the* Minimum Paired Domination *problem is NP-complete for GP*4*-graphs.*

3 NP-completeness result for split graphs

Theorem 5. *The* Semipaired Domination Decision *problem is NP-complete for split graphs.*

Proof. Clearly, the Semipaired Domination Decision problem is in NP. To show the hardness, we give a polynomial time reduction from the Domination Decision problem, which is well known NP-complete problem. Given a non-trivial graph $G = (V, E)$, where $V = \{v_i \mid i \in [n]\}$ and $E = \{e_j \mid j \in [m]\}$, we construct a split graph $G' = (V_{G'}, E_{G'})$ as follows:

Let $V_k = \{v_i^k \mid 1 \leq i \leq n\}$ and $U_k = \{u_i^k \mid 1 \leq i \leq n\}$ for $k \in [2]$. Now define $V_{G'} = V_1 \cup V_2 \cup U_1 \cup U_2$, and $E_{G'} = \{uv \mid u, v \in V_1 \cup U_1, u \neq v\} \cup \{v_i^2 v_j^1, u_i^2 u_j^1 \mid 1 \leq i \leq n \text{ and } v_j \in N_G[v_i]\}$. Note that the set $A = V_1 \cup U_1$ is a clique in G' and the set $B = V_2 \cup U_2$ is an independent set in G'. Since $V_{G'} = A \cup B$, the constructed graph G' is a split graph. Fig. 1 illustrates the construction of G' from G.

Now, to complete the proof of the theorem, we only need to prove the following claim.

Claim 6. *G has a dominating set of cardinality k if and only if G' has a semi-PD-set of size cardinality $2k$.*

□

4 Algorithm for trees

In this section, we present a linear-time algorithm to compute a minimum cardinality semipaired dominating set in trees.

Let $T = (V, E)$ be a tree, and $\beta = (v_n, v_{n-1}, \ldots, v_1)$ be the BFS ordering of vertices of T starting at a pendant vertex v_n. Let $\alpha = (v_1, v_2, \ldots, v_n)$ be the reverse ordering of β. In our algorithm, we process the vertices in the order they appear in α. Let $p(v_i)$ denote the parent of vertex v_i. If v_i is the root vertex, we assume $p(v_i) = v_i$.

The idea behind our algorithm is the following. We start with an empty set D, an array L and an array M. Initially $L[v_i] = 0$ and $M[v_i] = 0$ for all $v_i \in V$. We process the vertices one by one

in the order $\alpha = (v_1, v_2, \ldots, v_n)$. During each of the iterations, we update D, L and M suitably. During the iterations, $L[v_i] = 0$ if v_i is not selected in D, $L[v_i] = 1$ if v_i is selected in D but not semipaired, and $L[v_i] = 2$ if v_i is selected in D and semipaired. Also, $M[v_i] = k$ if v_k need to be semipaired with some vertex in $N_T[v_i] \setminus D$. At the end of the algorithm D becomes a minimum cardinality semi-PD-set of the given tree T. At the i^{th} iteration, we process the vertex v_i. While processing v_i, we update D, L and M as follows.
Case 1: $i \neq n, n-1$ and v_i is not dominated by D.
Subcase 1.1: For every $v_r \in N_T[p(v_i)]$, $M[v_r] = 0$.
Update $D = D \cup \{p(v_i)\}$, $L[p(v_i)] = 1$ and $M[p(v_j)] = j$, where $v_j = p(v_i)$.
Subcase 1.2: For some $v_r \in N_T[p(v_i)]$, $M[v_r] \neq 0$.
Let $C = \{v_r \in N_T[p(v_i)] \mid M[w] \neq 0\}$. Let v_k be the least index vertex in C and $m[v_k] = v_s$.
Update $L[p(v_i)] = L[v_s] = 2$, and $D = D \cup \{p(v_i)\}$.
Case 2: $i \in \{n, n-1\}$ and v_i is not dominated by D.
Update $L[v_{n-1}] = L[v_n] = 2$, and $D = D \cup \{v_{n-1}, v_n\}$.
Case 3: v_i is dominated by D and $M[v_i] = 0$.
No Update in D, L and M are made.
Case 4: v_i is dominated by D and $M[v_i] = k \neq 0$ (that is, v_k need to be semipaired with some vertex in $N_T[v_i] \setminus D$).
Subcase 4.1: $L[p(v_i)] = 0$.
Update $L[p(v_i)] = L[v_k] = 2$, $M[v_i] = 0$ and $D = D \cup \{p(v_i)\}$.
Subcase 4.1: $L[p(v_i)] = 1$.
This case will not arrive.
Subcase 4.3: $L[p(v_i)] = 2$.
Update $L[v_i] = L[v_k] = 2$, $M[v_i] = 0$ and $D = D \cup \{v_i\}$.

Theorem 7. *The* Minimum Semipaired Domination *problem is linear-time solvable in trees.*

References

[1] T. W. Haynes, M. A. Henning. Perfect graphs involving semitotal and semipaired domination. J. Comb. Optim., 36 (2018) 416-433.

[2] T. W. Haynes, M. A. Henning. Semipaired domination in graphs. J. Combin. Math. Combin. Comput., 104 (2018) 93-109.

[3] T. W. Haynes and M. A. Henning. Graphs with large semipaired domination number. To appear in Discuss. Math. Graph Theory, doi:10.7151/dmgt.2143.

[4] T. W. Haynes, P. J. Slater. Paired domination in graphs. Networks, 32 (1998) 199-206.

[5] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. Fundamentals of Domination in Graphs, volume 208. Marcel Dekker Inc., New York, 1998.

[6] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. Domination in graphs: Advanced topics, volume 209. Marcel Dekker Inc., New York, 1998.

[7] M. A. Henning, P. Kaemawichanurat. Semipaired Domination in Claw-Free Cubic Graphs. Graphs Combin., 34 (2018) 819-844.

[8] M. A. Henning, A. Pandey. Algorithmic aspects of semitotal domination in graphs. Theor. Comput. Sci., 766 (2019) 46-57.

Algorithmic configuration by learning and optimization

Gabriele Iommazzo[1,2], Claudia D'Ambrosio[1], Antonio Frangioni[2], and Leo Liberti[1]

[1]CNRS LIX, École Polytechnique, Palaiseau, France
[2]Dip. di Informatica, Università di Pisa, Pisa, Italy

Abstract

We propose a methodology, based on machine learning and optimization, for selecting a solver configuration for a given instance. First, we employ a set of solved instances and configurations in order to learn a performance function of the solver. Secondly, we solve a mixed-integer nonlinear program in order to find the best algorithmic configuration based on the performance function.

1 Introduction

In this work we address the configuration of general-purpose Mathematical Programming (MP) solvers. Most solvers have long lists of user-configurable parameters; tweaking them influences how the available algorithmic components work and how they interact with each other, and it can consequently have a significant impact on the quality of the obtained solution and/or on the efficiency of the solution process. Good solvers have effective default parameter configurations, carefully selected to provide "good" performances in most cases. Furthermore, solvers may embed heuristics that try to automatically adapt the parameter configuration to the characteristics of the instance at hand. However, default/automatic parameter configurations may still be highly suboptimal with specific instances, which require a manual search for the best parameter values. The motivation for this work lies in the fact that, due to the large amount of available parameters [10], manual tuning is highly nontrivial and time-consuming. This setting is an instance of the Algorithm Configuration Problem (ACP) [7].

Our approach for addressing the ACP on MP solvers is based on a two-fold process:

(i) in the *Performance Map Learning Phase* (PMLP), supervised Machine Learning (ML) techniques [13] are used to automatically learn a *performance function* which maps some features of the instance being solved, together with a given parameter configuration, into some measure of solver efficiency and effectiveness;

(ii) the formal properties defining the ML methodology underlying the PMLP are translated into MP terms; the resulting formulation, together with constraints encoding the compatibility of the configuration parameter values, is called the *Configuration Space Search Problem* (CSSP), a Mixed-Integer Nonlinear Program (MINLP) which, for a given instance, finds the configuration providing optimal performance w.r.t. the performance function.

The main novelty of our approach lies in the fact that we explicitly model and optimize the CSSP using the mathematical description of the ML technique used to learn the performance function. This is in contrast to most of the existing algorithmic configuration approaches, which instead

employ heuristics such as experimental design methods [1], local searches [8], genetic algorithms [2], evolutionary strategies [5] and other methods [3, 12].

We remark that, differently from many other approaches, we define the performance of the target algorithm as a function of both features and controls. In this way, we account for the fact that the best configuration of a solver may vary among instances belonging to the same class of problems ("per-instance" ACP, see e.g. [3, 9]), whereas some literature works assume instead that the solution to the ACP is invariant over instances pertaining to a given problem (for example, [1, 8, 14]).

The idea of using the components of a ML predictor to define a MP formulation has been already explored and a generalisation of this research can be found, say, in [11]. The proposed approach is in fact suitable for many other applications, besides MP solvers [7], where the outcome of executing some action (e.g., run a target algorithm) depends upon the features of the problem instance and the possible action controls by the user (e.g., the parameters of the target algorithm).

2 The PMLP and the CSSP

Let $\mathcal{A}$ be the target algorithm, and:

- $\mathcal{C}_\mathcal{A}$ be the set of feasible configurations of $\mathcal{A}$; we assume that (a) each configuration c is a vector of binary and/or discrete values representing categorical and numerical parameters and that (b) $\mathcal{C}_\mathcal{A}$ can be described by means of linear constraints;
- Π be the problem to be solved, consisting of an infinite set of instances, and $\Pi' \subset \Pi$ be the (finite) set of instances used as the training set for the PMLP phase;
- F_Π be the set of features used to describe instances, encoded by vectors of continuous or discrete/categorical values (in the latter case they are labelled by reals);
- $p_\mathcal{A} : F_\Pi \times \mathcal{C}_\mathcal{A} \longrightarrow \mathbb{R}$ be the performance function (evaluated using solver performance indicators within a given time limit), which maps a pair (f, c) (instance feature, configuration) to the outcome of an execution of $\mathcal{A}$, typically in terms of the integrality gap reported by the solver, i.e. the relative discrepancy between an optimal objective function bound and the best feasible solution found so far (but other measures are possible).

2.1 Performance Map Learning Phase

In the PMLP phase we use a supervised ML predictor, e.g., Support Vector Regression (SVR), to learn the coefficient vector $\bar{\theta}$ providing the parameters of a prediction model $\bar{p}_\mathcal{A}(\cdot, \cdot, \theta) : F_\Pi \times \mathcal{C}_\mathcal{A} \to \mathbb{R}$ of the performance function $p_\mathcal{A}(\cdot, \cdot)$. The training set for the PMLP is

$$\mathcal{S} = \big\{(f_i, c_i, p_\mathcal{A}(f_i, c_i)) \mid i \in \{1 \dots s\}\big\} \subseteq F_{\Pi'} \times \mathcal{C}_\mathcal{A} \times \mathbb{R}, \tag{1}$$

where $s = |\mathcal{S}|$ and the training set labels $p_\mathcal{A}(f_i, c_i)$ are computed on the training vectors (f_i, c_i). Our training includes a phase for determining the hyperparameters of the ML methodology by nested cross-validation [17]. We also assess the generalization error of the fully-trained predictor configured with the best hyperparameter setting.

2.2 Configuration Space Search Problem

For a given instance f and parameter vector θ, $\mathsf{CSSP}(f,\theta)$ is the problem of finding the configuration with best estimated performance $\bar{p}_{\mathcal{A}}(f,c,\theta)$:

$$\mathsf{CSSP}(f,\theta) \equiv \min_{c\in\mathcal{C}_{\mathcal{A}}} \quad \bar{p}_{\mathcal{A}}(f,c,\theta) \ . \tag{2}$$

The actual implementation of $\mathsf{CSSP}(f,\theta)$ depends on the MP formulation selected to encode $\bar{p}_{\mathcal{A}}$, which may require auxiliary variables and constraints to define the properties of the ML predictor.

If $\bar{p}_{\mathcal{A}}$ yields an accurate estimate of $p_{\mathcal{A}}$, we expect the optimum $\bar{c}$ of $\mathsf{CSSP}(f,\theta)$ to be a good approximation of the true optimal configuration c^* for solving f. However, we remark that not only $\mathsf{CSSP}(f,\theta)$ can be hard to solve, it also needs to be solved quickly (otherwise one might as well solve the instance f directly). Achieving a balance between PMLP accuracy and CSSP cost is one of the challenges of this research.

3 Experimental setup

In this paper we report results where the ML predictor of choice was SVR [16]. Its advantages are: (a) the PMLP for training an SVR can be formulated as a convex Quadratic Program (QP), which can be solved efficiently; (b) even complicated and possibly nonlinear performance functions can be learned by using the "kernel trick" [15], which reduces problematic nonlinear transformations of feature vectors to the much simpler computation of inner products; (c) the solution of the PMLP for SVR provides a closed-form algebraic expression of the performance map $\bar{p}_{\mathcal{A}}$, which allows an easier formulation of $\mathsf{CSSP}(f,\theta)$.

instance	bestFeas$_{CD}$	bestFeas$_{CSSP}$	optVal	bestRelax$_{CD}$	bestRelax$_{CSSP}$
i0001	0	0	1,193E+04	1,211E+04	**1,202E+04**
i0002	0	0	4,567E+03	4,791E+03	6,619E+06
i0003	0	0	1,155E+04	9,250E+06	1,285E+07
i0004	9,593E+03	0	1,117E+04	1,144E+04	**1,140E+04**
i0005	7,091E+03	0	8,960E+03	9,255E+03	9,322E+03
i0006	3,083E+03	3,135E+03	3,491E+03	3,609E+03	**3,597E+03**
i0007	6,921E+03	0	8,955E+03	9,060E+03	**9,028E+03**
i0008	0	0	1,539E+04	1,553E+04	**1,552E+04**
i0009	5,182E+03	0	8,778E+03	8,897E+03	**8,894E+03**
i0010	2,520E+03	0	7,721E+03	7,790E+03	1,209E+07
i0011	0	0	1,692E+04	1,705E+04	**1,704E+04**
i0012	0	0	5,691E+03	5,885E+03	5,984E+03
i0013	3,372E+03	0	3,372E+03	3,374E+03	9,527E+06
i0014	3,748E+03	0	3,832E+03	3,964E+03	1,061E+07
i0015	0	0	2,192E+03	2,449E+03	1,070E+07
i0016	2,377E+03	8,523E+02	2,382E+03	2,437E+03	**2,423E+03**
i0017	3,491E+03	0	3,781E+03	3,838E+03	4,809E+06
i0018	1,570E+02	0	6,084E+03	6,471E+03	**6,465E+03**
i0019	2,804E+03	0	4,055E+03	4,237E+03	**4,402E+03**
i0020	0	0	4,576E+03	4,646E+03	9,600E+06
i0021	0	0	4,245E+03	4,354E+03	1,343E+07

Table 1: lower and upper bounds (resp. "bestFeas" and "bestRelax") obtained using CPLEX's default configuration and the CSSP solution (resp. "*CD*" and "*CSSP*") to configure the solver

We used a Gaussian kernel during SVR training, which is the default choice in absence of any other meaningful prior [6]. This choice makes the CSSP a MINLP with a nonconvex objective function $\bar{p}_{\mathcal{A}}$. More precisely, our CSSP is:

$$\min_{c\in\mathcal{C}_{\mathcal{A}}} \sum_{i=1}^{s} \alpha_i \, e^{-\gamma\|(f_i,c_i)-(\bar{f},c)\|_2^2} \tag{3}$$

where, for all $i \leq s$, (f_i, c_i) belong to the training set, α_i are the dual solutions of the SVR, γ is the scaling parameter of the Gaussian kernel used in Eq. (3).

We tested our approach on 41 instances of the Hydro Unit Commitment problem [4]. Its purpose is to find the scheduling of a pump-storage hydro power station maximizing the revenue given by electricity selling. Each instance was encoded by 54 continuous features, representing hourly electricity prices, hourly inflows, initial and target water level of the considered reservoir. We configured 11 parameters of the IBM ILOG CPLEX MP solver [10]. Our preliminary computational experiments use CPLEX's default configuration and the CSSP solution to solve the instances. Table 1 shows that, in a number of cases, the proposed approach is capable of providing stronger relaxations than CPLEX's default. However, it still fails to find good feasible solutions, which yields overall larger integrality gaps than the solver's default. In order to tackle this issue and improve the general efficacy of the approach, we plan to test variants in the near future. We will most importantly test different features for the instances, employ alternative performance metrics for $p_{\mathcal{A}}$ and conduct experiments with other techniques than SVR.

References

[1] B. Adenso-Díaz and M. Laguna. Fine-tuning of algorithms using Fractional Experimental Design and Local Search. *Operations Research*, 54(1):99–114, 2006.

[2] C. Ansótegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, CP'09, pages 142–157, Berlin, Heidelberg, 2009. Springer-Verlag.

[3] N. Belkhir, J. Dreo, P. Savant, and M. Schoenauer. Feature based algorithm configuration: A case study with differential evolution. In *PPSN XIV*, volume 9921, pages 156–165, 2016.

[4] A. Borghetti, C. D'Ambrosio, A. Lodi, and S. Martello. An MILP approach for short-term hydro scheduling and unit commitment with head-dependent reservoir. *IEEE Transactions on Power Systems*, 23(3):1115–1124, 2008.

[5] M. Brendel and M. Schoenauer. Instance-based parameter tuning for Evolutionary AI Planning. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '11, pages 591–598. ACM, 2011.

[6] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[7] K. Eggensperger, M. Lindauer, and F. Hutter. Pitfalls and best practices in algorithm configuration. *CoRR*, abs/1705.06058, 2017.

[8] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: An automatic algorithm configuration framework. *J. Artif. Int. Res.*, 36(1):267–306, 2009.

[9] F. Hutter and H. Youssef. Parameter adjustment based on performance prediction: Towards an instance-aware problem solver. Technical report, In: Technical Report: MSR-TR-2005125, Microsoft Research, (2005).

[10] IBM. *IBM ILOG CPLEX Optimization Studio CPLEX Parameters Reference*, 2014.

[11] M. Lombardi, M. Milano, and A. Bartolini. Empirical decision model learning. *Artif. Intell.*, 244:343–367, 2017.

[12] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle. The irace package: iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[13] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[14] V. Nannen and A. E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 975–980. Morgan Kaufmann Publishers Inc., 2007.

[15] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[16] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[17] S. Varma and R. Simon. Bias in error estimation when using cross-validation for model selection. *BMC bioinformatics*, 7(91), 2006.

Inductive Shapley values in cooperative transportation games

Reinoud Joosten* & Eduardo Lalla-Ruiz

April 4, 2019

1 Problem and intuition

Applications of cooperative game theory to transportation problems may suffer from a double curse of dimensionality. Firstly, finding several widely used solutions to cooperative games constitutes a problem which is increasingly hard to solve if the number of agents that cooperate increases. Secondly, prominent problems studied in transportation frameworks are traveling salesman problems and vehicle routing problems, and these are known to be *NP*-hard.

One of the most important single-valued solutions for cooperative games, is the Shapley value. A value here solves the problem of how the worth of the grand coalition, i.e., the benefits of all agents cooperating, should be divided. One of the most *unattractive* properties of the Shapley value is that it becomes increasingly hard to compute for games with increasing numbers of players as mentioned. This problem is known to be *NP*-hard.

We consider appointing an outside agent to provide a fair distribution of the gains of cooperation. We have already argued that *CTPs* (cooperative transportation problems) consist of several *NP*-hard subproblems, so the agent should be provided with ample computational means. Our motivational idea is that computation of a solution to the problem of dividing the benefits of cooperation in a *CTP* may be subject to a time constraint. So, the outside agent may not be possible to compute all primitives, i.e., the worths of the coalitions, needed to compute the Shapley value.

We follow a *H(art-&)M(as-Colell)*-potential-based computation/approximation approach. We determine the exact *HM*-potential whenever possible and we approximate the *HM*-potential if it cannot be determined. We define an approximation, the inductive *HM*-potential, based on the true *HM*-potentials for all coalitions with at most U members. The *HM*-potential of

*Both authors**:** University of Twente, School of Behavioral, Management and Social Sciences, IEBIS, POB 217, 7500 Enschede, The Netherlands. **Email**: r.a.m.g.joosten@utwente.nl

the grand coalition is approximated by using the true worth of the grand coalition, which by assumption can always be computed, and the approximated potentials of all coalitions with one player missing.

In some cases, the inductive Shapley value is identical to well known alternatives. If the restriction is quite loose the Shapley value and the inductive Shapley value coincide. If the restriction is very tight, i.e., $U = 0$, the inductive Shapley value and the egalitarian value coincide, and if it is slightly less tight, i.e., $U = 1$, the former coincides with the *CIS* value (center of the imputation set).

The inductive Shapley value satisfies efficiency and the balanced contributions property for all games. Furthermore, depending on diligent choices regarding how to establish the approximated *HM*-potential, the inductive Shapley value satisfies symmetry for all games. Symmetry and the balanced contributions property are widely regarded as requirements of *fairness*. Another *fairness* aspect, social acceptability can be guaranteed only for subclasses of games. We introduce two new axioms of *fairness*, satisfied by the inductive Shapley value by design, i.e., sensitive up to any cardinality U and insensitive beyond any cardinality U.

2 Variations on Shapley values and potentials

A well known solution to cooperative games is the **Shapley value** Sh, for every $(N, v) \in G$ and every $i \in N$, given by

$$Sh_i(N, v) = \sum_{S \subseteq N : i \in S} \frac{(|S| - 1)!(|N| - |S|)!}{|N|!} \Delta_i^v(S). \tag{1}$$

The Shapley value is uniquely determined by efficiency, symmetry, linearity and the null-player property.

The *HM*-potential is a function attributing a real number to each game. The vector of marginal contributions of each player to the *HM*-potential of the grand coalition in a game coincides with the Shapley value. We may present this potential and an associated value as a pair.

Definition 1 *The HM-potential is the unique map* $P : G \to \mathbb{R}$ *given by* $P(N, v) = 0$ for $N = \emptyset$, and for $N \neq \emptyset$

$$\sum_{i \in N} [P(N, v) - P(N \backslash \{i\}, v)] = v(N), \text{ for all } (N, v) \in G.$$

The value associated to the HM-potential is the Shapley value Sh for all $(N, v) \in G$, $i \in N$ *given by*

$$Sh_i(N, v) = P(N, v) - P(N \backslash \{i\}, v).$$

Alternatively, the *HM*-potential is uniquely determined recursively by

$$P(M,v) = \frac{v(M) + \sum_{k\in N} P(M\backslash\{k\},v)}{|M|} \tag{2}$$

on every game (M,v) restricted to the subset $M \subseteq N$.

2.1 Inductive potentials and values

Let for given game (N,v) and let for given $U \subseteq N : V^U = \{P(S,v)|\ 0 \leq |S| \leq U\}$ and let $\{V^u\}_{u=0}^{|N|}$ denote the collection of all such sets, and let $g : \{V^u\}_{u=0}^{|N|} \to \mathbb{R}^{|N|}$. Then, we define the inductive *HM*-potential (with restriction U) $P^U : 2^N \to \mathbb{R}$ in a rather general manner:

$$\begin{aligned} P^U(S,v) &= P(S,v) \text{ if } |S| \leq U, \\ P^U(S,v) &= \sum_{j\in S} g_j(V^U) \text{ if } U < |S| \leq |N| - 1, \\ P^U(N,v) &= \frac{v(N) + \sum_{k\in N} P^U\left(N\backslash\{k\},v\right))}{|N|}. \end{aligned} \tag{3}$$

Note that if $U = |N|, |N| - 1$ all values of the inductive *HM*-potential $P^U(S,v)$ are equal to the *HM*-potentials $P(S,v)$, $S \in 2^N$. For the more challenging case that $U \leq |N| - 2$, we approximate the *HM*-potential for all S with $|S| > U$. For the final approximation namely $P^U(N,v)$, we turn back to reality by using the truly computed worth of the grand coalition, but still rely on the approximations of potentials for coalitions with one member less than the grand coalition.

Remark 1 *In order to avoid confusion we emphasize the following. The inductive Shapley value $ISh^U(N,v)$ is uniquely determined for each and every game and for $U = 0, 1, ..., |N|$ and g, so the set*

$$\{ISh^u(N,v)\}_{u=0}^{|N|}$$

is uniquely determined. However, only one element out of this set is chosen by the procedure and as the number U is not known in advance we have

$$ISh(N,v) = ISh^U(N,v) \in \{ISh^u(N,v)\}_{u=0}^{|N|}.$$

3 The computational procedure

Let CT_{max} be the maximum computing time allowed, let t_c denote the real-time running computation time. Now, we proceed with a description of the procedure in pseudo code.

Step 0 Compute $v(N)$, then

- $P^0(\emptyset, v) := 0$,
- $V^0 := \{P^0(\emptyset, v)\}$,
- $P^0(N\backslash\{i\}, v) := 0$ for all $i \in N$;
- $P^0(N, v) := v(N)$,
- $ISh_i^0(N, v) := P^0(N, v) - P^0(N\backslash\{1\}, v)$ for all $i \in N$;
- $ISh^0(N, v) := (ISh_1^0(N, v), ..., ISh_{|N|}^0(N, v))$;
- $K := 1$ and go to Step K.

Step K While $t_c < CT_{max}$,

- Compute $v(S)$ for all $S \subseteq N$, $|S| = K$,
- $P^K(S, v) := \frac{v(S)+\sum_{j\in S} P^{K-1}(S\backslash\{j\},v\}}{|S|}$ for all $S \subseteq N$, $|S| = K$;
- $V^K := V^{K-1} \cup \{P^K(S, v)|\ S \subseteq N,\ |S| = K\}$;
- $g_i^1\left(V^K\right) := \frac{1}{\binom{|N|-1}{K-1}} \sum_{S:|S|=K, \atop i\in S} (P^K(S, v) - P^K(S\backslash\{k\}, v)$ for $i \in N$;
- $P^K(N\backslash\{i\}, v) := \sum_{k\in N\backslash\{i\}} g_j^1\left(V^K\right)$ for all $i \in N$;
- $P^K(N, v) := \frac{v(N)+\sum_{j\in N} P^K(N\backslash\{j\},v)}{|N|}$;
- $ISh_i^K := P^K(N, v) - P^K(N\backslash\{i\}, v)$ for all $i \in N$;
- $ISh^K := \left(ISh_1^K, ..., ISh_{|N|}^K\right)$.

If $t_c = CT_{max}$, then **stop** with $ISh(N, v) = ISh^{K-1}(N, v)$.
Otherwise, if $K = |N|$, then **stop** with $ISh(N, v) = Sh(N, v)$.
Otherwise, set $K := K+1$, $ISh(N, v) := ISh^K(N, v)$, go to Step K. ■

The procedure stops in two cases with an inductive Shapley value $ISh(N, v)$. In one case the time constraint proved not binding, hence $U = |N|$ and $ISh(N, v) = Sh(N, v)$. In the other case, the time constraint was binding and $U < |N|$ and $ISh(N, v) = ISh^U(N, v)$. For the latter case, all information is used for potentials and worths up of coalitions up to cardinality U, and due to the failure to complete the computations on time, information available for larger coalitions is neglected with the exception of using the information on the worth of the grand coalition.

Combinatorial optimization in structural engineering: recent trends and future needs

Saeid Kazemzadeh Azad[1]

[1]Atilim University, Department of Civil Engineering, Ankara, Turkey
saeid.azad@atilim.edu.tr

Abstract

During the past decades optimization has received considerable attention in the literature of structural engineering. Generally, the optimum design of a structural system can be defined as seeking the best arrangement of structural elements that produces an economical final solution while satisfying a set of design constraints stipulated by the standard design codes. In civil engineering applications, typically, the optimum design of skeletal structures is carried out with respect to a discrete list of available sections, resulting in a combinatorial sizing optimization problem. In essence, development of optimization algorithms for handling such discrete optimization problems is basically due to the fact that the speed of existing computers is not high enough to facilitate evaluating every possible solution in a timely manner. Therefore, search techniques capable of generating reasonable solutions without performing an exhaustive search have become popular in the structural engineering applications.

Unquestionably, the majority of the contemporary approaches proposed for discrete sizing optimization of skeletal structures belong to the class of metaheuristic techniques. Regarding the popularity of metaheuristics in structural engineering applications, the present study strives to survey the recently developed structural optimization metaheuristics and outlines the advantages and shortcomings of these techniques as well as the future research needs in discrete sizing optimization applications.

Keywords: Optimum design, Structural engineering, Combinatorial optimization, Metaheuristics

Efficient Algorithms for the Recoverable (Robust) Selection Problem

Thomas Lachmann[1] and Stefan Lendl[2]

[1]Institute of Analysis and Number Theory, Graz University of Technology
[2]Institute of Discrete Mathematics, Graz University of Technology

1 Introduction

The Selection problem is widely studied in the computer science and optimization literature. Let $E = [n]$ be the set of base elements, with associated non-negative costs α_i for each $i \in E$. Given an integer $p \in [n]$ we want to choose a subset $A \subseteq E$ of size p that minimizes the cost $\sum_{i \in X} \alpha_i$. The problem can be solved in $O(n)$ time using a linear time algorithm for finding the p-th largest element [1]. We denote by $\mathcal{S}_p^\alpha(E)$ an arbitrary instance of this problem and write $A = \mathcal{S}_p^\alpha(E)$ if A is an optimal solution to the given instance of the selection problem.

In this paper we investigate a natural generalization of the Selection problem. For the element set $E = [n]$ we are given two cost vectors $\alpha_i, \beta_i \in \mathbb{R}$ and parameters $p, q \in \mathbb{N}$. We have to select $A, B \subseteq E$ both of size p that minimize the cost $\sum_{i \in A} \alpha_i + \sum_{j \in B} \beta_j$ such that the intersection $|A \cap B| \geq q$. We denote an instance of this problem, the Recoverable Selection problem, by $\mathcal{RS}_{p,q}^{\alpha,\beta}(E)$ and if (A^*, B^*) is an optimal solution of this problem we write $(A^*, B^*) = \mathcal{RS}_{p,q}^{\alpha,\beta}(E)$. Kasperski and Zieliński [2] showed that this problem can be solved in $O(qn^2)$ time using a reduction to a minimum cost flow problem.

Our Results We show in Section 2 that the Recoverable Selection problem can be solved using a simple greedy algorithm. This is of special interest since for this problem no matroidal structure is known that would directly imply such a result. In Section 3 we study the structure of optimal solutions with respect to two parameters and obtain discrete-convexity and unimodality results. Based on this detailed mathematical analysis we are able to obtain a linear time algorithm using prune and search [5].

Recoverable Robust Optimization – an Application The concept of recoverable robust optimization was introduced by Liebchen et al. [4]. The Recoverable Selection problem can be used to solve the recoverable robust version of the classic Selection problem with interval uncertainties [2]. In this case, instead of fixed costs for each element, we are given a scenario set $\mathcal{U}$ and for each scenario $s \in \mathcal{U}$ the costs of element $i \in E$ are denoted by $c_i^s \geq 0$ and setup costs C_i. In the robust optimization literature the interval uncertainty representation is a popular choice for defining scenario sets [3], which we denote by $\mathcal{U}^I$. For each $i \in E$ we are given an interval $[\underline{c}_i, \bar{c}_i]$ of possible cost realizations. Then the scenario set with interval uncertainties is given by $\mathcal{U}^I = \prod_{i \in E} [\underline{c}_i, \bar{c}_i]$. Then the Recoverable Robust Selection problem to be solved is

$$\min_{X \subseteq E:\ |X|=p} \max_{s \in \mathcal{U}^I} \sum_{i \in X} C_i + \min_{\substack{Y \subseteq E, |Y|=p \\ |Y \setminus X| \leq k}} \sum_{i \in Y} c_i^s.$$

Kasperski and Zieliński [2] observed that this is equivalent to solving $\mathcal{RS}^{C,\bar{c}}_{p,p-k}(E)$.

2 A Greedy Algorithm

Algorithm 1 is a, easy to implement, greedy algorithm for the Recoverable Selection problem.

Algorithm 1: Greedy algorithm with growing selection parameter.

```
1 A := ∅, B := ∅
2 for l := 1, ..., p do
3 |   (a, b) := argmin{α_i + β_j : (i, j) ∈ E^2, i ∈ E \ A, j ∈ E \ B,
  |                                  |(A + i) ∩ (B + j)| ≥ q − (p − l)}
4 |   A := A ∪ {a}
  |   B := B ∪ {b}
5 return (A, B)
```

To obtain an efficient implementation and prove correctness, it is necessary to classify the different cases in which the minimum in line 3 in each iteration of the loop in line 2 can occur:

α-greedy and β-greedy This is the case if we select $a = \operatorname{argmin}\{\alpha_i \colon i \in E \setminus A\}$ and $b = \operatorname{argmin}\{\beta_j \colon j \in E \setminus B\}$.

α-greedy and β-filling In this case we select $a = \operatorname{argmin}\{\alpha_i \colon i \in E \setminus A\}$, and dependent on a then $b = \operatorname{argmin}\{\beta_i \colon i \in E \setminus B, i \in A \cup \{a\}\}$. We call b the filling element of the step.

β-greedy and α-filling The symmetric case, i.e. $b = \operatorname{argmin}\{\beta_j \colon j \in E \setminus B\}$, and dependent on b then $a = \operatorname{argmin}\{\alpha_i \colon i \in E \setminus A, i \in B \cup \{b\}\}$. We call a the filling element.

$(\alpha+\beta)$-greedy The step $a = b = \operatorname{argmin}\{\alpha_i + \beta_i \colon i \in E \setminus (A \cup B)\}$.

Based on this case distinction we can implement Algorithm 1 in $O(n \log n)$ time using sorting and priority queues. In summary we obtain the following result.

Theorem 1. *The greedy algorithm (Algorithm 1) solves the* Recoverable Selection *problem in $O(n \log n)$ time.*

3 A Linear Time Algorithm

To achieve a linear running time we repeatedly use the fact that $\mathcal{S}^{\alpha}_{p}(E)$ can be calculated in $O(n)$ time. The main idea is to introduce parameters for different structural properties of solutions and then perform prune and search for the optimal values of those parameters in a two stage approach. Before these structural parameters can be introduced we perform a simple preprocessing, removing trivial to select pairs of elements. This preprocessing procedure achieves that given an arbitrary instance (E, p, q, α, β) it obtains an instance $(E', p', q', \alpha, \beta)$ with the properties

- $p' \leq \hat{p} \leq p$,
- $X' = \mathcal{S}^{\alpha}_{\hat{p}}(E')$,
- $Y' = \mathcal{S}^{\beta}_{\hat{p}}(E')$,
- $E' = X' \cup Y'$,
- $X' \cap Y' = \emptyset$.

Prune and Search for the Intersection Size s in X

The main idea of the algorithm is to introduce a parameter $s \in \{0, 1, \ldots, q\}$, where $s = |X \cap A \cap B|$ (symmetrically, then $q - s = |Y \cap A \cap B|$). For every $s \in \{0, 1, \ldots, q\}$ let $A^s_1, B^s_1 \subseteq X$ be such that $|A^s_1| = p - (q - s), |B^s_1| = s$ minimizing $\sum_{i \in A^s_1} \alpha_i + \sum_{j \in B^s_1} \beta_j$ and $A^s_1 \cap B^s_1 = B^s_1$ and $A^s_2, B^s_2 \subseteq Y$

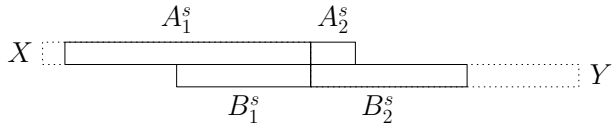


Figure 1: Illustration of the sets involved in the search for s^*.

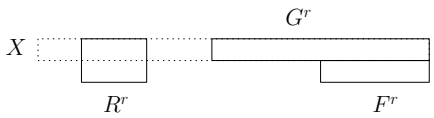


Figure 2: Visualization of sets involved in the definition of $h(r)$.

such that $|B_2^s| = p - s, |A_2^s| = q - s$ and $B_2^s \cap A_2^s = A_2^s$ minimizing $\sum_{i \in A_2^s} \alpha_i + \sum_{j \in B_2^s} \beta_j$. This directly implies that $(A, B) = (A_1^s \cup A_2^s, B_1^s \cup B_2^s)$ is a feasible solution to $\mathcal{RS}_{p,q}^{\alpha,\beta}(E)$ with exactly s elements of the intersection in X and $q - s$ in Y. In addition we observe that there is an s such that $(A_1^s \cup A_2^s, B_1^s \cup B_2^s)$ is an optimal solution, which we denote by s^*.

We analyze the cost function with respect to this parameter s. For this purpose let

$$f(s) = \sum_{i \in A_1^s \cup A_2^s} \alpha_i + \sum_{j \in B_1^s \cup B_2^s} \beta_j.$$

Our main structural result for $f(s)$ is that $f(s)$ is a discrete-convex function, i.e. $2f(s) \leq f(s-1) + f(s+1)$ for all $s \in \{1, 2, \ldots, q-1\}$. A discrete function f is called unimodal if there is some s' such that for all $s \leq s'$ it holds that $f(s)$ is monotonically decreasing in s and for all $s \geq s'$ it holds that $f(s)$ is monotonically increasing in s. A plateau of f is a sequence $s_1, s_2, \ldots, s_l$ with $l > 1$ such that $f(s_1) = f(s_2) = \cdots = f(s_l)$. It is a well known result that a discrete-convex function is unimodal and has at most one plateau at its minimum.

Based on this it is possible to efficiently check if $s = s^*$, $s^* > s$ or $s^* < s$ for any given s by determining $f(s+1)$ and $f(s-1)$, which results in a fast prune and search for s^*, because determining $f(s-1)$ and $f(s+1)$ can be done in the same time as determining $f(s)$ (see below).

Prune and Search for the Number of $(\alpha + \beta)$-Greedy Steps

In this section we explain how to find sets $A_1^s, B_1^s \subseteq X$ during the search for s^* in $O(\Delta s)$ time, where Δs is the current size of the search region. The case of finding A_2^s, B_2^s follows symmetrically.

Again, the main idea of the algorithm is to introduce a parameter $r \in \{0, 1, \ldots, s\}$, that can be loosely interpreted as the number of $(\alpha+\beta)$-greedy steps performed. Based on r we define the sets
• $G^r = \mathcal{S}_\alpha^{p-q+s-r}(X)$, • $F^r = \mathcal{S}_\beta^{s-r}(G^r)$, • $R^r = \mathcal{S}_{\alpha+\beta}^r(X \setminus G^r)$.

Here, it is important that for R^r if there are multiple elements with same sum $\alpha + \beta$ we select the one with smaller α value. If also this is the same (i.e. there are multiple elements with identical costs) we use the same order as is used to select G^r with respect to α.

Observe that $(G^r \cup R^r, F^r \cup R^r)$ is a feasible solution for the sets (A_1^s, B_1^s) (see Figure 2). We denote the cost of this solution by

$$h(r) = \sum_{i \in G^r \cup R^r} \alpha_i + \sum_{j \in F^r \cup R^r} \beta_j.$$

It is easy to see that there is always an optimal solution of the form $(G^r \cup R^r, F^r \cup R^r)$ for some correctly chosen r, which we denote by r^*. To find those solutions efficiently we first analyze

Figure 3: Visualization of the functions f (left) and h (right).

properties of the function $h(r)$ in terms of r, similarly as we did for $f(s)$. For $h(r)$ we show unimodality, but observe that it is not discrete convex as it is the case for $f(s)$ (see Figure 3 for a comparison of the structure of $f(s)$ and $h(r)$).

We show how to perform a prune and search for r^* and the corresponding optimal solutions A_1^s, B_1^s. The basic idea is again similar as in the search for s^*. The major difference in this case is that long plateaus can appear in $h(r)$. This is why it does not suffice to just calculate the values $h(r-1), h(r), h(r+1)$, since if they are equal we cannot efficiently decide whether $r^* < r, r^* > r$ or $r^* = r$. This is why instead of evaluating at $r-1$ and $r+1$ in addition to the center point r the evaluation is performed at the quarter points $r^{\triangleleft}, r^{\triangleright}$ to the left and right. The major complication in this binary search are the cases where we have equality among the evaluation points. We can handle this by proving additional properties about the plateaus of $h(r)$. Using this, pruning either happens because we know that all the elements in between two evaluation points build a plateau or since the properties of unimodality imply that the minimum cannot lie in some regions.

To achieve the claimed running time of $O(\Delta s)$, it is not feasible to select from the whole sets X, G^r and $X \setminus G^r$ during the execution of the algorithm. We are able to handle this efficiently by showing that certain sets of elements can be fixed and the selection can be performed on smaller sets of candidate elements. Based on this we obtain our main result.

Theorem 2. *The* Recoverable Selection *problem can be solved in $O(n)$ time.*

Acknowledgement. We would like to thank Gerhard Woeginger for posing this problem and several helpful discussions. Stefan Lendl acknowledges the support of the Austrian Science Fund (FWF): W1230. Thomas Lachmann acknowledges the support of the Austrian Science Fund (FWF): Y-901.

References

[1] Manuel Blum, Robert W Floyd, Vaughan Pratt, Ronald L Rivest, and Robert E Tarjan. Time bounds for selection. *Journal of computer and system sciences*, 7(4):448–461, 1973.

[2] Adam Kasperski and Paweł Zieliński. Robust recoverable and two-stage selection problems. *Discrete Applied Mathematics*, 233:52–64, 2017.

[3] Panos Kouvelis and Gang Yu. *Robust discrete optimization and its applications*, volume 14. Springer Science & Business Media, 2013.

[4] Christian Liebchen, Marco Lübbecke, Rolf Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In *Robust and online large-scale optimization*, pages 1–27. Springer, 2009.

[5] Nimrod Megiddo. Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM journal on computing*, 12(4):759–776, 1983.

Matroid Sum with Cardinality Constraints on the Intersection

Stefan Lendl[1], Britta Peis[2], and Veerle Timmermans[2]

[1]Graz University of Technology, Institute of Discrete Mathematics
[2]RWTH Aachen, Department of Management Science,

1 Introduction

Since their introduction in the 1930s, the theory of matroids and submodular functions have become an integral part of discrete optimization. This combinatorial stucture naturally arises in practical settings, e.g., they generalize the notion of linear independence in vector spaces and the notion of forests in graphs. Formally, a *matroid* $\mathcal{M}$ is a pair $(E, \mathcal{I})$, where E is a finite set of resources and $\mathcal{I}$ is a family of subsets of E, called the *independent sets*. Set $\mathcal{I}$ has the following three properties:

1. The empty set is an independent set: $\emptyset \in \mathcal{I}$.
2. Set $\mathcal{I}$ is closed under taking subsets: if $I \subseteq J$ and $J \in \mathcal{I}$, then $I \in \mathcal{I}$.
3. Set $\mathcal{I}$ has the *exchange property*: if $I, J \in \mathcal{I}$ and $|I| < |J|$, then there exists an $e \in J$ such that $I \cup \{e\} \in \mathcal{I}$.

A *basis* is an independent set that becomes dependent on adding any element of E and the *base set* $\mathcal{B}$ contains all bases of $(E, \mathcal{I})$. Given the base set $\mathcal{B}$, one can easily recover the original set $\mathcal{I}$. Hence, a matroid $\mathcal{M}$ can also be denoted as $(E, \mathcal{B})$. For more details on matroids, we refer to [4].

The Model Given two matroids $\mathcal{M}_1 = (E, \mathcal{B}_1)$ and $\mathcal{M}_2 = (E, \mathcal{B}_2)$ on a common ground set E with base sets $\mathcal{B}_1$ and $\mathcal{B}_2$, some integer $k \in \mathbb{N}$, and two cost functions $c_1, c_2 \colon E \to \mathbb{R}$, we consider the optimization problem to find a base $X \in \mathcal{B}_1$ and a base $Y \in \mathcal{B}_2$ minimizing $c_1(X) + c_2(Y)$ subject to either a lower bound constraint $|X \cap Y| \leq k$, an upper bound constraint $|X \cap Y| \geq k$, or an equality constraint $|X \cap Y| = k$ on the size of the intersection of the two bases X and Y. Here, as usual, we write $c_1(X) = \sum_{e \in X} c_1(e)$ and $c_2(Y) = \sum_{e \in Y} c_2(e)$ to shorten notation. Let us denote the following problem by $(P_{=k})$.

$$\begin{aligned} \min\ & c_1(X) + c_2(Y) \\ \text{s.t. } & X \in \mathcal{B}_1 \\ & Y \in \mathcal{B}_2 \\ & |X \cap Y| = k \end{aligned}$$

Accordingly, if constraint $|X \cap Y| = k$ is replaced by upper bound constraint $|X \cap Y| \leq k$, the problem is called $(P_{\leq k})$, and finally, if constraint $|X \cap Y| = k$ is replaced by the lower bound constraint $|X \cap Y| \geq k$, the problem is called $(P_{\geq k})$. Certainly, it only makes sense to consider integers k in the range between 0 and $K := \min\{\mathrm{rk}(\mathcal{M}_1), \mathrm{rk}(\mathcal{M}_2)\}$, where $\mathrm{rk}(\mathcal{M}_i)$ for $i \in \{1, 2\}$ is the rank of matroid $\mathcal{M}_i$, i.e., the cardinality of each basis in $\mathcal{M}_i$ which is unique.

The recoverable robust matroid basis problem. There is a strong connection between the model described in this paper and the recoverable robust matroid base problem (RecRobMatroid) studied in, e.g., the PhD thesis of Christina Büsing [1]. In RecRobMatroid, we are given a matroid $\mathcal{M} = (E, \mathcal{B})$ on a ground set E with base set $\mathcal{B}$, some integer $k \in \mathbb{N}$, a first stage cost function c_1 and an uncertainty set $\mathcal{U}$ that contains different scenarios S, where each scenario $S \in \mathcal{U}$ gives a possible second stage cost $S = (c_S(e))_{e \in E}$.

RecRobMatroid then consists out of two phases: in the first stage one needs to pick a base X. Then, the scenario $S \in \mathcal{U}$ is revealed and there is a recovery stage, where one needs to pick a second basis Y that differs at most k elements from the original basis X: $|X \cap Y| \geq rk(\mathcal{M}) - k$. The goal is to minimize the worst-case cost $c_1(X) + c_S(Y)$. The recoverable robust matroid basis problem can be written as follows:

$$\min_{X \in \mathcal{B}} \left(c_1(X) + \max_{S \in \mathcal{U}} \min_{\substack{Y \in \mathcal{B} \\ |X \cap Y| \geq rk(\mathcal{M}) - k}} c_S(Y) \right) \tag{1}$$

There are several ways in which the uncertainty set $\mathcal{U}$ can be represented, and one popular way is the *interval uncertainty representation*. In this representation, we assume that the uncertainty set $\mathcal{U}$ can be represented by a set of $|E|$ intervals:

$$\mathcal{U} = \left\{ S = (c_S(e))_{e \in E} \mid c_S \in [c'(e), c'(e) + d(e)], e \in E \right\}$$

In the worst-case scenario $\bar{S}$ we have for all $e \in E$ that $c_{\bar{S}}(e) = c'(e) + d(e)$. When we define $c_2(e) := c_{\bar{S}}(e)$, it is clear that recoverable robust matroid base problem is a special case of $(P_{\geq})$, in which $\mathcal{B}_1 = \mathcal{B}_2$.

Büsing presented an algorithm for RecRobMatroid which is exponential in k. In 2017, Hradovich, Kaperski, and Zielinski [3] proved that RecRobMatroid can be solved in polynomial time via some iterative relaxation algorithm and asked for a strongly polynomial time algorithm. Shortly after, the same authors presented in [2] a strongly polynomial time primal dual algorithm for the special case of RecRobMatroid on a graphical matroid. The question whether a strongly polynomial time algorithm for RecRobMatroid on general matroids exists was posed as an open question.

2 Our Contribution

Reduction of $(P_{\leq k})$ and $(P_{\geq k})$ to weighted matroid intersection. We first note that $(P_{\leq k})$ and $(P_{\geq k})$ are computationally equivalent. To see this, consider any instance $(\mathcal{M}_1, \mathcal{M}_2, k, c_1, c_2)$ of $(P_{\geq k})$, where $\mathcal{M}_1 = (E, \mathcal{B}_1)$, and $\mathcal{M}_2 = (E, \mathcal{B}_2)$ are two matroids on the same ground set E with base sets $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively. Define $c_2^* = -c_2$, $k^* = \mathrm{rk}(\mathcal{M}_1) - k$, and let $\mathcal{M}_2^* = (E, \mathcal{B}_2^*)$ with $\mathcal{B}_2^* = \{E \setminus Y \mid Y \in \mathcal{B}_2\}$ be the dual matroid of $\mathcal{M}_2$. Since

(i) $|X \cap Y| \leq k \iff |X \cap (E \setminus Y)| = |X| - |X \cap Y| \geq \mathrm{rk}(\mathcal{M}_1) - k = k^*$, and

(ii) $c_1(X) + c_2(Y) = c_1(X) + c_2(E) - c_2(E \setminus Y) = c_1(X) + c_2^*(E \setminus Y) + c_2(E)$,

where $c_2(E)$ is a constant, it follows that (X, Y) is a minimizer of $(P_{\geq k})$ if and only if $(X, E \setminus Y)$ is a minimizer of $(P_{\leq k^*})$ for the instance $(\mathcal{M}_1, \mathcal{M}_2^*, k^*, c_1, c_2^*)$, and vice versa. Similarly, it can be shown that any problem of type $(P_{\leq k})$ polynomially reduces to an instance of type $(P_{\geq k^*})$.

We show that, $(P_{\leq k})$ (and, hence, also $(P_{\geq k})$), can be polynomially reduced to weighted matroid intersection. Since weighted matroid intersection can be solved in strongly polynomial time by some very elegant primal-dual algorithm (cf. Lawler 1970), this answers the open question raised in [3] affirmatively

A strongly polynomial primal-dual algorithm for $(P_{=k})$. As we can solve matroid sum with both, lower and upper bound constraints on the intersection of the bases, the question arises whether or not the problem with equality constraint $(P_{=k})$ can be solved in strongly polynomial time. Also here we give an affirmative answer, and we provide a strongly polynomial time algorithm that constructs an optimal solution for $(P_{=k})$. Our algorithm can be seen as a generalization of the algorithm presented by Hradovich et al. in [3]. However, the analysis of our algorithm turns out to be much simpler than the one in [3].

The general idea of our algorithm is as follows. First, we find the optimal solution for

$$\min\{c_1(X) + c_2(Y) \mid X \in \mathcal{B}_1,\ Y \in \mathcal{B}_2\}$$

without any constraint on the intersection. This problem can be solved with a matroid greedy algorithm in $\mathcal{O}(m \log m)$ time, where $m = |E|$. Let $(\bar{X}, \bar{Y})$ be an optimal solution of this matroid sum problem.

1. If $|\bar{X} \cap \bar{Y}| = k$, we are done.
2. Else, if $|\bar{X} \cap \bar{Y}| = k' < k$, our algorithm starts with the optimal solution $(\bar{X}, \bar{Y})$ for $(P_{=k'})$, and iterative increases k' by one until $k' = k$. Our algorithm maintains as invariant an optimal solution $(\bar{X}, \bar{Y})$ for the current problem $(P_{=k'})$.
3. Else, if $|\bar{X} \cap \bar{Y}| > k$, we instead consider an instance of $(P_{=k^*})$ for $k^* = \mathrm{rk}(\mathcal{M}_1) - k$, costs c_1 and $c_2^* = -c_2$, and the two matroids $\mathcal{M}_1 = (E, \mathcal{B}_1)$ and $\mathcal{M}_2^* = (E, \mathcal{B}_2^*)$. As seen above, an optimal solution $(X, E \setminus Y)$ of problem $(P_{=k^*})$ corresponds to an optimal solution (X, Y) of our original problem $(P_{=k})$, and vice versa. Moreover, $|\bar{X} \cap \bar{Y}| > k$ for the initial base pair $(\bar{X}, \bar{Y})$ implies that $|\bar{X} \cap (E \setminus \bar{Y})| = |\bar{X}| - |\bar{X} \cap \bar{Y}| < k^*$. Thus, starting with the initial feasible solution $(\bar{X}, E \setminus \bar{Y})$ for $(P_{=k^*})$, we can iteratively increase $|\bar{X} \cap (E \setminus \bar{Y})|$ until $|\bar{X} \cap (E \setminus \bar{Y})| = k^*$, as described in step 2.

The difficulty of the algorithm is to recompute the optimal solution when iteratively increasing k. To tackle this problem, we do not only keep track of optimal solution $(\bar{X}, \bar{Y})$ for a certain k, but also a solution of the dual that satisfies some optimality conditions. Then, by a sequence of primal and dual updates we are able to find a min-cost solution such that the size of the intersection increased by one.

Hardness of polymatroid base problems. Finally, we consider the generalization of the problems from matroids to polymatroids. Recall that a function $f : 2^E \to \mathbb{R}$ is called *submodular* if $f(U) + f(V) \geq f(U \cup V) + f(U \cap V)$ for all $U, V \subseteq E$. Function f is called *monotone* if $f(U) \leq f(V)$ for all $U \subseteq V$, and *normalized* if $f(\emptyset) = 0$. Given a submodular, monotone and normalized function f, the pair (E, f) is called a *polymatroid*, and f is called *rank function* of the polymatroid (E, f). The associated *polymatroid base polytope* is defined as:

$$\mathcal{B}(f) := \left\{ x \in \mathbb{R}_+^{|E|} \mid x(U) \leq f(U)\ \forall U \subseteq E,\ x(E) = f(E) \right\},$$

where, as usual, $x(U) := \sum_{e \in U} x_e$ for all $U \subseteq E$.

Bases of a polymatroid base polytope are not necessarily $0-1$ vectors anymore. Generalizing set-theoretic intersection and union from sets (a.k.a. $0-1$ vectors) to arbitrary vectors can be done via the following binary operations, called meet and join: given two vectors $x, y \in \mathbb{R}^{|E|}$ the meet of

x and y is $x \wedge y := (\min\{x_e, y_e\})_{e \in E}$, and the join of x and y is $x \vee y := (\max\{x_e, y_e\})_{e \in E}$. Instead of the size of the intersection, we now talk about the size of the meet, abbreviated by

$$|x \wedge y| := \sum_{e \in E} \min\{x_e, y_e\}.$$

Let f_1, f_2 be two polymatroid rank functions with associated polymatroid base polytopes $\mathcal{B}(f_1)$ and $\mathcal{B}(f_2)$, let $c_1, c_2 : E \to \mathbb{R}$ be two cost functions on E, and let k be rational number. The following problem, which we denote by $(\hat{P}_{\geq k})$, is a direct generalization of $(P_{\geq k})$ from matroids to polymatroids.

$$\begin{aligned} \min \ & \sum_{e \in E} c_1(e)x(e) + \sum_{e \in E} c_2(e)y(e) \\ \text{s.t. } & x \in \mathcal{B}(f_1) \\ & y \in \mathcal{B}(f_2) \\ & |x \wedge y| \geq k \end{aligned}$$

Interestingly, it turns out that $(\hat{P}_{\geq k})$ can be reduced to an instance of the *polymatroidal flow problem*, which is known to be computationally equivalent to a submodular flow problem and can thus be solved in strongly polynomial time. Afterwards, we show that the two problems $(\hat{P}_{\leq k})$ and $(\hat{P}_{=} k)$, which can be obtained from $(\hat{P}_{\geq k})$ by replacing constraint $|x \wedge y| \geq k$ by either $|x \wedge y| \leq k$, or $|x \wedge y| = k$, respectively, are weakly NP-hard.

Acknowledgement. We would like to thank András Frank, Björn Tauer and Thomas Lachmann for several helpful discussions about this topic. We also thank Jannik Matuschke, Tom McCormick, Rico Zenklusen, Satoru Iwata, Mohit Singh, Michel Goemans, and Guyla Pap for fruitful discussions at the HIM workshop in Bonn, and at the workshop on Combinatorial Optimization in Oberwolfach.

Stefan Lendl acknowledges the support of the Austrian Science Fund (FWF): W1230.

References

[1] Christina Büsing. *Recoverable robustness in combinatorial optimization.* Cuvillier Verlag, 2011.

[2] Mikita Hradovich, Adam Kasperski, and Paweł Zieliński. Recoverable robust spanning tree problem under interval uncertainty representations. *Journal of Combinatorial Optimization*, 34(2):554–573, 2017.

[3] Mikita Hradovich, Adam Kasperski, and Paweł Zieliński. The recoverable robust spanning tree problem with interval costs is polynomially solvable. *Optimization Letters*, 11(1):17–30, 2017.

[4] James G Oxley. *Matroid theory*, volume 3. Oxford University Press, USA, 2006.

Stationary Nash Equilibria Conditions for Stochastic Positional Games

Dmitrii Lozovanu[1] and Stefan Pickl[2]

[1]Institute of Mathematics and Computer Science, Moldova; e-mail:lozovanu@math.md
[2]Universität der Bundeswehr, Germany; e-mail:stefan.pickl@unibw.de

1 Introduction and Problem Formulation

We consider a class of stochastic positional games that generalizes the mean payoff games on graphs introduced by Ehrenfeucht and Mycielski [2] and considered by Gurvich et al [5] and Alpern [1]. We specify the considered class of stochastic positional games by applying the concept of positional games to Markov decision processes with average and discounted optimization criteria. An m-player average stochastic positional game consists of the following elements:

- a state space X (which we assume to be finite);
- a partition $X = X_1 \cup X_2 \cup \cdots \cup X_m$, where X_i represents the position set of the players $i \in \{1, 2, \ldots, m\}$;
- a finite set $A(x)$ of actions in each state $x \in X$;
- a step reward $f^i(x, a)$ with respect to each player $i \in \{1, 2, \ldots, m\}$ in each state $x \in X$ and for an arbitrary action $a \in A(x)$;
- a transition probability function $p : X \times \prod_{x \in X} A(x) \times X \to [0, 1]$ that gives the probability transitions $p^a_{x,y}$ from an arbitrary $x \in X$ to an arbitrary $y \in X$ for a fixed action $a \in A(x)$, where $\sum_{y \in X} p^a_{x,y} = 1, \ \forall x \in X, \ a \in A(x)$;
- a starting state $x_0 \in X$.

The game proceeds in a sequence of stages and starts in the state x_0 at the moment of time $t = 0$. The player $i \in \{1, 2, \ldots, m\}$ who is the owner of the state position x_0 ($x_0 \in X_i$) chooses an action $a_0 \in A(x_0)$ and determines the rewards $f^1(x_0, a_0), f^2(x_0, a_0), \ldots, f^m(x_0, a_0)$ for the corresponding players $1, 2, \ldots, m$. After that the game passes to a state $y = x_1 \in X$ according to a probability distribution $\{p^{a_0}_{x_0,y}\}$. At the moment of time $t = 1$ the player $k \in \{1, 2, \ldots, m\}$ who is the owner of the state position x_1 ($x_1 \in X_k$) chooses an action $a_1 \in A(x_1)$ and players $1, 2, \ldots, m$ receive the corresponding rewards $f^1(x_1, a_1), f^2(x_1, a_1), \ldots, f^m(x_1, a_1)$. Then the game passes to a state $y = x_2 \in X$ according to a probability distribution $\{p^{a_1}_{x_1,y}\}$ and so on indefinitely. Such a play of the game produces a sequence of states and actions $x_0, a_0, x_1, a_1, \ldots, x_t, a_t, \ldots$ that defines a stream of stage rewards $f^1(x_t, a_t), f^2(x_t, a_t), \ldots, f^m(x_t, a_t), \quad t = 0, 1, 2, \ldots$. *The average stochastic positional game* is the game with payoffs of the players

$$\omega^i_{x_0} = \lim_{t \to \infty} \inf \ \mathsf{E}\left(\frac{1}{t} \sum_{\tau=0}^{t-1} f^i(x_\tau, a_\tau)\right), \quad i = 1, 2, \ldots, m$$

where E is the expectation operator with respect to the probability measure in the Markov process induced by actions chosen by players in their position set and fixed starting state x_0. Each player in this game has the aim to maximize his average reward per transition.

An m-player discounted stochastic positional game is determined by the same elements mentioned above and a discount factor γ $(0 < \gamma < 1)$. In such a game each player chooses actions in his position set in order to maximize his expected discounted sum of stage rewards. So, *the discounted stochastic positional game* is the game with payoffs of the players

$$\sigma_{x_0}^i = \mathsf{E}\left(\sum_{\tau=0}^{\infty} \gamma^\tau f^i(x_\tau, a_\tau)\right), \quad i = 1, 2, \ldots, m.$$

In the case $p_{x,y}^a \in \{0, 1\}, \forall a \in A(x), \ \forall x, y \in X$ the considered stochastic positional games become deterministic positional games with mean and discounted payoffs from [1, 2, 5].

We study the problem of the existence Nash equilibria for the considered games when players use pure and mixed stationary strategies of a selection the action in the states.

2 Stochastic Positional Games in Stationary Strategies

A strategy of player $i \in \{1, 2, \ldots, m\}$ in a stochastic positional game is a mapping s^i that provides for every state $x_t \in X_i$ a probability distribution over the set of actions $A(x_t)$. 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 $x_t = x \in X_i$ (i. e. s^i do not depend on t), then s^i is called *a stationary strategy.* Thus, we can identify the set of mixed stationary strategies $\mathbf{S}^i$ of player i with the set of solutions of the system

$$\begin{cases} \sum\limits_{a \in A(x)} s_{x,a}^i = 1, & \forall x \in X_i; \\ s_{x,a}^i \geq 0, & \forall x \in X_i, \ \ \forall a \in A(x) \end{cases}$$

where $s_{x,a}^i$ expresses the probability that player i chooses action $a \in A(x)$ in $x \in X_i$. Each basic solution of this system corresponds to a pure stationary strategy.

Let $\mathbf{s} = (s^1, s^2, \ldots, s^m)$ be a profile of stationary strategies (pure or mixed strategies) of the players. Then we can find the probability transition matrix $P^{\mathbf{s}} = (p_{x,y}^{\mathbf{s}})$ of the Markov process induced by $\mathbf{s}$ as follows

$$p_{x,y}^{\mathbf{s}} = \sum_{a \in A(x)} s_{x,a}^i p_{x,y}^a \quad \text{for} \quad x \in X_i, \ \ i = 1, 2, \ldots, m. \tag{1}$$

Therefore if $Q^{\mathbf{s}} = (q_{x,y}^{\mathbf{s}})$ is the limiting probability matrix of $P^{\mathbf{s}}$ then the average payoffs per transition $\omega_{x_0}^1(\mathbf{s}), \ \omega_{x_0}^2(\mathbf{s}), \ldots, \omega_{x_0}^m(\mathbf{s})$ for the players are determined as follows

$$\omega_{x_0}^i(\mathbf{s}) = \sum_{k=1}^{m} \sum_{y \in X_k} q_{x_0,y}^{\mathbf{s}} f^i(y, s^k), \quad i = 1, 2, \ldots, m, \tag{2}$$

where

$$f^i(y, s^k) = \sum_{a \in A(y)} s_{y,a}^k f^i(y, a), \ \ \text{for} \ \ y \in X_k, \ k \in \{1, 2, \ldots, m\} \tag{3}$$

expresses the step reward of player i in the state $y \in X_k$ when player k uses the strategy s^k.

The functions $\omega_{x_0}^1(\mathbf{s}), \ \omega_{x_0}^2(\mathbf{s}), \ \ldots, \ \omega_{x_0}^m(\mathbf{s})$, defined according to (2), (3) on $\mathbf{S} = \mathbf{S}^1 \times \mathbf{S}^2 \times \cdots \times \mathbf{S}^m$, determine a game in normal form that we denote by $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}}, \{\omega_{x_0}^i(\mathbf{s})\}_{i=\overline{1,m}} \rangle$. This game corresponds to the *average stochastic positional game in stationary strategies.* In general,

we can consider a normal form of average stochastic positional game in stationary strategies when the starting state is chosen randomly according to a given distribution $\{\theta_x\}$ on X. In this case we obtain a game in normal form $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}},\ \{\omega^i_\theta(\mathbf{s})\}_{i=\overline{1,m}}\ \rangle$, where

$$\omega^i_\theta(\mathbf{s}) = \sum_{x\in X} \theta_x \omega^i_x(\mathbf{s}), \quad i = 1, 2, \ldots, m.$$

If $\theta_x = 0, \forall x \in X \setminus \{x_0\}$ and $\theta_{x_0} = 1$ the considered game becomes a stochastic positional game with fixed starting state x_0.

The normal form of a *discounted stochastic positional game in stationary strategies* we denote by $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}}, \{\sigma^i_{x_0}(\mathbf{s})\}_{i=\overline{1,m}}\ \rangle$, where $\sigma^i_{x_0}(\mathbf{s}),\ i = 1, 2, \ldots, m$ for a given $\mathbf{s} = (s^1, s^2, \ldots, s^m) \in \mathbf{S}$ is defined as follows. We consider the matrix $W^{\mathbf{s}} = (w^s_{x,y})$, where $W^{\mathbf{s}} = (I - \gamma P^{\mathbf{s}})^{-1}$ and $P^{\mathbf{s}} = (p^{\mathbf{s}}_{x,y})$ is the probability transition matrix determined according to (1). Then

$$\sigma^i_{x_0}(\mathbf{s}) = \sum_{k=1}^{m} \sum_{y\in X_k} w^{\mathbf{s}}_{x_0,y} f^i(y, s^k), \quad i = 1, 2, \ldots, m.$$

In the case when the starting state is chosen randomly according to a given distribution $\{\theta_x\}$ on X we obtain a normal form of a discounted stochastic positional game in stationary strategies $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}}, \{\sigma^i_\theta(\mathbf{s})\}_{i=\overline{1,m}}\ \rangle$, where

$$\sigma^i_\theta(\mathbf{s}) = \sum_{x\in X} \theta_x \sigma^i_x(\mathbf{s}), \quad i = 1, 2, \ldots, m.$$

3 The Main Results

For an average stochastic positional game in stationary strategies $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}}, \{\omega^i_\theta(\mathbf{s})\}_{i=\overline{1,m}}\ \rangle$ we obtained the explicit representation of payoff functions $\omega^i_{x_0}(\mathbf{s}),\ i = 1, 2, \ldots, m$. We have shown that these payoff functions can be represented as follows

$$\omega^i_\theta(s^1, s^2, \ldots, s^m) = \sum_{k=1}^{m} \sum_{x\in X_k} \sum_{a\in A(x)} s^k_{x,a} f^i(x, a) q_x, \qquad i = 1, 2, \ldots, m$$

where q_x for $x \in X$ are determined uniquely from the following system of linear equations

$$\begin{cases} q_y - \sum\limits_{k=1}^{m} \sum\limits_{x\in X_k} \sum\limits_{a\in A(x)} s^k_{x,a}\ p^a_{x,y}\ q_x = 0, & \forall y \in X; \\ q_y + w_y - \sum\limits_{k=1}^{m} \sum\limits_{x\in X_k} \sum\limits_{a\in A(x)} s^k_{x,a}\ p^a_{x,y}\ w_x = \theta_y, & \forall y \in X \end{cases}$$

and each $\omega^i_\theta(s^1, s^2, \ldots, s^m)$ is quasi-monotonic (quasi-convex and quasi-concave) with respect to strategy s^i on $\mathbf{S}^i$ and graph continuous in the sense of Dasgupta and Maskin [3]. Based on these properties we proved the existence of mixed stationary Nash equilibria for average stochastic positional games. Additionally we have shown that an arbitrary two player zero-sum average stochastic positional game possesses a pure stationary Nash equilibrium.

For a discounted stochastic positional game in stationary strategies $\langle \{\mathbf{S}^i\}_{i=\overline{1,m}}, \{\sigma^i_\theta(\mathbf{s})\}_{i=\overline{1,m}}\ \rangle$ we obtained the explicit representation of payoff functions $\sigma^i_{x_0}(\mathbf{s}),\ i = 1, 2, \ldots, m$. We have shown

that these payoff function can be represented as follows

$$\sigma_\theta^i(s^1, s^2, \ldots, s^m) = \sum_{k=1}^{m} \sum_{x \in X_k} \sum_{a \in A(x)} s_{x,a}^k f^i(x,a) q_x, \qquad i = 1, 2, \ldots, m,$$

where q_x for $x \in X$ are determined uniquely from the following system of linear equations

$$q_y - \gamma \sum_{k=1}^{m} \sum_{x \in X_k} \sum_{a \in A(x)} s_{x,a}^k \, p_{x,y}^a \, q_x = \theta_y, \qquad \forall y \in X$$

and each $\omega_\theta^i(s^1, s^2, \ldots, s^m)$ is quasi-monotonic (quasi-convex and quasi-concave) with respect to strategy s^i on $\mathbf{S}^i$ and continuous on $\mathbf{S}$. Based on these properties and the results from [6] we proved the existence of pure stationary Nash equilibria for discounted stochastic positional games.

4 Conclusion

Stochastic positional games with finite state and action spaces generalize deterministic positional games on graphs from [1, 2, 5]. For a nonzero average stochastic positional game a Nash equilibrium in pure stationary strategies may not exist, however for an arbitrary average stochastic positional game a Nash equilibrium in mixed stationary strategies always exists. In the case of two player zero-sum average stochastic positional games there exists a Nash equilibrium in pure stationary strategies. For a discounted stochastic positional game, there always exists a Nash equilibrium in pure stationary strategies.

References

[1] Alpern, S. *Cycles in extensive form perfect information games.* J. Math. Anal. Appl. **159**, 1-17, 1991.

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

[3] Dasgupta, P. and Maskin, E. *The existence of equilibrium in discontinuous economic games.* Rev. Econ. Stud. **53**, 1-26, 1986.

[4] Debreu, G. *A social equilibrium existence theorem.* Proceedings of the National Academy of Sciences, **38**, 886-893.

[5] Gurvich, V. Karzanov, A. and Khachiyan, L. *Cyclic games and an algorithm to find minimax cycle means in directed graphs.* USSR Comput. Math. Math. Phis., **28**, 85–91, 1988.

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

The graceful chromatic number for some particular classes of graphs

Radu Mincu[1], Camelia Obreja[1], and Alexandru Popa[1,2]

[1]Department of Computer Science, University of Bucharest
[2]National Institute for Research and Development in Informatics

Abstract

Graph colorings are a major area of study in graph theory involving the constrained assignment of labels(colors) to vertices or edges. There are many types of colorings defined. The most common type of coloring is the proper vertex k-coloring which is defined as a vertex coloring from a set of k colors such that no two adjacent vertices share a common color.

Our central focus in this paper is a variant of the proper vertex k-coloring problem, termed *graceful coloring* introduced by Gary Chartrand in 2015 and defined as follows. A *graceful k-coloring* of a nonempty graph G is a proper vertex coloring $c : V(G) \to [k]$, where $k \geq 2$, that induces a proper edge coloring $c' : E(G) \to [k-1]$ defined by $c'(uv) = |c(u) - c(v)|$.

In this work we find the *graceful chromatic number* for some well-known classes of graphs such as Friendship graph, Petersen graph, Cactus graph and others.

1 Introduction and preliminaries

Graph colorings are a fundamental topic in graph theory and originate from the Four Color Problem of Francis Guthrie from 1852. Since then, researchers study intensively the topic. The most known type of coloring is the proper vertex k-coloring in which the goal is to color the vertices of an undirected graph with k colors such that any two adjacent vertices are colored with distinct colors. The proper vertex k-coloring is motivated by applications such as pattern matching, scheduling, designing seating plans, exam timetabling and solving Sudoku puzzles. Besides this classical graph coloring, many other types of colorings were introduced: harmonious, graceful, set, multiset, metric, sigma, modular [7] etc.

In this paper we focus on another variant of the proper k-coloring problem, termed *the graceful k-coloring*, introduced by Chartrand and defined as follows.

Definition 1 (Graceful coloring)**.** *A graceful k-coloring of a nonempty graph G is a proper vertex coloring $c : V(G) \to [k]$, where $k \geq 2$, that induces a proper edge coloring $c' : E(G) \to [k-1]$ defined by $c'(uv) = |c(u) - c(v)|$. A vertex coloring c of a graph G is a graceful coloring if c is a graceful k-coloring for some $k \in N$.*

Next, we present the *graceful chromatic number*, which is immediately related to the concept of graceful coloring.

Definition 2 (Graceful chromatic number)**.** *The graceful chromatic number of a graph G, denoted by $\chi_g(G)$, is the minimum k for which G has a graceful k-coloring.*

Bi, Byers, English, Laforge and Zhang [1, 2, 4] study the concept of graceful coloring. We emphasize that the concept of graceful coloring is different than the related graceful labeling that was introduced by Rosa in 1967 [6]. We refer the reader to a comprehensive survey in [5].

Next, we present a summary of the known results on the graceful chromatic number. First, notice that there are immediate lower and upper bounds for the graceful chromatic number of a graph.

Observation 1. *([1]) Let $\chi(G)$ be the chromatic number of G and $grac(G)$ be the gracefulness of G, i.e., the smallest positive integer k for which is possible to label the vertices of G with distinct elements of the set $\{0, 1, 2, \ldots, k\}$ in such way that an edge is labeled $\{1, 2, \ldots, m\}$ and distinct edges receive distinct labels. Then:*

$$\chi(G) \leq \chi_g(G) \leq grac(G) \leq 2^{n-1}$$

From Definition 1 an important observation follows:

Observation 2. *([1]) If c is a coloring of a nontrivial connected graph, then c is a graceful coloring in G if and only if:*

- for each vertex v of G the vertices in the closed neighborhood of v are assigned distinct colors by c and

- for each path (x,y,z) of order 3 in G, $c(y) \neq (c(x) + c(z))/2$.

Remark 3. *([7]) As a consequence of first condition in previous observation, it follows that if G is a nontrivial connected graph, then $\chi_g(G) \geq \Delta(G) + 1$, where $\Delta(G)$ is the maximum degree in G.*

Recall that the *distance* $d(u, v)$ between two vertices is the shortest $u - v$ path in G, and the *diameter* $diam(G)$ is the largest distance between any two vertices of G. From Observation 2 it follows:

Corollary 4. *([1]) For a connected graph G of order $n \geq 3$ and diameter at most 2: $\chi_g(G) \geq n$.*

We present a lower bound for the graceful chromatic number of a connected graph.

Corollary 5. *([4]) If G is a connected graph with minimum degree $\delta \geq 2$, then $\chi_g(G) \leq \lceil 5\delta/3 \rceil$.*

The graceful chromatic number was studied on the following classes of graphs: trees, cycles (for a cycle C_n with $n \geq 4$ vertices: $\chi_g(C_n) = 4$ if $n \neq 5$, else $\chi_g(C_n) = 5$) [1], (regular) complete bipartite graph $G = (V, E)$ of order $n \geq 3$ ($\chi_g(G) = n$) [1], Thomsen graph $K_{3,3}$ or the *utility graph* ($\chi_g(K_{3,3}) = n$), complete tripartite graphs (for each integer $p \leq 2$, $\chi_g(K_{p,p,p}) \leq 4p + 1$ if p is even and $\chi_g(K_{p,p,p}) \leq 4p$ if p is odd) [7] and some well known graphs as wheel of order $n \geq 6$ ($\chi_g(W_n) = n$) [1], stars $K_{1,n-1}, n \geq 3$ ($\chi_g(K_{1,n-1}) = n$) [7], paths ($\chi_g(P_n) = 4$ for $n \geq 5$) [1], caterpillar (caterpillar T with maximum degree $\Delta \geq 2$: $\Delta + 1 \leq \chi_g(T) \leq \Delta + 2$) [1], cubical graph Y_4 ($\chi_g(Y_4) = 5$) [7].

According to [7], not all graphs of diameter 2 have the graceful chromatic number equal with their order:

Proposition 6. *([7]) There exist infinite classes of connected graphs G of order n such that $diam(G) = 2$ and $\chi_g(G) > n$.*

2 Our results

In this section we present new results for the graceful coloring for several individual graphs and for several classes of graphs. We first consider some graphs with diameter 2 and $\chi_g(G) = n$, followed by some graphs with diameter 2 and $\chi_g(G) \neq n$ (in fact $\chi_g(G) > n$), and finally some

graphs with *diameter* > 2. Many of considered graphs are highly symmetrical: strongly regular (Petersen graph), symmetric (like Heawood graph, Mobius-Kantor graph or Desarques graph) or semi-symmetric. Others are families of graphs, like platonic solids (octahedron, dodecahedron, icosahedron), Friendship graphs, truncated solids or Snarks.

The definitions for the graph classes showcased below can be found in [3].

Graphs with diameter 2 and $\chi_g(G) = n$

We present the results concerning some graphs with diameter 2. In the previous work, the following diameter 2 graphs have been shown to have the graceful chromatic number of their order: the class of star graphs [7], wheel graphs with more than 5 vertices [1], the Utility graph [7].

We find that there are several others graphs with diameter 2 having graceful chromatic number equal with their order: individual graphs (e.g. Diamond graph, Prism graph, Wagner graph, Petersen graph) and family graphs (e.g. the Friendship graph F_n with $2n + 1(n \geq 2)$ vertices, the Fan graph $F_{m,n-m}$ where $n \geq 5$).

Graphs with diameter 2 and $\chi_g(G) > n$

There are some graphs of diameter 2 which have their graceful chromatic number greater than their order, such as: Moser spindle graph ($n = 7$ and $\chi_g(G) = 8$), House graph ($n = 5$ and $\chi_g(G) = 6$).

Graphs with diameter greater than 2

Next we present our results regarding the chromatic number for some graphs with diameter> 2.

First, we enumerate our results on k-regular graphs (with *diameter* $\neq 2$). Durer graph ($n = 12$, $\chi_g(G) = 6$), Truncated tetrahedron graph (n=12, $\chi_g(G) = 5$), Heawood graph (n=14, $\chi_g(G) = 5$), Tutte eight-cage ($n = 30$, $\chi_g(G) = 6$), Dodecahedron (has 20 vertices and needs only 6 colors for a graceful coloring). Observe that for most of them $\chi_g(G) < n$.

Notice that there are some graphs which need n colors for a graceful coloring, e.g. Flower snark graphs family, Icosahedron graph.

We also have results regarding some irregular graphs: for Double star graphs $S_{n,m}$ with $n+m+2$ vertices the graceful chromatic number is $max(n, m)+1$; Jellyfish graph $J_{n,m}$ have $n+m+4$ vertices and $\chi_g(J_{n,m}) = max(n, m) + 3$; Crown C_n has $2n$ vertices, $\chi_g(C_n) = 5$ for $n \neq 5$ and $\chi_g(C_5) = 6$; Umbrella graph $U_{n,m}$, with $n + m$ vertices has $\chi_g(U_{n,m}) = n + 2$; Helm graph H_n has n vertices and $\chi_g(H_n) = n + 1$.

We additionally study other more restricted classes of graphs such as: French graph, Golomb graph, Goldner-Harrary graph, Hershal graph, Prism graph; Sun graph, Cactus, Web, Braid, Triangular book, Gear graph, Mongolian tent graph, Mongolian ger graph.

An example

Proposition 7. *For Jellyfish graph $J_{n,m}$ and $n >= m$ the graceful chromatic number is $\chi_g(J_{n,m}) = n + 3$.*

Proof. The Jellyfish graph $J_{n,m}$ is defined as follows. The graph $J_{n,m}$ has the following set of vertices: $\{u, v, u_1, v_1, a_1, a_2, \ldots a_n, b_1, b_2, \ldots b_m\}$. The edges are as follows: two cycles $\{u, v, v_1\}$ and $\{u, v, u_1\}$ which share an edge, (u, v), n pendants connected to u_1, $(a_1, u_1), (a_2, u_1), \ldots, (a_n, u_1)$, and m pendants connected to v_1, $(b_1, v_1), (b_2, v_1), \ldots, (b_m, v_1)$.

First, we show a graceful coloring with $n + 3$ colors if $n > m$. Then we show a graceful coloring with $n + 3$ colors for $n = m$.

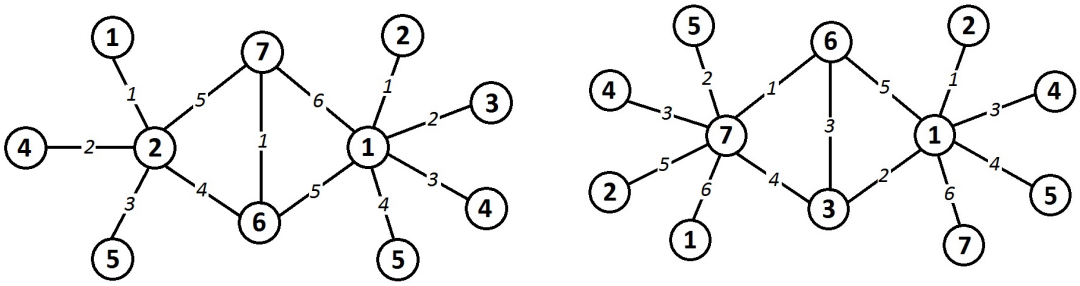


Figure 1: A graceful 7-coloring of $J_{3,4}$ and, respectively, a graceful 7-coloring of $J_{4,4}$.

We consider first the case $n > m$. The coloring $c : V \to \{1, 2, \ldots, n+3\}$ is as follows: $c(u_1) = 1$, $c(a_1) = 2, \ldots, c(a_n) = n + 1$,$c(u) = n + 2$ and $c(v) = n + 3$. Therefore the induced coloring c' for edges is as follows: $c'(u_1, a_1) = 1$, $c'(u_1, a_2) = 2, \ldots,$ $c'(u_1, a_n) = n$, $c'(u_1, u) = n + 1$, $c'(u_1, v) = n + 2$, and $c'(u, v) = 1$. Thus, there are only one color which appear twice in the edges coloring set, 1, and the edges with this color are not adjacent each other.

For vertex v_1 the color $c(v_1)$ can be in set $\{1, 2, \ldots, n + 3\} \setminus \{1, n + 2, n + 3\}$. Let be $c(v_1) = 2$. Thus $c'(v_1, u) = n$ and $c'(v_1, v) = n + 1$. The vertices b_i, $i \in [1, m]$, which are pendant to v_1, can have $c(b_i)$ in the set of colors $\{1, 4, \ldots, n + 1\}$, which has $n - 1$ elements. Since $n > m$, this is an appropriate graceful coloring for Jellyfish graph.

Next, we show a graceful coloring with $n + 3$ colors for the Jellyfish graph $J_{n,n}$. To color vertices $u_1, a_1, \ldots, a_n, u, v$ (which are forming a star) we need $n + 3$ colors. Same for vertices $v_1, b_1, \ldots, b_n, u, v$. Since neither vertices u_1, v_1, nor vertices u, v cannot have the same color, we want to find out which colors can be assign to vertices u_1, v_1, u and v in order to get a graceful coloring of $J_{n,n}$ graph. Let $c(u_1) = 1$ and $c(v_1) = n + 3$ be the colors for the vertices u_1, v_1 respectively. Thus there are two colors for vertices u, v in set of $\{2, \ldots, n + 2\}$ such that we have a $(n + 3)$-graceful coloring. □

In Figure 1 we present a graceful $7-coloring$ for Jellyfish $J_{3,4}$ and, respectively, an $7-coloring$ for Jellyfish graph with $n = m = 4$.

References

[1] Z. Bi, A. Byers, S. English, E. Laforge, and P. Zhang. Graceful colorings of graphs. *J. Combin. Math. Combin. Comput.*, 2018.

[2] Z. Bi, A. Byers, and P. Zhang. Revisiting graceful labelings of graphs. *J. Combin. Math. Combin. Comput.*, 2018.

[3] H. de Ridder. Information system on graph classes and their inclusions. `http://www.graphclasses.org`, Dec. 2018.

[4] S. English and P. Zhang. On graceful colorings of tree. *Mathematica Bohemica*, pages 57–73, 2017.

[5] J. Gallian. A dynamic survey of graph labeling. *Electron J Combin DS6*, 19, 11 2000.

[6] A. Rosa. On certain valuations of the vertices of a graph. In Gordon and Breach, editors, *Theory of Graphs*, pages 349–355, 1967.

[7] P. Zhang. *A Kaleidoscopic View of Graph Colorings*. SpringerBriefs in Mathematics. Springer International Publishing, 2016.

On Uniquely Colourable Graphs

Samuel Mohr*[1]

[1]Institut für Mathematik der Technischen Universität Ilmenau, Weimarer Straße 25, 98693 Ilmenau, Germany

Abstract

A uniquely k-colourable graph is a graph with exactly one partition of the vertex set into k colour classes. Here, we investigate some constructions of uniquely k-colourable graphs and give a construction of K_k-free uniquely k-colourable graphs with equal colour class sizes.

We use standard terminology from graph theory and consider *simple*, *finite* graphs G with vertex set $V(G)$ and edge set $E(G)$. A *k-colouring* of a graph G with $k \in \mathbb{N}$ is a partition $\mathcal{C}$ of the vertex set $V(G)$ into k non-empty sets $A_1, \dots, A_k$. The colouring $\mathcal{C}$ is called *proper* if each set is an independent set of G, that means that there are no two adjacent vertices of G in the same colour class $A \in \mathcal{C}$. The *chromatic number* $\chi(G)$ is the minimum k such that there is a proper k-colouring of G.

We call a graph G *uniquely k-colourable* if $\chi(G) = k$ and for any two proper k-colourings $\mathcal{C}$ and $\mathcal{C}'$ of G, we have $\mathcal{C} = \mathcal{C}'$. It is easy to see that the complete graph K_k on k vertices is uniquely k-colourable and we can obtain a family of uniquely k-colourable graphs by consecutively adding a vertex and join it to all vertices except those of one colour class. This raises the question if all uniquely k-colourable graphs contain K_k as a subgraph.

The properties of uniquely colourable graphs have been widely studied, for example in [3, 5, 8, 1, 7, 2, 4]. One such property—can be found in [3]—is that the union of any two distinct colour classes induces a connected graph. Assume to the contrary that there is a graph G with unique colouring $\mathcal{C}$ and there are $A, B \in \mathcal{C}, A \neq B$ such that $G[A \cup B]$ has at least two components. Let H be such a component and consider the colouring $\tilde{\mathcal{C}}$ with $\tilde{\mathcal{C}} = (\mathcal{C} \setminus \{A, B\}) \cup \{(A \setminus V(H)) \cup (B \cap V(H))\} \cup \{(B \setminus V(H)) \cup (A \cap V(H))\}$. Then $\tilde{\mathcal{C}}$ is a proper colouring distinct from $\mathcal{C}$, a contradiction. We say $\tilde{\mathcal{C}}$ is obtained from $\mathcal{C}$ by a *Kempe change along H*.

This implies that in a uniquely k-colourable graphs every vertex has a neighbour in every other colour class. Hence, it is connected and has minimum degree at least $k-1$. Furthermore, a uniquely k-colourable graphs is $(k-1)$-connected. To see this, assume that there is a non-complete graph G with a unique k-colouring $\mathcal{C}$ and for two non-adjacent vertices x, y, there is a separator S with $|S| \leq k-2$. But then there are distinct $A, B \in \mathcal{C}$ with $A \cap S = \emptyset = B \cap S$ and $(G - S)[A \cup B]$ is connected. Since x and y have neighbours in $A \cup B$, they cannot be separated by S, a contradiction.

This question whether a uniquely k-colourable graphs always contains K_k as a subgraph was first disproved by Harary, Hedetniemi, and Robinson [5]. They gave a uniquely 3-colourable graph F without triangles. For larger $k \geq 4$ a uniquely k-colourable graph is $F + K_{k-3}$, where $G_1 + G_2$ is the complete join of the two graphs G_1 and G_2.

*Gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – 327533333.

Several years later, Xu [8] proved that the number of edges of a uniquely k-colourable graph on n vertices is at least $(k-1)\,n - \binom{k}{2}$ and that this is best possible. He further conjectured that uniquely k-colourable graphs with exactly this number of edges have K_k as a subgraph [8]. This conjecture was disproved by Akbari, Mirrokni, and Sadjad [1]. They constructed a K_3-free uniquely 3-colourable graph G on 24 vertices and 45 edges. For the cases of $k \geq 4$, again $G + K_{k-3}$ disproves the conjecture.

We are interested in constructions of uniquely k-colourable graphs such that the colour classes have "nearly the same size". One useful concept for this is the critical chromatic number introduced by Komlós [6] in the context of bounds on a Tiling Turán number. Given a k-colourable graph H on h vertices, let $\sigma(H)$ be the smallest possible size of a colour class in any proper k-colouring of H. Then the *critical chromatic number* is defined by

$$\chi_{cr}(H) = (\chi(H) - 1) \cdot \frac{h}{h - \sigma(H)}.$$

The critical chromatic number fulfils $\chi(H) - 1 < \chi_{cr}(H) \leq \chi(H)$ and equality holds if and only if in every k-colouring of H the colour classes have the same size.

All constructions above have critical chromatic number close to $\chi(G) - 1 = k - 1$.

In the following, we give a new construction of uniquely k-colourable graphs, see Theorem 1. Given a uniquely k-colourable graph H without K_k and $\chi_{cr}(H) = \chi(H)$, this construction leads to a uniquely $(k+1)$-colourable graph G without K_{k+1} and $\chi_{cr}(G) = \chi(G)$. We further distinguish this construction from a result of Nešetřil [7] and probabilistic proofs of the existence of uniquely colourable graphs, for example, in [2].

Theorem 1. *Let H be a uniquely k-colourable graph on n vertices. Then there is a graph $G = \nu(H)$ with the following properties:*

1. *G is uniquely $(k+1)$-colourable,*
2. *$\omega(G) = \omega(H) + 1$,*
3. *$\chi_{cr}(G) = \chi(G)$ if $\chi_{cr}(H) = \chi(H)$,*
4. *$|E(G)| = (3k+1)\,|E(H)| + (k-1)\,n$ and $|V(G)| = (k+1)\,n$,*
5. *The minimum degree of G is $2\,\delta(H) + 1$,*
6. *H is an induced subgraph of G.*

Furthermore, we give uniquely 3-colourable graphs with critical chromatic number 3 and, in particular, find the smallest one among these. Using these graphs, the construction by Theorem 1 leads to uniquely k-colourable graphs G without K_k such that $\chi_{cr}(G) = \chi(G) = k$.

References

[1] Saeed Akbari, VS Mirrokni, and BS Sadjad. K_r-free uniquely vertex colorable graphs with minimum possible edges. *Journal of Combinatorial Theory, Series B*, 82(2):316–318, 2001.

[2] Béla Bollobás and Norbert Sauer. Uniquely colourable graphs with large girth. *Canadian Journal of Mathematics*, 28(6):1340–1344, 1976.

[3] Gary Chartrand and Dennis P Geller. On uniquely colorable planar graphs. *Journal of Combinatorial Theory*, 6(3):271–278, 1969.

[4] Thomas Emden-Weinert, Stefan Hougardy, and Bernd Kreuter. Uniquely colourable graphs and the hardness of colouring graphs of large girth. *Combinatorics, Probability and Computing*, 7(4):375–386, 1998.

[5] Frank Harary, ST Hedetniemi, and RW Robinson. Uniquely colorable graphs. *Journal of Combinatorial Theory*, 6(3):264–270, 1969.

[6] János Komlós. Tiling turán theorems. *Combinatorica*, 20(2):203–218, 2000.

[7] Jaroslav Nešetřil. On uniquely colorable graphs without short cycles. *Časopis pro pěstování matematiky*, 98(2):122–125, 1973.

[8] Xu Shaoji. The size of uniquely colorable graphs. *Journal of Combinatorial Theory, Series B*, 50(2):319–320, 1990.

Optimally rescheduling jobs under LIFO constraints

Gaia Nicosia[1], Andrea Pacifici[2], Ulrich Pferschy[3], Edoardo Polimeno[4], and Giovanni Righini*[4]

[1]University Roma Tre, Italy
[2]University Roma Tor Vergata, Italy
[3]University of Graz, Austria
[4]University of Milan, Italy

We consider the problem of optimally rescheduling jobs from a given input sequence to a new one, with the constraint that (a) jobs can only be postponed but not preponed and (b) when jobs are picked up from the incoming sequence, they are put in a stack of given finite capacity before being reinserted in their new positions. Therefore if job i precedes job j in the input sequence and both jobs are extracted and reinserted, then job j must precede job i in the new sequence. We denote this by Last-In-First-Out (LIFO) constraint. We consider three objective functions to be optimized by the new sequence. The problem arises from industrial production processes optimization, where production lots must traverse several working phases and it may be convenient to reorganize their sequence, owing to - for instance - different processing times in each phase. The resulting combinatorial optimization problem is a variation of that studied by Nicosia et al. [1], where the job rearrangement is subject to the "postpone only" constraint, but not to the Last-In-First-Out constraint.

An ILP model.

We are given an ordered set N of n jobs, the processing time p_i of each job $i \in N$, the due date δ_i of each job $i \in N$ and the maximum size S of the stack.

We use binary variables $x_{ij} \in \{0,1\}$ $\forall i < j \in N$ to represent the actions of a robot that can extract jobs from the sequence and reinsert them in a successive position: each x_{ij} variable set to 1 indicates that job i is extracted and it is reinserted immediately after the subsequence that initially terminates with job j.

LIFO constraints have the form $x_{ij} + x_{kh} \leq 1 \quad \forall i \leq k \leq j < h \in N$. As a consequence, only *nested moves* ($i < k < h \leq j$) and *disjoint moves* ($i < j < k < h$) are allowed.

Stack size constraints have the form $\sum_{i,j \in N: i \leq p, j > p} x_{ij} \leq S \quad \forall p = 1, \ldots, n-1$. They impose that no more than S moves can be nested in one another.

We consider three objective functions: minimization of the total completion time, minimization of the maximum lateness, minimization of the number of late jobs.

For each job $i \in N$ we define its completion time c_i in the initial sequence as $c_i = \sum_{k=1}^{i} p_k$ and its completion time c'_i in the final sequence as $c'_i = c_i - \sum_{k=1}^{i-1} \sum_{h=i}^{n} p_k x_{kh} + \sum_{k=i+1}^{n} (x_{ik} \sum_{h=i+1}^{k} p_h)$. The term $\sum_{k=1}^{i-1} \sum_{h=i}^{n} p_k x_{kh}$ indicates the advance of job i when some jobs preceding it in N are moved after it; the term $\sum_{k=i+1}^{n} (x_{ik} \sum_{h=i+1}^{k} p_h)$ indicates the delay of job i when it is postponed. All completion times can be trivially computed in $O(n)$ time with the recursion $c_1 = p_1$ and

*The fifth author acknowledges the support of Regione Lombardia, grant agreement n. E97F17000000009, Project AD-COM

$c_i = c_{i-1} + p_i \ \forall i = 2, \ldots, n$. Furthermore, it is possible to compute the duration $d(i,j)$ of each subsequence $[i, \ldots, j]$ in $O(n^2)$ time, as follows: $d(i,i) = p_i \ \forall i = 1, \ldots, n$ and $d(i,j) = d(i, j-1) + p_j \ \forall i = 1, \ldots, n-1 \ \forall j = i+1, \ldots, n$. We define the lateness t_i of each job $i \in N$ as the delay with respect to its due date: $t_i = c_i - \delta_i \ \forall i \in N$. The lateness t'_i in the final sequence is $t'_i = c'_i - \delta_i \ \forall i \in N$.

Owing to the space limit, here we describe only two of the three exact optimization algorithms we have devised for the three objective functions listed above.

Total completion time minimization.

We consider the objective function z_1: minimization of the total completion time.

Moves. Consider a subsequence $[i, \ldots, j]$ in N with $i < j$; the move (i,j) consists of deleting job $i \in N$ and reinserting it just after all jobs of the subsequence. We define its cost $m(i,j)$ as $m(i,j) = \sum_{k=i+1}^{j} p_k - (j-i)p_i$. The term $\sum_{k=i+1}^{j} p_k$ indicates the delay of job i; the term $(j-i)p_i$ indicates the total advance of the other jobs in the subsequence, i.e. those in $[i+1, \ldots, j]$. The value of $m(i,j)$ indicates the variation of z_1 as a consequence of the move (i,j).

Remark 1. The completion times of the jobs preceding i and following j are not affected by the move (i,j).

Remark 2. The value of $m(i,j)$ does not depend on the order of the jobs in the subsequence $[i+1, \ldots, j]$.

Sequential moves. One of the two modes in which moves can be combined in a feasible solution is in a sequence, i.e. one move is completed before the next one begins. Necessary and sufficient condition for two sequential moves (i,j) and (k,h) to be compatible is that $i < j < k < h$ (or viceversa $k < h < i < j$). The effect on z_1 of two sequential moves is equal to the sum of the two effects computed separately, because the two subsequences are disjoint (see Remark 1).

Therefore, after pre-computing all costs $m(i,j)$ for all (i,j) pairs, it is possible to find the optimal sequence of moves by solving an instance of the shortest path problem on an acyclic digraph. The digraph has a node for each job in N and an additional dummy node numbered $n+1$. The arc set contains all arcs $(i, i+1)$ for each $i \in N$; they have zero cost and they correspond to not moving job i. The arc set also contains all arcs $(i, j+1)$ for all pairs $(i,j) \in N \times N : j > i$; they have cost equal to $m(i,j)$ and they correspond to moves (i,j). The cost $w^*(i,j)$ of the shortest path for each pair $(i,j) \in N \times (N \cup \{n+1\}) : j > i$ can be easily computed in $O(n^2)$ by dynamic programming.

Nested moves. The second way in which two moves can be combined is when they are nested. Necessary and sufficient condition for two nested moves (i,j) and (k,h) to be compatible is that $i < k < h \leq j$ (or viceversa $k < i < j \leq h$). We define *levels* of nested moves: a move that does not contain moves nested in it is a move at level 1; a move containing moves at level up to $l-1$ is a move at level l. Hence, in the remainder moves are no longer indicated by a pair (i,j) but by a triple (i,j,l) to indicate their level too.

The effect on z_1 of two nested moves is equal to the sum of the two separate effects, because of Remark 2: the cost $m(i,j,l)$ does not depend on the order of the subsequence $[i+1, \ldots, j]$, which is the only subsequence that can be affected by moves nested in move (i,j,l).

Therefore, after precomputing all optimal costs $w^*(i,j)$ described above, obtainable from sequences of moves at level $l-1$ for each subsequence $[i, \ldots, j]$ of N, it is possible to compute the

optimal cost $m(i,j,l)$ of each move (i,j,l), i.e. a move that can contain (sequences of) nested moves. For $l=1$, $m(i,j,1)$ corresponds to $m(i,j)$ as previously defined. For $l>1$, the cost of a move (i,j,l), beyond the effects already described for the computation of $m(i,j)$, must also take into account the possibility of optimally rescheduling the subsequence $[i+1,\ldots,j]$ with nested moves at lower levels. Therefore we have:

$$\begin{cases} m(i,j,1) = m(i,j)\ \forall (i,j) \in N \times (N \cup \{n+1\}) : j > i \\ m(i,j,l) = m(i,j) + w^*(i+1,j,l-1)\ \forall (i,j) \in N \times (N \cup \{n+1\}) : j > i,\ \forall l = 2,\ldots,S. \end{cases}$$

This definition of $m(i,j,l)$ allows to define the costs of the arcs of the digraph for each level l, once we have computed the costs w^* at level $l-1$. In turn, this allows to compute the optimal costs w^* at level l.

The optimal solution is found by computing $w^*(1,n+1,S)$.

Complexity. The initialization requires $O(n^2)$ time. For each level $l=1,\ldots,S$ the algorithm must compute m and w^* for each pair of jobs (i,j). Each m-value for $l>1$ requires $O(1)$ time. Each single cost $w^*(i,j,l)$ for $l \geq 1$ would require $O(n^2)$ for each pair (i,j), but all costs $w^*(i,j,l)$ can be computed in $O(n^2)$ over all j for fixed indices i and l. Therefore the complexity for computing w^* at each level l is $O(n^3)$. Hence the overall complexity of the algorithm is $O(n^3 S)$, which is upper bounded by $O(n^4)$ in case there are no limits to the size of the stack.

Maximum lateness minimization.

Similar results are obtained for the second objective function, z_2: minimizing the maximum lateness.

In an initialization we compute the completion times c_i, the durations $d(i,j)$ and also the lateness of each job and each subsequence. Given the completion times, the lateness values can be computed in $O(n)$ time as $t_i = c_i - \delta_i$. After that, the values of the maximum lateness in each subsequence $t^*(i,j)$ can be computed in $O(n^2)$ time with the recursion $t^*(i,i) = t_i\ \forall i = 1,\ldots,n$ and $t^*(i,j) = \max\{t^*(i,j-1), t_j\}\ \forall i = 1,\ldots,n-1\ \forall j = i+1,\ldots,n$.

Moves. We indicate with t_i the lateness of job i before a move (i,j) and with t'_i the lateness of job i after the move. When a move (i,j) is done, the lateness of all jobs k with $i+1 \leq k \leq j$ decreases by p_i, while the lateness of job i increases by $d(i+1,j)$.

As a main difference with respect to the previous case, we now denote by $m(i,j)$ the optimal value that the objective function would have if it were evaluated on the subsequence $[i,\ldots,j]$, instead of the variation in the objective function due to the move (i,j). Hence we define

$$\begin{cases} m(i,i) = t_i \quad \forall i \in N \\ m(i,j) = \max\{t_i + d(i+1,j),\ t^*(i+1,j) - p_i\} \quad \forall (i,j) \in N \times N : i < j. \end{cases}$$

The term $t_i + d(i+1,j)$ indicates the new lateness of job i, i.e. t'_i; the term $t^*(i+1,j) - p_i$ indicates the new maximum lateness of the subsequence $[i+1,\ldots,j]$, i.e. the maximum value of t'_k for $i+1 \leq k \leq j$.

Sequential moves. Owing to the min-max objective function, the effect on z_2 of two sequential moves is not equal to the sum of the two separate effects. However the following property still holds.

Remark 3. The completion time and the lateness of the jobs preceding i and following j in N are not affected by a move (i, j).

Therefore, moves on disjoint subsequences can be evaluated independently. It follows that also in this case, after pre-computing all costs $m(i, j)$ for all job pairs, it is possible to compute the optimal sequence of moves by solving an instance of a shortest path problem on an acyclic digraph with a min-max objective function. The digraph has a node for each job and an additional node numbered $n + 1$. Each arc $(i, j + 1)$ with $i \in N, j \in N$ and $j > i$ has cost equal to $m(i, j)$. With this digraph, min-max shortest paths can be computed in $O(n^2)$ time for each pair of jobs $(i, j) \in N \times (N \cup \{n + 1\})$.

A min-max shortest path is a path that minimizes the maximum among the costs of its arcs. Given an origin node $i \in N$, the optimal label of each node $j = i + 1, \ldots, n + 1$ is computed as $label_i(j) = \min_{k=i,\ldots,j-1}\{\max\{label_i(k), m(k, j - 1)\}\}$, with $label_i(i) = 0$.

The optimal cost $label_i^*(j)$ of each (i, j) shortest path is the minimum value of the maximum lateness that can be achieved in the subsequence $[i, \ldots, j - 1]$ with a sequence of moves, i.e. $label_i^*(j) = t^*(i, j - 1)$.

Nested moves.

Remark 4. In general, a different order of the jobs in the subsequence $[i + 1, \ldots, j]$ implies a different value of $m(i, j, l)$.

For $l = 1$ the cost $m(i, j, 1)$ corresponds to $m(i, j)$ as already defined. For the next levels the algorithm must take into account the possibility of optimally rescheduling the jobs in the subsequence $[i + 1, \ldots, j]$ with nested moves of lower levels. After precomputing all optimal costs $t^*(i, j, l-1)$ for each subsequence $[i, \ldots, j]$ that can be obtained with sequential moves at level $l-1$, it is possible to evaluate the optimal cost of a move on each subsequence at level $l > 1$. Therefore we have:

$$\begin{cases} m(i, i, l) = t_i \quad \forall i \in N,\ \forall l = 1, \ldots, S \\ m(i, j, 1) = m(i, j) \quad \forall (i, j) \in N \times N : j > i \\ m(i, j, l) = \max\{t_i + d(i + 1, j), t^*(i + 1, j, l - 1) - p_i\} \quad \forall (i, j) \in N \times N : j > i,\ \forall l = 2, \ldots, S. \end{cases}$$

This definition of $m(i, j, l)$ allows to define the costs of the arcs of the digraph for each level l, once given the costs of the moves at level $l - 1$.

The optimal solution is found by computing $t^*(1, n, S)$, i.e. the optimal cost a min-max shortest path from 1 to $n + 1$ at level S.

Complexity. For each level $l = 1, \ldots, S$ the algorithm must compute m and t^* for each pair (i, j). Each computation of a value of m requires $O(1)$ time. Once fixed i and l, one can compute all values $t^*(i, j, l)$ for each j in $O(n^2)$ time. Therefore the algorithm has the same complexity as in the previous case, i.e. $O(n^3 S)$.

References

[1] A. Alfieri, G. Nicosia, A. Pacifici, U. Pferschy, *Constrained job rearrangements on a single machine*, in "New Trends in Emerging Complex Real Life Problems" eds. P. Daniele, L. Scrimali, pp. 33-41, AIRO Springer Series Vol. 1, 2018.

An exact algorithm for the maximum weight perfect matching problem with conflicts

Temel Öncan[1], M. Hakan Akyüz[1], and İ. Kuban Altınel[2]

[1]Endüstri Mühendisliği Bölümü, Galatasaray Üniversitesi, İstanbul, 34357, Türkiye
[2]Endüstri Mühendisliği Bölümü, Boğaziçi Üniversitesi, İstanbul, 34342, Türkiye

Abstract

In this work, we consider the maximum weight perfect matching problem with conflicts, which is known to be $\mathcal{NP}$-hard. We propose a tailor-made branch-and-bound algorithm with a non-dichotomized branching rule based on a maximum weight stable set relaxation of the problem. We have realized preliminary computational experiments on randomly generated test instances and compared the computational performance of the new algorithm with the one of a well-known commercial solver. Based on the obtained results we can say that it is promising.

Keywords: Maximum Weight Perfect Matching, Integer Programming, Conflicts

1 Introduction

Given a graph $G = (V(G), E(G))$ with the vertices $V(G)$, edges $E(G)$, nonnegative weights c_e and conflicting edge lists $N_C(e)$ for each edge $e \in E(G)$, the Maximum Weight Perfect Matching Problem with Conflicts (MWPMC) aims to determine a perfect matching with maximum weight such that no two edges in the conflicting edge set are in the optimal solution. To the best of our knowledge, the only work addressing the MWPMC is [1] where the authors introduced the problem and have also proved its $\mathcal{NP}$-hardness. A special case of the MWPMC, namely the maximum weight matching with conflicts in bipartite graph, is considered in [4]. The contribution of this work is to design a tailor-made exact solution procedure, i.e. a branch-and-bound (B&B) algorithm, for the MWPMC. We have realized preliminary computational experiments on randomly generated instances and compared the performance of the proposed B&B algorithm with a binary integer linear programming (BILP) formulation of the MWPMC solved on CPLEX solver. According to the obtained results, we can say that the B&B algorithm is very efficient.

2 Two Mathematical Programming Models

Given a graph G let us define $\delta_G(v)$ be the set of edges incident with vertex v and $d_G(v)$ denote the degree of vertex v. Note that $d_G(v) = |\delta_G(v)|$ holds where $|S|$ represents the cardinality of set S. Furthermore, let $N_C(e)$ be the set of edges conflicting with edge $e \in E(G)$. Clearly, for two edges e and f in $E(G)$, when $f \in N_C(e)$ holds then $e \in N_C(f)$ is satisfied, as well. We let binary decision variable x_e be equal to 1 if edge $e \in E(G)$ is in the perfect matching, 0 otherwise. Then a

BILP formulation of the MWPMC can be given as follows:

$$\max z = \sum_{e \in E(G)} c_e x_e \tag{1}$$

s.t.

$$\sum_{e \in \delta_G(v)} x_e = 1 \qquad v \in V(G) \tag{2}$$

$$x_e + x_f \leq 1 \qquad e \in E(G); f \in N_C(e) \tag{3}$$

$$x_e \in \{0,1\} \qquad e \in E(G). \tag{4}$$

Here, the objective function (1) maximizes the sum of the edge weights in the solution. Constraints (2) guarantee that each vertex is incident with exactly one edge in the solution. Constraints (3) stipulate that at most one of the conflicting edges can be in the solution. Finally, constraints (4) are for the binary restrictions on the decision variables.

Conflict relations can be represented using the conflict graph $C = (V(C), E(C))$ where each vertex in $V(C)$ corresponds to an edge in $E(G)$ and each conflicting edge pair in $E(G)$ defines an edge in $E(C)$. Besides, two edges in $E(G)$ which are incident to the same vertex in $V(G)$ are also denoted with an edge in $E(C)$, since they are in conflict as a natural consequence of the matching problem. Then, the second BILP formulation for the MWPMC is as follows.

$$\max z = \sum_{e \in V(C)} c_e x_e \tag{5}$$

$$\text{s.t.} \sum_{e \in V(C)} x_e = \frac{|V(G)|}{2} \tag{6}$$

$$x_e + x_f \leq 1 \qquad \{e, f\} \in E(C) \tag{7}$$

$$x_e \in \{0,1\} \qquad e \in V(C). \tag{8}$$

The objective function (5) tries to maximize total weight of the edges in the solution. Constraints (6) stipulate that the number of selected vertices in $V(C)$ i.e. edges in $E(G)$, constitutes a perfect matching. Constraints (7) are for the conflicting edge pairs in $E(G)$ and constraints (8) restrict the the decision variable to be binary. Observe that, the BILP problem given by (5)-(8) is equivalent to the solution of Maximum Weight Stable Set Problem (MWSS) with an additional restriction on the number of selected vertices in $V(C)$, namely edges in $E(G)$. Hence, we name this problem as the Maximum Weight Perfect Stable Set Problem (MWPSS), which is an extension of the well-known $\mathcal{NP}$-complete MWSS [2] defined on the conflict graph C.

3 Branch-and-bound algorithm

The outline of the proposed B&B algorithm is given with Algorithm 1. Although most of the steps are self-explanatory we introduce the following notation to better explain the steps of the algorithm. Let t be the search node index of the B&B tree and $\mathcal{L}$ be the set of active B&B nodes. Let $I^{(t)}$ be the set of edges that must be included into the solution at node t. Then, given $I^{(t)}$, all edges which are in conflict with the edges in $I^{(t)}$ constitute $X^{(t)}$ which denotes the set of edges which must be excluded from the solution. Hence, at any B&B node t we solve a subproblem of the MWPMC, say MWPMC$^{(t)}$, which is defined on subgraph $G^{(t)} = (V(G^{(t)}), E(G^{(t)}))$ of G, where $V(G^{(t)})$ is the set of vertices excluding the ones incident to the edges in $I^{(t)}$ and $E(G^{(t)}) = E(G) \setminus \{I^{(t)} \cup X^{(t)}\}$,

Algorithm 1: Branch-and-bound algorithm for solving MWPMC (MWPSS) using MWSS relaxations

Input: A graph $G = (V(G), E(G))$ edge weights $c_e \geq 0$, conflict graph $C = (V(C), E(C))$;
Output: A maximum weight conflict free perfect matching $M^* = (V(M^*), E(M^*))$
begin
(Initialization): Set $t = 0$, MWPMC$^{(0)} \leftarrow$ MWPMC, G$^{(0)}$= G, C$^{(0)}$= C, $\mathcal{L}$ = {MWPMC$^{(0)}$}, $I^{(0)}$= $\emptyset$, $X^{(0)}$= $\emptyset$,$\underline{z} = -\infty$
(Termination test): **if** $\mathcal{L} = \emptyset$, **then** output $E(M^*)$ and **stop**.
(Lower bounding): Select and delete a problem from $\mathcal{L}$, say MWPMC$^{(t)}$,
if $\left|E(G^{(t)})\right| < \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ **then** there is no conflict free perfect matching G with edges in $I^{(t)} \cup E(G^{(t)})$. Set $\overline{z}^{(t)} = -\infty$, to prune MWPMC$^{(t)}$ and go to ***Pruning***
else
  Detect the maximum cardinality stable set of $C^{(t)}$, i.e. $S^{(t)}$ and compute the stability number $\alpha(C^{(t)})$
  if $\alpha(C^{(t)}) < \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ **then** there is no conflict free perfect matching of G with edges in $I^{(t)} \cup E(G^{(t)})$. Set $\overline{z}^{(t)} = -\infty$ to prune MWPMC$^{(t)}$ and go to ***Pruning***
  else
    if $\alpha(C^{(t)}) = \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ **then** $I^{(t)} \cup S^{(t)}$ are the edges of a conflict free perfect matching;
      if $w(I^{(t)} \cup S^{(t)}) > \underline{z}$ **then** update the lower bound and incumbent by setting $\underline{z} = w(I^{(t)} \cup S^{(t)})$, $E(M^*) \leftarrow I^{(t)} \cup S^{(t)}$ and go to ***Upper bounding***
    end if
  end if
end if
(Upper bounding): Solve MWSS$^{(t)}$ relaxation on $C^{(t)}$
Let $S_w^{(t)}$ be a maximum weight stable set on $C^{(t)}$ and $\alpha_w(C^{(t)})$ be its weight
if $S_w^{(t)} \neq \emptyset$ **then** Set $\overline{z}^{(t)} = \alpha_w(C^{(t)}) + w(I^{(t)})$
else Set $\overline{z}^{(t)} = -\infty$
end if
(Pruning):
i. if $\overline{z}^{(t)} \leq \underline{z}$, **then** go to ***Termination test***.
ii. if $\left|S_w^{(t)}\right| = \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ and $\overline{z}^{(t)} > \underline{z}$ **then** $I^{(t)} \cup S_w^{(t)}$ are the edges of a better perfect matching. Update the lower bound $\underline{z} = \overline{z}^{(t)}$ and update the incumbent $E(M^*) \leftarrow I^{(t)} \cup S_w^{(t)}$ go to ***Termination test***
iii. if $\left|S_w^{(t)}\right| < \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ and $\overline{z}^{(t)} > \underline{z}$ **then** $I^{(t)} \cup S_w^{(t)}$ are not the edges of a conflict free perfect matching. Find a vertex-maximal induced subgraph $C\left[W^{(t)}\right]$ of $C^{(t)}$ with $W^{(t)} \subseteq V(C^{(t)})$ and go to ***Division***.
(Division): Order the subset of vertices $V(C^{(t)}) \setminus W^{(t)}$ into a sequence $\{e_1, e_2, \ldots, e_m\}$ where $m = \left|W_{(t)} \setminus V(C^{(t)})\right|$. Then create m subproblems $\{\text{MWPSS}^{(ti)}\}$ for $i = 1, 2, \ldots, m$ where $\{\text{MWPSS}^{(ti)}\}$ is obtained from $\{\text{MWPSS}^{(t)}\}$ by enforcing edge e_i to be in the perfect matching. Add them to the active node list $\mathcal{L}$ with $\overline{z}^{(ti)}$=$\overline{z}^{(t)}$ for $i = 1, 2, \ldots, m$ and go to ***Termination test***
end

which are so-called free edges. On the other hand, at node t we define the extended conflict graph $C^{(t)} = (V(C^{(t)}), E(C^{(t)}))$ where $V(C^{(t)})$ correspond to $E(G^{(t)})$ and $E(C^{(t)})$ is the set edges of $C^{(t)}$ which correspond to conflicting edge pairs in $E(G^{(t)})$ and the edge pairs in $E(G^{(t)})$ incident with the same vertex in $V(G^{(t)})$. To obtain lower and upper bounds on the MWPMC$^{(t)}$ we solve the Maximum (cardinality) Stable Set Problem (MSS) and MWSS on $G^{(t)}$, respectively. To this end, we run the exact algorithm devised in [3] as a subprocedure in both *Lower Bounding* and *Upper Bounding* steps of Algorithm 1.

Observe that in the third case of the *Pruning* step, $\left|S_w^{(t)}\right| < \frac{|V(G)|}{2} - \left|I^{(t)}\right|$ and $\overline{z}^{(t)} > \underline{z}$ hold. This implies that the current maximum weight stable set $S_w^{(t)}$ on $C^{(t)}$ together with the edges in $I^{(t)}$ does not yield a perfect matching in G. Therefore, we need to insert at least one edge of $G^{(t)}$ from the set $V(C^{(t)}) \setminus S_w^{(t)}$ into the solution. Then, at this stage we run a greedy heuristic to find a maximal induced set in $G^{(t)}$. That is to say, given $S_w^{(t)}$ we try to find $\left|S_w^{(t)}\right|$ disjoint cliques of $G^{(t)}$ and the union of these cliques is denoted with $W^{(t)} \subseteq V(C^{(t)})$. Then each edge in $V(C^{(t)}) \setminus W^{(t)}$ is inserted one by one into the solution and hence we create $m = \left|V(C^{(t)}) \setminus W^{(t)}\right|$ subproblems.

There is no standard test library for the MMPWC hence we have randomly produced 25 test instances. All experiments are performed on an HPE SRV DL380 GEB9 Server with 2.20 GHz-E5-2650-v4 processor and 192 GB RAM operating within Windows Server 2016. We have solved the BILP formulation given with (1)-(4) via CPLEX 12.7.0 The number of vertices (edges), namely $|V(G)|$ ($|E(G)|$) varies from 10 to 44 (from 10 to 757). The number of conflicting edge pairs, i.e. $|E(C)|$, are between 25 and 12000. We have observed the average CPU time required by the B&B algorithm is 57.25 secs. However, this value is 58.78 secs. for the BILP formulation solved with CPLEX. In our computations we have imposed a CPU time limit of 600 secs. In all cases we have obtained optimum solutions with the BILP formulation within the CPU time limit. On the other hand, in only one case, although the B&B algorithm found the optimum solution, it stopped before enumerating all active B&B nodes in $\mathcal{L}$ because of the CPU time limit.

Acknowledgement

Acknowledgement: This work is supported by the Galatasaray University Scientific Research Project (Grant No: 18.402.009) and the Scientific and Technical Research Council of Turkey - TÜBİTAK (Grant No: 217M477)

References

[1] Andreas Darmann, Ulrich Pferschy, Joachim Schauer, and Gerhard J. Woeginger. "Path, trees and matchings under disjunctive constraints". In: *Discrete Applied Mathematics* 159 (2011), pp. 1726–1735.

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

[3] Stephan Held, Cook William, and Edward C. Sewell. "Maximum-weight stable sets and safe lower bounds for graph coloring". In: *Mathematical Programming Computation* 4 (2012), pp. 363–381.

[4] Temel Öncan and İ. Kuban Altınel. "A Branch-and-Bound Algorithm for the Minimum Cost Bipartite Perfect Matching Problem with Conflict Pair Constraints". In: *Electronic Notes in Discrete Mathematics* 64 (2018), pp. 5–14.

Multi-diffusion in Hb-graphs

Xavier Ouvrard[1,2], Jean-Marie Le Goff[1], and Stéphane Marchand-Maillet[2]

[1]CERN, 1 Esplanade des Particules, 1211 Geneva 23, Switzerland
[2]University of Geneva, CUI, Battelle (Bat A)- Route de Drize, 7 - 1227 Carouge, Switzerland

Abstract

The concept of hyper-bag-graphs (hb-graphs for short) has been introduced in Ouvrard et al. (2018c). A hb-graph is defined as a family of multisets - called hb-edges - of same universe, considered as the hb-graph vertex set. Hb-graphs extend hypergraphs by allowing vertices to have a hb-edge dependent multiplicity function. A hb-graph is called a natural hb-graph when all individual multiplicities are non-negative integers. Such a hb-graph supports repetitions of vertices in its hb-edges. Hb-graphs were firstly introduced to allow the construction of a tensor of e-adjacency in general hypergraphs that is interpretable in term of hb-graph m-uniformisation, allowing to make an alternative proposal to the one made in Ouvrard et al. (2017, 2018b). Due to the hb-edge dependent multiplicity function, hb-graphs can store additional information compared to hypergraphs and therefore provide richer support for knowledge discovery. Using exchange-based diffusion, hb-graphs have already shown their efficiency for retrieving the importance of vertices and hb-edges in co-occurence networks (Ouvrard et al. (2018d)). Hb-graphs have also shown their efficiency in the visualisation of co-occurence networks by allowing to extend the hypergraph-based framework proposed in Ouvrard et al. (2018a).[1]

1 Hb-graphs[2]

A full introduction to hb-graphs and their applications is made in Ouvrard et al. (2019c) where basics on multisets are also given. We consider $V = \{v_i : i \in [\![n]\!]\}$ a nonempty finite set. A **hyper-bag-graph** - or **hb-graph** - is a family of msets of universe V and support a subset of V. The msets are called the **hb-edges** and the elements of V the **vertices**. We write $\mathcal{H} = (V, E)$ such a hb-graph and $E = (e_i)_{i \in [\![p]\!]}$ the family of hb-edges. Each hb-edge $e_i \in E$ is of universe V and has a multiplicity function associated to it: $m_{e_i} : V \to \mathbb{W}$ where $\mathbb{W} \subseteq \mathbb{R}^+$. When there is no ambiguity, the notation m_i is used for m_{e_i} and m_{ij} for $m_{e_i}(v_j)$.

A hb-graph is with **no repeated hb-edges** if: $\forall i_1 \in [\![p]\!], \forall i_2 \in [\![p]\!] : e_{i_1} = e_{i_2} \Rightarrow i_1 = i_2$.

A hb-graph where each hb-edge is a mset with multiplicity range a subset of $\mathbb{N}$ then the hb-graph is called a **natural hb-graph**. For a general hb-graph each hb-edge has to be seen as a weighted system of vertices, where the weights of each vertex are hb-edge dependent. In a natural hb-graph the multiplicity function can be viewed as a duplication of the vertices.

The **support hypergraph** of a hb-graph is the hypergraph with vertex set the hb-graph vertex set and with hyperedge family the family of hb-edge supports of the hb-graph. In multisets, as elements have multiplicities, one can refer to the m-cardinality of the multiset as the sum of the multiplicity of its elements. The cardinality of a multiset corresponds to the cardinality of its support. Hence, hypergraph features like range, co-range, uniform hypergraph, degree, regular

[1]cf poster presented at AMLD on https://www.infos-informatique.net
[2]This section has already been presented at CBMI 2018 and MCCCC 32

hypergraph can be extended to hb-graphs by considering its support hypergraph. For the hb-graph itself, the multiplicity of the hb-edge elements has to be considered using the m-notion associated: m-range, m-co-range, m-uniform hb-graph, m-degree, m-regular.

We show that a hb-graph can be associated to a numbered-copy hypergraph in a unique manner. Paths and cycles in hypergraphs can be extended with refinements to m-paths and m-cycles in hb-graphs as the location of the copy of the intermediate vertices can also differ (inside the intersection or in the union). The connectivity in a hb-graph is related to its support hypergraph connectivity.

k-adjacency refers to k vertices that considered as a m-set are included in one hb-edge. $\overline{k}$-adjacency refers to the maximal k such that k-adjacency in the hb-graph is ensured. Vertices of a hb-edge with a nonzero multiplicity are said e-adjacent. $\overline{k}$-adjacency and e-adjacency refer to the same notion in m-uniform hb-graphs, but differ in general hb-graph. A vertex in a hb-edge is said to be incident to this hb-edge. Hb-graphs can be represented either using the edge standard or in the subset standard, like it has been achieved for hypergraphs by Mäkinen (1990).

We define the incidence matrix of a hb-graph - intensively used in Ouvrard et al. (2018d) for diffusion by exchanges in hb-graphs - as the matrix $H$of the vertex multiplicity inside the hb-edges, i.e.: $H \triangleq \left[m_j\left(v_i\right)\right]_{\substack{i\in[\![n]\!]\\ j\in[\![p]\!]}}$.

Building a tensor of e-adjacency for general hb-graphs has been achieved in Ouvrard et al. (2019c). It takes several steps and it is coupled to a m-uniformisation process of the original hb-graph.

Definition 1. *The **straightforward** e**-adjacency tensor** $\mathcal{A}_{str,\mathcal{H}}$ **of a hb-graph** $\mathcal{H}=(V,E)$ of m-rank $r_{\mathcal{H}}$* [3] *is the tensor of canonical hypermatrix representation $\boldsymbol{A}_{str,\boldsymbol{\mathcal{H}}}$ defined by: $\boldsymbol{A}_{str,\boldsymbol{\mathcal{H}}} \triangleq \sum_{i\in[\![p]\!]} c_{e_i}\boldsymbol{R}_{\boldsymbol{e_i}}$ where for $e_i=\left\{v_{j_1}^{m_{ij_1}},\ldots,v_{j_{k_i}}^{m_{ij_{k_i}}}\right\}\in E$, $c_{e_i}=\dfrac{r_{\mathcal{H}}}{\#_m e_i}$ is the dilatation coefficient and where $\boldsymbol{R}_{\boldsymbol{e_i}}=\left(r_{i_1\ldots i_{r_{\mathcal{H}}}}\right)$ is the hypermatrix whose elements have only two possible values: 0 and: $\rho_{str,e_i}=\dfrac{m_{ij_1}!\ldots m_{ij_{k_i}}!m_{i\,n+1}!}{r_{\mathcal{H}}!}\#_m e_i$ - with $m_{i\,n+1}=r_{\mathcal{H}}-\#_m e_i$. The indices of the non-zero elements of $\boldsymbol{R}_{\boldsymbol{e_i}}$ are obtained by permutation of the elements of the multiset: $\left\{j_1^{m_{ij_1}},\ldots,j_{k_i}^{m_{ij_{k_i}}},(n+1)^{m_{i\,n+1}}\right\}$.*

This tensor is nonnegative, symmetric, globally invariant to vertex permutations in the original hb-graph. This tensor allows the unique reconstruction of the hb-graph it is originated from. The fact that the tensor is symmetric allows it to be described in the number of hb-edges. We have shown that the m-degree of vertices can be retrieved from the e-adjancy tensor as well as the hb-edge distribution and the hb-graph m-uniformity level. Some first results on spectral analysis of hb-graphs have been adapted from the results obtained in Qi and Luo (2017) for nonnegative symmetric tensors. The additional vertex increases some of the bounds obtained in Qi and Luo (2017).

2 Knowledge discovery with hb-graphs[4]

We introduced an exchange-based diffusion in hb-graphs in Ouvrard et al. (2018d) which is incident-matrix-based. This diffusion process allows not only to rank vertices but also hb-edges on the basis of their connectivity and centrality. The exchange-based diffusion, as it is shown in Figure 1, iterates

[3] $r_{\mathcal{H}}=\max_{e\in E}\#_m(e)$, where $\#_m(e)=\sum_{v\in V} m_e\left(v\right)$ is the m-cardinality of the hb-edge $e\in E$

[4] The exchange-based diffusion has been presented at CBMI 2018. The concepts and results concerning the multi-diffusion have not yet been published.

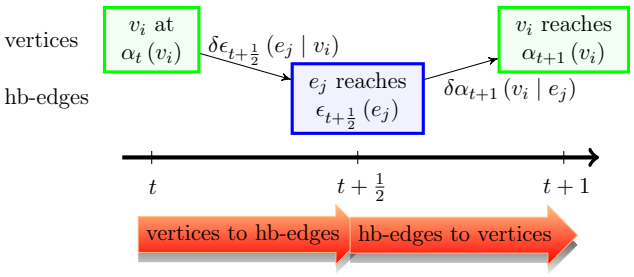


Figure 1: Diffusion by exchange: principle Ouvrard et al. (2018d)

a two-phase step over vertices and hb-edges. In the first phase, the vertices diffuse their values to the hb-edges they are incident in a proportion corresponding to their normalised weighted multiplicity in these hb-edges. In the second phase, the hb-edges share their values with the vertices they hold proportionnaly to the normalised multiplicity of their vertices. The processus has been shown to be stochastic in Ouvrard et al. (2019b) and to converge in any connected hb-graph. It ranks the vertices with relation to their m-degree and the hb-edges according to their m-cardinality. A full evaluation has been conducted on random hb-graphs and has been applied to a co-occurence network of keywords extracted from Arxiv metadata abstracts.

Ouvrard et al. (2018a) build a hypergraph framework for modeling and visualising information in a multi-facetted co-occurence network. The information space is viewed by its multiple facets that are linked using the common references. We have refined this approach to use hb-graphs for the visualisation part: it has been implemented to visually query Arxiv with an enriched browsing experience[5].

In this part we introduce the mathematical aspects of a multi-diffusion process, that combines both the work done on diffusion over hb-graphs and the hb-graph-extended hypergraph framework for the modeling of co-occurence networks. This new still on-going work will allow us to enhance knowledge discovery in a multifacetted information space. This work uses the information of the diffusion to allow global knowledge discovery at the information space level.

To build co-occurences over a dataset that is multi-facetted, one chooses a dimension of reference that has to be the same for all the facets. For instance, in a publication database, the publication can be chosen as a reference and the different types of metadata attached to the publication in the dataset, such as keywords, authors, organisations, ... can be the basis for the different facets of the information space.

We consider an information space with $K+1$ facets containing data instances of different types $(T_k)_{1\leqslant k\leqslant K+1}$, one of them being chosen as the type of reference. We can always consider that it is the $K+1$-th, even if we need to reorder the sequence of types. The data occurences of a given type T_k, $1 \leqslant k \leqslant K+1$ contitute a set V_k attached to the k-th facet. We write $R = V_{K+1}$ the set of references. Each facet is modeled with a hb-graph $\mathcal{H}_k = (V_k, E_k)$, $1 \leqslant k \leqslant K$. The vertex set V_k corresponds to the different occurences of data instances of type T_k found in the information space. E_k is the family of hb-edges, each hb-edge being constituted of co-occurences of data instances of type T_k that are attached to individual references $r \in R$. This family can potentially contain repeated hb-edges.

We consider $\mathcal{R}_k$ the equivalence relation defined for $e, e' \in E_k$ by: $e\mathcal{R}e' \Leftrightarrow e = e'$. We write for $e \in E_k$: $\overline{e} = \{e' \in E_k : e\mathcal{R}_k e'\}$ and $\overline{E_k} = E_k/\mathcal{R}_k$ the set of non-repeated hb-edges. We define an

[5]Poster presented at AMLD Lausanne 2019 https://www.infos-informatique.net, article under writing at the time of submission

application $w_e : \overline{E_k} \to \mathbb{N}$ such that for $\overline{e} \in \overline{E_k} : w_e(\overline{e}) = |\overline{e}|$.

We write: $\overline{\mathcal{H}_k} = \left(V_k, \overline{E_k}, w_{e,k}\right)$ the weighted hb-graph with no repeated hb-edges.

We consider: $\rho_k : R \to \overline{E_k}$ such that: $\forall r \in R, \exists \overline{e} \in \overline{E_k} : \rho_k(r) = \overline{e}$; ρ_k is surjective.

We consider also: $\nu_k : \overline{E_k} \to \mathcal{P}(R)$ such that: $\forall \overline{e} \in \overline{E_k} : \nu_k(\overline{e}) = \{r \in R : \rho_k(r) = \overline{e}\}$.

As the set R of references is common to the different hb-graphs, we can consider: $\rho : R \to \overline{E_1} \times ... \times \overline{E_K}$ such that: $\forall r \in R : \rho(r) = (\rho_1(r), ..., \rho_K(r))$.

Keeping the same set of references for all facets, the hb-graphs of the different facets are inter-connected via the references chosen for their construction. This interconnection enables the navigation through the different facets as it is shown in Ouvrard et al. (2018a) and in Ouvrard et al. (2019a): the references act as pivots in between the different facets interconnecting them.

We can then consider for all $r \in R$: $e_r = \{\rho_1(r)^{m_{r,1}}, ..., \rho_K(r)^{m_{r,K}}\}$. We have: $\forall j \in [\![k]\!] : m_{r,j} = w_{e,j}(\rho_k(r))$. e_r is the multiset of the class of hb-edges that are linked to the reference r for the different facets viewed as vertices of a new hb-graph $\mathcal{H} = \left(\bigcup_{i \in K} \overline{E_i}, E\right)$, where the vertex set is constituted of the hb-edge labels of the quotient set of hb-edges of the different facets.

A diffusion by exchange is always possible not only on each hb-graph $\overline{\mathcal{H}_k}$ but also on $\mathcal{H}$. But in this case the information obtained is only related to the connectivity in each hb-graph. Different strategies are foreseen to overcome this difficulty: they are presented at this conference.

References

Mäkinen, E., 1990. How to draw a hypergraph. International Journal of Computer Mathematics 34, 177–185.

Ouvrard, X., Le Goff, J., Marchand-Maillet, S., 2018a. Hypergraph modeling and visualisation of complex co-occurence networks. Electronic Notes in Discrete Mathematics 70, 65–70.

Ouvrard, X., Le Goff, J., Marchand-Maillet, S., 2018b. On adjacency and e-adjacency in general hypergraphs: Towards a new e-adjacency tensor. Electronic Notes in Discrete Mathematics 70, 71–76.

Ouvrard, X., Le Goff, J., Marchand-Maillet, S., 2019a. Hb-graph modeling and visualisation of complex co-occurence networks. Article under writing .

Ouvrard, X., Le Goff, J.M., Marchand-Maillet, S., 2017. Adjacency and tensor representation in general hypergraphs part 1: e-adjacency tensor uniformisation using homogeneous polynomials. arXiv preprint arXiv:1712.08189 .

Ouvrard, X., Le Goff, J.M., Marchand-Maillet, S., 2018c. Adjacency and tensor representation in general hypergraphs. part 2: Multisets, hb-graphs and related e-adjacency tensors. arXiv preprint arXiv:1805.11952 .

Ouvrard, X., Le Goff, J.M., Marchand-Maillet, S., 2018d. Diffusion by exchanges in hb-graphs: Highlighting complex relationships. CBMI Proceedings .

Ouvrard, X., Le Goff, J.M., Marchand-Maillet, S., 2019b. Diffusion by exchanges in hb-graphs: Highlighting complex relationships extended version. Under submission .

Ouvrard, X., Le Goff, J.M., Marchand-Maillet, S., 2019c. On hb-graphs and their application to general hypergraph e-adjacency tensor. Article under submission to the MCCCC32 proceedings .

Qi, L., Luo, Z., 2017. Tensor analysis: spectral theory and special tensors. volume 151. SIAM.

Mathematical programming for influence diagrams

Axel Parmentier[1], Victor Cohen[1], Vincent Leclère[1], Guillaume Obozinski[2], and Joseph Salmon[3]

[1]École des Ponts Paristech, CERMICS, Marne-la-Vallée, France
[2]École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
[3]IMAG, Univ Montpellier, CNRS, Montpellier, France

Abstract

Influence Diagrams are a flexible tool to represent discrete stochastic optimization problems, including Markov Decision Process (MDP) and Partially Observable MDP as standard examples. More precisely, given random variables considered as vertices of an acyclic digraph, a probabilistic graphical model defines a joint distribution via the conditional distributions of vertices given their parents. In an influence diagram, the random variables are represented by a probabilistic graphical model whose vertices are partitioned into three types : chance, decision and utility vertices. The user chooses the distribution of the decision vertices conditionally to their parents in order to maximize the expected utility. We present a mixed integer linear formulation for solving an influence diagram, as well as valid inequalities, which lead to a computationally efficient algorithm. We also show that the linear relaxation yields an optimal integer solution for instances that can be solved by the "single policy update", the default heuristic algorithm for addressing influence diagrams.

Context

Machine learning techniques are applied with growing success in Operations Research [Bengio et al., 2018]. Most contributions apply supervised learning techniques to take heuristic choices within discrete optimization algorithms. Surprisingly, Bengio et al. do not mention contributions that use structured learning [Nowozin et al., 2011], the branch of Machine Learning whose goal is to leverage combinatorial structure in learning algorithms. Graphs, and more precisely, probabilistic graphical models [Koller and Friedman, 2009], play a central role in structured learning. We believe that using probabilistic graphical models can be fruitful in stochastic optimization. There are two main options to solve a stochastic optimization problem numerically: computations are done either directly on the distribution, or on a sample approximation. The first option has many advantages but suffers from the curse of dimensionality. Fluid approximations have been proposed [Bertsimas and Mišić, 2016, Waserhole et al., 2013] to break the curse of dimensionality. These fluid approximations happen to be special cases of variational approximations, which have been proposed in graphical model theory [Koller and Friedman, 2009] for the same reason, and widely studied in that context. But this link is not made in the fluid approximation literature. The present work is the fruit of a collaboration between researchers in machine learning and researchers in operations research to bridge the gap between the two communities on that topic. This talk will be tailored for an Operations Research audience.

We place ourselves in the context of influence diagrams, a framework to model discrete stochastic optimization problems using probabilistic graphical models. We show how problems considered using fluid approximations fit in that context. We introduce mixed integer linear programming

formulations for influence diagrams, and leverage the notion of d-separation from probabilistic graphical model theory to introduce a family of valid inequalities. These valid inequalities can be used in the context of fluid approximations, and more generally to strengthen mixed integer linear programming formulation that work on distributions. The rest of this abstract introduces the framework of influence diagrams, briefly surveys the literature, and states our contributions. The work behind this talk is available in the following preprint [Parmentier et al., 2019].

The framework of influence diagrams

Let $G = (V, E)$ be a directed graph, and for each vertex v in V, let X_v be a random variable taking value in a finite state space $\mathcal{X}_v$. We say that, the random vector $X_V := \{X_v \,|\, v \in V\}$ factorizes as a *directed graphical model* on G if, for all $x_V \in \prod_{v\in V} \mathcal{X}_v$, we have

$$\mathbb{P}(X_V = x_V) = \prod_{v\in V} p(x_v|x_{\mathrm{prt}(v)}), \tag{1}$$

where $\mathrm{prt}(v)$ is the set of parents of v, that is, the set of vertices u such that (u, v) belongs to E, and $p(x_v|x_{\mathrm{prt}(v)}) = \mathbb{P}(X_v = x_v|X_{\mathrm{prt}(v)} = x_{\mathrm{prt}(v)})$. Further, given an arbitrary collection of conditional distribution $\left\{p(x_v|x_{\mathrm{prt}(v)})\right\}_{v\in V}$, Equation (1) uniquely defines a probability distribution on $\mathcal{X}_V = \prod_{v\in V} \mathcal{X}_v$.

Influence diagrams are stochastic optimization problems where probabilities are modeled using a probabilistic graphical model. Let $(V^{\mathrm{d}}, V^{\mathrm{c}}, V^{\mathrm{u}})$ be a partition of V where V^{c} is the set of *chances vertices*, V^{d} is the set of *decision vertices*, and V^{u} is the set of *utility vertices* (the ones with no descendants). We say that $(V^{\mathrm{u}} \cup V^{\mathrm{c}}, V^{\mathrm{d}}, E)$, sometimes simply denoted by G is an *influence diagram*. Consider a set of conditional distributions $\mathfrak{p} = \left\{p(x_v|x_{\mathrm{prt}(v)})\right\}_{v\in V^{\mathrm{u}}\cup V^{\mathrm{c}}}$, and a collection of *reward functions* $r = \{r_v\}_{v\in V^{\mathrm{u}}}$, with $r_v : \mathcal{X}_v \to \mathbb{R}$. Then we call $(G, \mathcal{X}_V, \mathfrak{p}, r)$ a *parametrized influence diagram*.

Let Δ_v denote the set of conditional distributions $\delta_{v|\mathrm{prt}(v)}$ on $\mathcal{X}_v$ given $\mathcal{X}_{\mathrm{prt}(v)}$. Given the set of conditional distributions $\mathfrak{p}$, a *policy* $\delta \in \Delta = \prod_{v\in V^{\mathrm{d}}} \Delta_v$, uniquely defines a distribution $\mathbb{P}_\delta$ on $\mathcal{X}_V$ through

$$\mathbb{P}_\delta(X_V = x_V) = \prod_{v\in V^{\mathrm{u}}\cup V^{\mathrm{c}}} p(x_v|x_{\mathrm{prt}(v)}) \prod_{v\in V^{\mathrm{d}}} \delta_{v|\mathrm{prt}(v)}(x_v|x_{\mathrm{prt}(v)}). \tag{2}$$

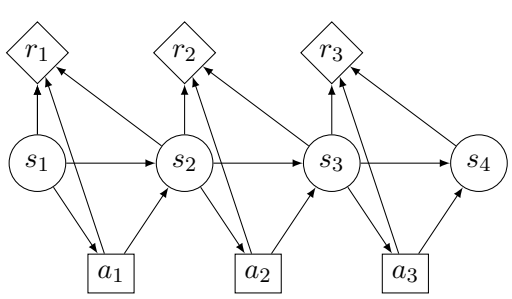
Let $\mathbb{E}_\delta$ denote the corresponding expectation. The purpose of this talk is to provide tractable mathematical programming formulation for the *Maximum Expected Utility* problem

$$\max_{\delta\in\Delta} \quad \mathbb{E}_\delta\left(\sum_{v\in V^{\mathrm{u}}} r_v(X_v)\right), \tag{3}$$

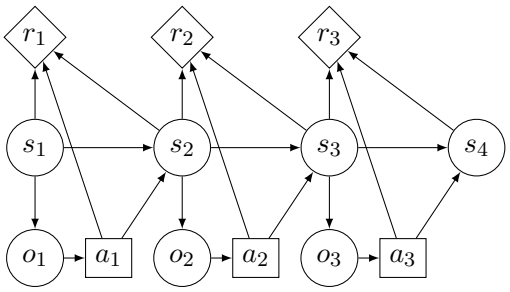
which aims at finding a policy δ that maximizes utilities.

Influence diagrams can model several stochastic optimization problems as special cases. Consider for instance a maintenance problem where at time t a machine is in state s_t. The action a_t taken by the decision maker according to the current state is typically maintaining it (which is costly) or not (which increase the probability of failure). State and decision yields a new (random) state s_{t+1}, and the triple (s_t, a_t, s_{t+1}) induce a reward r_t. This is an example of Markov decision processes (MDP) which are probably the simplest influence diagram, represented in Figure 1a.

In practice, the actual state s_t of the machine is not known, but we can only have some observation o_t, which leads to a more complex influence diagram known as Partially observed Markov decision processes (POMDP). In theory an optimal decision should be taken knowing all



(a) A Markov decision process (MDP)



(b) A Partially Observed Markov Decision Process (POMDP)

Figure 1: Influence diagrams examples, where we represent chance vertices ($V^{\mathrm{u}} \cup V^{\mathrm{c}}$) in circles, decision vertices (V^{d}) in rectangles, and utility vertices in diamonds (V^{u}).

past obervations and decisions (which is the *perfect recall* case). However, this would leads to untractable decision strategies which requires long memory. It is usual to restrict the decision a_t to be taken only with respect to observation o_t, as illustrated in Figure 1b.

Influence diagrams enable to model richer interactions. Consider for instance two chess players : Bob and Alice. They are used to play chess and for each game they bet a symbolic coin. However, they can refuse to play. Suppose that Alice wants to play chess every day. At each time step, she has a current confidence level s_t. The day of the game, her current mental fitness is denoted o_t. When Bob meets Alice, he takes the decision to play depending on her attitude and her appearance of the day, denoted u_t. Then Bob can accept or decline the challenge, and his decision is denoted a_t. Let v_t denote the winner (getting a reward r_t). Then, Alice's next confidence level is affected by the result of the game and her previous confidence level. This stochastic decision problem can be modeled by an influence diagram as shown in Figure 2.

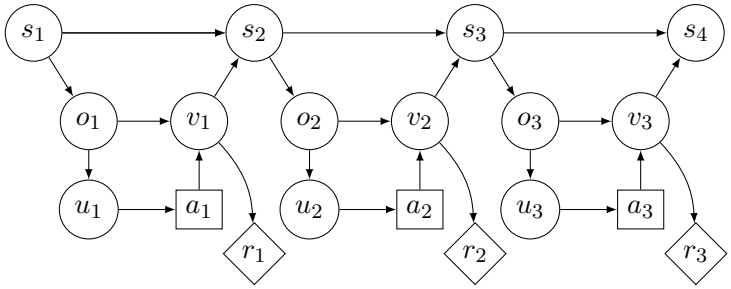


Figure 2: Bob and Alice chessgame

Influence diagram were introduced by Howard and Matheson [1984] to model stochastic optimization problems using a probabilistic graphical model framework. Originally the decision makers were assumed to have perfect recall [Shachter, 1986]. Lauritzen and Nilsson [2001] relaxed this assumption and provided a simple (coordinate descent) algorithm to find a good policy: the Single Policy Update (SPU) algorithm. The same authors also introduced the notion of soluble ID as a sufficient condition for SPU to converge to an optimal solution. This notion has been generalized by Koller and Milch [2003] to obtain a necessary and sufficient condition. SPU finds a locally optimal policy, but performs exact inference, and is therefore limited by the treewidth. More recently,

Mauá and Campos [2011] have introduced a new algorithm, *Multiple Policy Update*, which has both an exact and a heuristic version and relies on dominance to discard partial solutions. To the best of our knowledge, mathematical programming approaches have not been proposed for influence diagrams.

Contributions

We provide a mixed integer linear programming (MILP) formulation to the MEU Problem (3), and valid inequalities that strengthen the formulation. Numerical experiments show the relevance of our approach. We give interpretation the linear relaxations obtained in terms of graphs. Finally, we show that influence diagrams that can be solved to optimality through SPU can be solved by (continuous) linear programming using our formulation.

References

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *arXiv preprint arXiv:1811.06128*, 2018.

Dimitris Bertsimas and Velibor V Mišić. Decomposable markov decision processes: A fluid optimization approach. *Operations Research*, 64(6):1537–1555, 2016.

RA Howard and JE Matheson. Influence diagrams, readings in the principles and practice of decision analysis. *Strategic Decision Systems, Menlo Park, Calif*, 1984.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Daphne Koller and Brian Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181 – 221, 2003. ISSN 0899-8256. doi: https://doi.org/10.1016/S0899-8256(02)00544-4. URL `http://www.sciencedirect.com/science/article/pii/S0899825602005444`. First World Congress of the Game Theory Society.

Steffen L Lauritzen and Dennis Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47(9):1235–1251, 2001.

Denis D Mauá and Cassio Campos. Solving decision problems with limited information. In *Advances in Neural Information Processing Systems*, pages 603–611, 2011.

Sebastian Nowozin, Christoph H Lampert, et al. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

Axel Parmentier, Victor Cohen, Vincent Leclère, Guillaume Obozinski, and Joseph Salmon. Mathematical programming for influence diagrams. *arXiv preprint arXiv:1902.07039*, 2019.

Ross D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986. doi: 10.1287/opre.34.6.871. URL `https://doi.org/10.1287/opre.34.6.871`.

Ariel Waserhole, Vincent Jost, and Nadia Brauner. Pricing techniques for self regulation in vehicle sharing systems. *Electronic Notes in Discrete Mathematics*, 41:149–156, 2013.

Ground staff shift planning under delay uncertainty at Air France

Julie Poullet[1] and Axel Parmentier[2]

[1]École polytechnique, France, julie.poullet@polytechnique.edu
[2]CERMICS, École des Ponts Paristech, France, axel.parmentier@enpc.fr

Abstract

Airlines' ground staff agents perform many jobs at airports such as passengers check-in and planes cleaning. Shift planning aims at building the sequences of jobs operated by ground staff agents, so that all jobs are operated at minimum cost. Flight leg delays disrupt ground staff schedules, which leads to high additional costs. We therefore introduce a stochastic version of the shift planning problem that takes into account the cost of disruptions due to delay and a column generation approach to solve it. Column generation's key element is the algorithm for the pricing subproblem, which we model as a stochastic resource constrained shortest path problem. Numerical experiments on industrial instances have proven the relevance and efficiency of our approach. Including delay costs allows airlines to reduce the total operating costs by 3% to 5%, and column generation can solve to optimality instances with up to two hundred and fifty jobs, and one hundred scenarios.

Keywords: *Stochastic ground staff scheduling, stochastic shift planning, column generation, stochastic resource constrained shortest path, flights delay*

1 Introduction

Deterministic ground staff scheduling problems have been less studied than airplane and crew scheduling problems as they are not specific to airlines and can be treated using standard approaches of the personnel scheduling literature [1]. However, this last assertion becomes false when delay is taken into account. Indeed, the personnel scheduling literature focuses on uncertainty about demand volume, demand arrival time, and manpower availability [1], but does not take into account the propagation of delay. On the contrary, the construction of sequences of flight legs that do not propagate delay has been extensively studied for airplanes and crews, see e.g. [2, 3, 4]. But to the best of our knowledge, the problem of building sequences of tasks for ground staff agents that do not propagate delay has not been considered yet.

This work is the result of a project initiated by Air France, the main French airline, to build schedules resilient to delay. We propose a *stochastic ground staff shift planning problem* which takes into account delay propagation in Air France's operational model. Stochasticity is modeled using scenario based distributions.

To solve this problem, we provide a column generation approach which solves to optimality instances with up to 210 jobs and 200 scenarios in a few minutes. As delay propagates along shifts, stochasticity is handled in the pricing subproblem. The latter is solved through a framework for resource constrained shortest path problem recently introduced by the second author [5]. Inside that framework, a non-trivial algebraic structure models delay propagation. To the best of our knowledge, this is the first practically efficient algorithm for shift planning with delay propagation.

2 Problem statement

Every job j is characterized by a *fixed time interval* $[t_j^{\mathrm{b}},t_j^{\mathrm{e}}]$, where t_j^{b} represents its *beginning time* and t_j^{e} its *ending time* in the nominal case, i.e. without delay. A shift sh is a sequence of jobs $j^1, j^2, \ldots, j^k$, beginning at $h_{\mathsf{sh}}^{\mathrm{b}}$ and finishing at $h_{\mathsf{sh}}^{\mathrm{e}}$, such that $t_{j^i}^{\mathrm{e}} \leq t_{j^{i+1}}^{\mathrm{b}}$ for all i in $\{1,\ldots,k-1\}$, $h_{\mathsf{sh}}^{\mathrm{b}} \leqslant t_{j^1}^{\mathrm{b}}$ and $t_{j^k}^{\mathrm{e}} \leqslant h_{\mathsf{sh}}^{\mathrm{e}}$. Working rules impose that the *total duration* τ between $h_{\mathsf{sh}}^{\mathrm{b}}$ and $h_{\mathsf{sh}}^{\mathrm{e}}$ does not exceed a *maximum duration* $\tau^{\max}$, and that shift contains a *break* whose location is subject to specific constraints. We denote by $\mathcal{SH}$ the set of all feasible shifts.

When a flight is delayed, its delay propagates to the associated jobs, influencing their beginning time, their ending time, or both. It causes potential issues in the schedule of the agent operating them since some job successions may no longer be possible. At Air France, in that situation, *back-up agents* are sent to operate all the jobs that the initial agent cannot operate. A job j is said to be *rescheduled* if it is operated by a back-up agent, which happens when the agent who is supposed to operate the job j is still doing a previous job when j starts.

We make the *simplifying assumption* that a job in a shift sh can be only rescheduled due to the job right before it in sh, but not due to the previous ones.

Operating a shift sh generates two sources of costs for the airline. The first one comes from agents wage and is a non-decreasing function of the total duration $c^{\mathrm{w}}(\tau)$. The second one is due to a fixed cost c^{back} incurred for each rescheduled job. Let $n_{\mathsf{sh}}^{\mathrm{back}}$ be the (random) number of jobs of sh rescheduled. The expected cost c_{sh} of a shift is $c_{\mathsf{sh}} = c^{\mathrm{w}}(h_{\mathsf{sh}}^{\mathrm{e}} - h_{\mathsf{sh}}^{\mathrm{b}}) + \mathbb{E}\big(c^{\mathrm{back}} n_{sh}^{\mathrm{back}}\big)$.

The *stochastic ground staff shift planning problem* consists in finding a set $\mathcal{S}$ of shifts of minimum cost and such that each job in J is operated by at least one shift in $\mathcal{S}$.

3 Column generation approach

The following *master problem* introduces a set-partitioning formulation of the stochastic ground staff shift planning problem.

$$\min_{y} \quad \sum_{\mathsf{sh}\in\mathcal{SH}} c_{\mathsf{sh}} y_{\mathsf{sh}} \tag{1a}$$

$$\text{s.t.} \quad \sum_{\mathsf{sh}\ni j} y_{\mathsf{sh}} = 1, \quad \forall j \in J \tag{1b}$$

$$y_{\mathsf{sh}} \in \{0,1\}, \quad \forall \mathsf{sh} \in \mathcal{SH} \tag{1c}$$

Binary variable y_{sh} indicates if a shift $\mathsf{sh} \in \mathcal{SH}$ belongs to the solution. The notation $\mathsf{sh} \ni j$ means that the job j is realized during the sequence of sh, thus the constraint (1b) ensures that all jobs in J are covered. Master problem (1) is therefore immediately equivalent to our stochastic ground staff shift planning problem.

By denoting by λ_j the dual variable associated to constraint (1b) for job J, the underlying *pricing subproblem* of the master problem is:

$$\min_{\mathsf{sh}\in\mathcal{SH}} c_{\mathsf{sh}} - \sum_{j\in\mathsf{sh}} \lambda_j$$

The instances of the following numerical results come from runway jobs operated by Air France. *All instances are solved to optimality.* Table 1 proves the relevance of our approach, since it enables to reduce the total operating cost by 3% to 5%. These numbers are with respect to the deterministic approach which does not take into account potential rescheduling when building the shifts. It also shows that the difficulty of our approach lies in solving the pricing subproblem.

Instance	Number of tasks	Pricing time (%)	Total time (hh:mm:ss)	Improvement vs deterministic problem (%)
1	49	85.1	00:00:22	3.31
2	111	98.4	00:02:10	4.32
3	210	99.4	00:27:22	5.49
4	256	99.7	03:43:09	5.17

Table 1: Performances of our approach to solve master problem (1) with 200 scenarios

4 Pricing subproblem

The difficulty of the pricing subproblem lies in the non-linearity and the stochasticity of shift cost. Contrary to classical pricing subproblem of that kind, which are often modeled as deterministic shortest path problems, or deterministic resource constrained shortest path problems, we need to model ours as a stochastic resource constrained shortest path problem.

We classically model our problem using a digraph $D = (V, A)$, where the vertices are the jobs, and the arcs are the pairs of jobs which can be chained. For more clarity, we present a simplified situation: a fixed beginning time h^{b} and ending time h^{e} for all shifts, without lunch break. The set of vertices is $V = \{o\} \cup \left(\bigcup_{j \in J} \{j\}\right) \cup \{d\}$, where o and d are the origin and destination vertices, representing the beginning and ending time of the shift. The set of arcs A contains all the ordered pair (j^1, j^2) such that $t^{\mathrm{e}}_{j^1} \leq t^{\mathrm{b}}_{j^2}$, as well as all the pairs (o, j) and (j, d) for j in J.

Proposition 1. *There is a bijection between shifts and o-d paths in D.*

We then seek to decompose the cost c_{sh} along the vertices j, or arcs (j^1, j^2) between two jobs. However, given the delay propagation, the need for rescheduling a job depends on the jobs previously operated in the shift, and impacts the future ones. In other words, the cost of a vertex in a path is determined by the sequences of vertices before it, and impact the costs of the sequence of vertices after it. Figure 1 presents four tasks whose beginning time and ending time are shown both in the nominal case, i.e. without delay, and under a scenario ω, in which delay occurs. We focus on job j^4 and try to assign to it a cost. Let's consider only three different potential shifts: shift $\mathsf{sh}_1 = j^3 - j^4$, $\mathsf{sh}_2 = j^2 - j^4$ and $\mathsf{sh}_3 = j^1 - j^2 - j^4$. These shifts are feasible since all proposed sequences are feasible in the case without delay. However the cost of job j^4 varies under scenario ω, regarding the shift considered. In shift sh_1, job j^4 is not rescheduled. In shift sh_2, job j^4 is rescheduled since job j^2 now finishes after its beginning time. Finally, in shift sh_3, job j^4 is not rescheduled. Indeed the agent operates job j^1, and since job j^2 has to be rescheduled, the agent is available to operate job j^4. The example provided in Figure 1 sheds light on the difficulty of determining whether a task is rescheduled or not. It also underlines the combinatorial part of the problem we are dealing with.

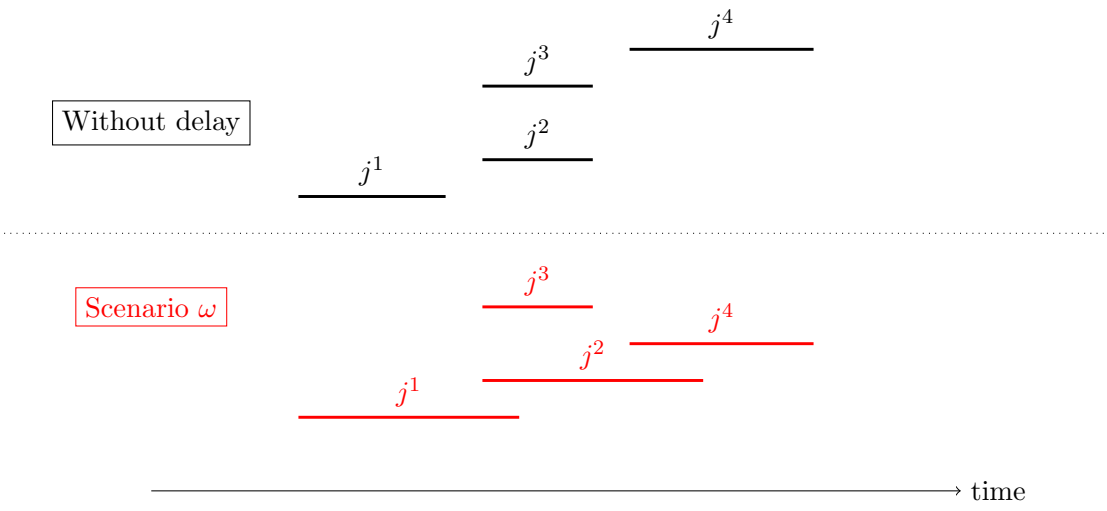


Figure 1: Partial shift sh with five jobs with rescheduled and non-rescheduled first job under two different scenarios

To overcome this difficulty, we model the pricing subproblem within the framework for stochastic resource constrained shortest path problems (MRCSP) recently introduced by the second author [5]. It leverages the notion of lattice ordered monoid to derive an algorithm particularly efficient on stochastic shortest path problem.

Modeling our problem as a MRCSP requires the introduction of an algebraic structure (a lattice ordered monoid) that keeps track, for every scenario, of the duration and the cost of a path P in two cases: when the first task of P is not rescheduled and when it is. It thus enables us to propagate delay along the tasks of a shift.

References

[1] 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, 2013.

[2] Shan Lan, John-Paul Clarke, and Cynthia Barnhart. Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation science*, 40(1):15–28, 2006.

[3] Oliver Weide, David Ryan, and Matthias Ehrgott. An iterative approach to robust and integrated aircraft routing and crew scheduling. *Computers & Operations Research*, 37(5):833–844, 2010.

[4] Michelle Dunbar, Gary Froyland, and Cheng-Lung Wu. An integrated scenario-based approach for robust aircraft routing, crew pairing and re-timing. *Computers & Operations Research*, 45:68–86, 2014.

[5] Axel Parmentier. Algorithms for non-linear and stochastic resource constrained shortest path. *Mathematical Methods of Operations Research*, Sep 2018.

On the Problem Class of Optimal Technology Implementation into a Multisectoral Energy System (OTIMES)

Andreas Schwenk[1]

[1]TH Köln – University of Applied Sciences, Germany

Abstract

We address the problem of implementing energy storage and energy converter devices into an existing multisectoral energy network, described by a graph $G = (V, E)$. The objective is to reduce the total greenhouse gas emissions by using limited monetary investment costs. The underlying problem class is upper bound to $0-1$ Mixed Integer Quadratic Constrained Programming (MIQCP). In this extended abstract, we primarily denote modeling aspects. We will present relaxations to $0-1$ Mixed Integer Linear Programming (MILP) in the talk.

1 Introduction

We consider the problem of Optimal Technology Implementation into a Multisectoral Energy System (abbreviated, OTIMES, $\otimes$). Given a set of energy networks with distinct energy sectors (e.g. power, gas and heat), a set of consumers demands varying amounts of energy over a period of time. A set of producers supplies conventional and renewable energy over a period of time. The underlying network flow problem optimally transmits energy from sources to sinks such that renewable resources are consumed first. To further minimize the total greenhouse gas emissions, we integrate the following technologies into the system:

(a) Consider a surplus of supplied renewable energy at time t_1 that is not demanded immediately. Then, at time $t_2 > t_1$, energy must possibly be retrieved from another, conventional energy supplier with higher greenhouse gas emissions. The implementation of an energy *storage*, e.g. in form of a battery, supports to decrease the *cumulated* waste of renewable sources.

(b) Storing produced renewable energy in the same sector is not always an option. For example, large batteries that buffer power energy are costly, as well as emit high greenhouse gas emissions at time of production. It can be beneficial to *convert* surplus renewable energy into another energy sector using e.g. Power-To-Gas (abbreviated, P2G).

From the point of view of the application domain, there is referring research available, e.g. in [2]. The high degree of freedom in terms of simultaneous location finding and device parameter estimation considered here, is not known to the author.

2 Notation

We will use graph-theoretical notation in the following, despite different terminology is common in the application domain. A *graph* is an ordered pair $G = (V, E)$ with a set of vertices (nodes) $V(G)$ and a set of edges $E(G)$. Our considered graphs are directed and finite, unless otherwise stated. For simplicity, let $[n]$ declare the set $\{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$.

3 Modeling

Let $G = (V, E)$ be a digraph that describes a set of physical energy networks. Initially, each energy sector i is represented by a subgraph $G_i \subseteq G$. It applies $G_i \cap G_j = \emptyset$ for all $i \neq j$. The graph G_i contains cycles for physical reliability purposes only. We consider discrete time steps $t \in [T]$. For simplicity, all energy resources are represented in kilowatt hours (kWh).

The set of n vertices $V(G)$ is partitioned into the disjoint union $V := A \cup B \cup S \cup N$ of producers (sources) A, consumers (sinks) B, storage devices S, and intermediate nodes N. A producer $a \in A$ offers the amount of energy $p_t^+(a) \in \mathbb{R}_0^+ \cup \{\infty\}$ at time t. The production of energy induces greenhouse gas emissions $\gamma(a) \in \mathbb{R}_0^+$ that depend on the energy flow f. A consumer $b \in B$ demands the amount of energy $p_t^-(b) \in \mathbb{R}_0^+$ at time t.

The set of m edges $E(G)$ is partitioned into the disjoint union $E := C \cup S \cup N$ of converter edges C, storage edges S and intermediate edges N. All lines of the physical networks are mapped to the set of edges of the graph G. We denote the maximum energy flow of an edge $e \in E$ as capacity $c(e) \in \mathbb{R}_0^+$. Transmission losses are approximated by an efficiency factor $\eta(e) \in [0, 1] \subseteq \mathbb{R}$. The current energy flow at time t is described by $f_t(e) \in \mathbb{R}_0^+$ [kWh].

We integrate technical devices into the network by an *a priori* definition of *candidate* locations. Storage and converter devices are modeled as depicted in Fig. 1. The actual *instantiation* of a device is controlled by a binary variable $I(e \in E) \in \mathbb{Z}_2$. The objective is to choose an optimal subset of technology instances out of all candidates. A device is instantiated, iff $I(e) \in \mathbb{Z}_2$ equals $1 :=$ True. We can reduce the number of candidates drastically by defining heuristics, e.g. a condition $\|p(u) - p(v)\|_2 < \epsilon$, with $p(u)$ and $p(v)$ the geometrical positions of $u, v \in V(G)$. The capacity $c(v)$ of a storage device is upper bound to the cumulative capacity of incident edges of v.

A set of device types is encoded into a database (matrix) for storage devices $\mathcal{S}$ and a device database (matrix) for converter devices $\mathcal{C}$, respectively:

(a) The i-th row of $\mathcal{S} = (s_{i,j}) \in \mathbb{R}^{m \times n}$ describes a storage device of type $s_i = (\eta, c_l, c_u, \gamma, \alpha, \hat{k}_0, \hat{k}_1, \tilde{k})$. For candidates that are incident with $I(e \in uv)$, we set the constraints $c_l \leq c(v) \leq c_u$, $\hat{k}(v) = \hat{k}_1 \cdot c(v) + \hat{k}_0$ and $\tilde{k}(v) = \tilde{k} \cdot f_t(e)$, with *fixed* costs $\hat{k} \in \mathbb{R}_0^+$ and *variable* costs $\tilde{k} \in \mathbb{R}_0^+$.

(b) The i-th row of $\mathcal{C} = (c_{i,j}) \in \mathbb{R}^{m \times n}$ describes a converter device of type $c_i = (\eta, c_l, c_u, \gamma, \alpha, \hat{k}_0, \hat{k}_1, \tilde{k})$. For candidates $I(e)$, we set the constraints $c_l \leq c(e) \leq c_u$, $\hat{k}(e) = \hat{k}_1 \cdot c(e) + \hat{k}_0$ and $\tilde{k}(e) = \tilde{k} \cdot f_t(e)$.

4 Definition

The objective function minimizes the total greenhouse gas emissions of all producers and devices:

$$\min \quad \left\{ \sum_{\{u \in \{A \cup S\} | e = uv\}} \left(\sum_t \gamma(u) f_t(u, v) \right) \quad + \quad \sum_{e = uv \in C} \gamma(e) f_t(u, v) \right\} \tag{1}$$

with the set of decision variables $x = \{f_t, I, s_1, s_2, \sigma_t, \eta, c, \gamma, \alpha, \hat{k}, \tilde{k}\}$. The size $n := |x|$ depends on the size of the concrete problem instance. The constraints are defined in the following:

(a) Limit the overall costs $k = f(\hat{k}, \tilde{k})$ to an upper bound $k_u \in \mathbb{R}_0^+$:

$$\sum_{v \in S} I(u, v) \left(\hat{k}(v) + \sum_t f_t(u, v) \tilde{k}(v) \right) + \sum_{e = uv \in C} I(u, v) \left(\hat{k}(e) + \sum_t f_t(u, v)\ \tilde{k}(e) \right) \ \leq \ k_u \tag{2}$$

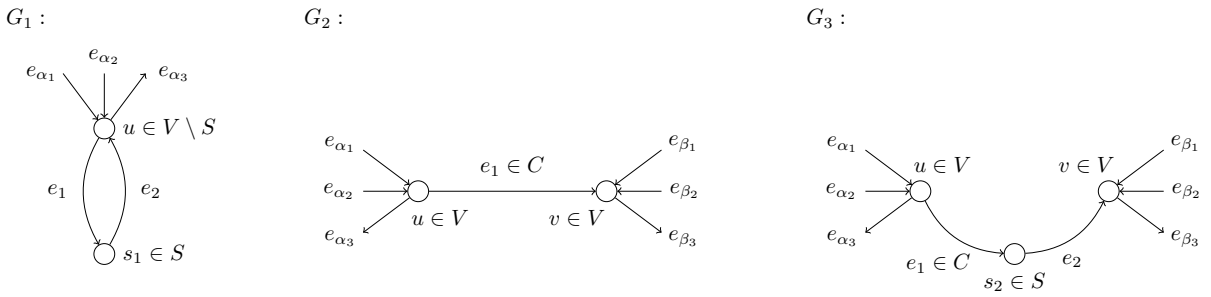


Fig. 1: G_1: Modeling of a storage device: Vertex u is part of the initial energy network. The storage *candidate* is described by edges $e_1, e_2 \in E(G)$ and a vertex $s_1 \in V(G)$; thus we apply $V(G) := V(G) \cup \{s_1\}$ and $E(G) := E(G) \cup \{e_1, e_2\}$. G_2: Depicts a simple converter device, that converts energy from node $u \in V(G)$ to node $v \in V(G)$. The energy sector for both nodes is distinct. G_3: Representation of a complex device that combines both storage and converter capabilities.

(b) The definition of constraints for the underlying network flow problem, with line efficiency $\eta \in [0,1] \subseteq \mathbb{R}$, time $t \in [T]$, and instance variables $I(e) \in \mathbb{Z}_2$ is as follows:

$$\begin{aligned}
&\sum_{e=uv \in E} f_t(u,v) = p_t^+(u) && \forall t \;\; \forall u \in A \\
&\sum_{e=uv \in E} f_t(u,v) = p_t^-(v) && \forall t \;\; \forall v \in B \\
&\sum_{e=uv \in E} \eta(u,v) f_t(u,v) \;\; - \sum_{e=uv \in E} f_t(v,u) = 0 && \forall t \;\; \forall v \in N \\
&\sum_{e=uv \in E} I(u,v)\eta(u,v) f_t(u,v) \;\; - \sum_{e=uv \in E} I(u,v) f_t(v,u) = 0 && \forall t \;\; \forall \{v \mid v \in S \vee uv \in C\}
\end{aligned} \tag{3}$$

We denote the state of charge (SoC) for storage devices $v \in S$ at time t by $\sigma_t(v)$, as well as the maximum capacity by $c(v)$. The maximum input and output flow is bound by $0 \leq f_t(e) \leq \alpha\, \sigma_t$, with some constant factor $\alpha \in [0,1] \subseteq \mathbb{R}$ to approximate the real (nonlinear) physical behavior $\sigma(t) = c(v)(1 - e^{-t/\tau})$:

$$\begin{aligned}
0 \;\leq\; f_t(u,v) \;\leq\; \alpha(v)\, \sigma_t && \forall t \;\; \forall \{v \mid uv \in E \;\wedge\; v \in S\} \\
0 \;\leq\; f_t(u,v) \;\leq\; c(u,v) && \forall t \;\; \forall uv \in E
\end{aligned} \tag{4}$$

(c) The charging and discharging behavior of a storage $s \in S$ is defined as follows:

$$\begin{aligned}
&\sigma_0(v) = 0 && && \forall v \in S \\
&\sigma_t(v) = \sigma_{t-1}(v) + I(u,v) f_t(u,v) - I(u,v) f_t(v,u) && \forall t \setminus \{0\} && \forall v \in S \\
&0 \leq \sigma_t(v) \leq c(v) && && \forall v \in S
\end{aligned} \tag{5}$$

(d) Let $s(v) \in \mathbb{Z}_2^{m(\mathcal{S})}$ be binary a vector that selects the i-th device type from database $\mathcal{S}$ for a storage device $v \in S$. Then we determine the device parameters for the instance as follows:

$$\begin{aligned}
&\textstyle\sum_i s_i(v) = 1 && && \forall v \in S \\
&\eta(v) = s(v) \cdot \mathcal{S}_{*,1}, && s(v) \cdot \mathcal{S}_{*,2} \;\leq\; c(v) \;\leq\; s(v) \cdot \mathcal{S}_{*,3} && \forall v \in S \\
&\gamma(v) = s(v) \cdot \mathcal{S}_{*,4}, && \alpha(v) = s(v) \cdot \mathcal{S}_{*,5} && \forall v \in S \\
&\hat{k}(v) = c(v)\; s(v) \cdot \mathcal{S}_{*,6} + s(v) \cdot \mathcal{S}_{*,7}, && \tilde{k}(v) = c(v)\; s(v) \cdot \mathcal{S}_{*,8} && \forall v \in S
\end{aligned} \tag{6}$$

(e) Let $s(e) \in \mathbb{Z}_2^{m(\mathcal{C})}$ be a binary vector that selects the i-th device type from database $\mathcal{C}$ for a converter device at edge $e \in E(G)$. We apply the modeling from (d), but replace all occurrences of $\mathcal{S}$ by $\mathcal{C}$ and all occurrences of v by $e \in C$.

5 Problem class

Let x be the set of n decision variables and let m be the number of constraints. All OTIMES-constraints can be transformed into the following form:

$$g_i \;:\; \sum_j \left(\alpha_j \prod_{k=1}^{\ell} y_k \right) \;\leq\; 0 \qquad \forall i \in [m], \quad \text{with} \;\; \alpha_j \in \mathbb{R} \;\; \text{and} \;\; y_k \in x := \{x_1, x_2, \ldots, x_n\} \tag{7}$$

For $\ell \geq 3$, the product $y_1 y_2 \ldots y_\ell$ can be substituted by $\tilde{y} y_{\ell-1} y_\ell$ recursively, whereby $\tilde{y} = y_1 y_2 \ldots y_{\ell-2}$ is added to the set of decision variables and $g_{m+1} \;:\; \tilde{y} - y_1 y_2 \ldots y_{\ell-2} \;=\; 0$ is added to the set of constraints. The following kinds of quadratic terms $\tilde{y} := y_1 \; y_2$ remain in OTIMES:

(a) $y_1 \in \mathbb{Z}_2$, $y_2 \in \mathbb{R}$: Assuming y_2 is bounded to $0 \leq y_2 \leq c$, with $c \in \mathbb{R}$, then $\tilde{y}$ can be substituted by $\tilde{y} \leq c \cdot y_1 \;\wedge\; \tilde{y} \leq y_2 \;\wedge\; \tilde{y} \geq 0 \;\wedge\; \tilde{y} \geq y_2 - (1 - y_1)c$. The reformulation is linear.

(b) If both $y_{1,2} \in \mathbb{R}$ are continuous, then $\tilde{y}$ cannot be linearized without approximation, and the term remains quadratic.

The problem is upper bound to $0-1$ Mixed Integer Quadratic Constrained Programming (MIQCP) and has the form:

$$\min f_0(x) \qquad \text{s.t.} \;\; f_i(x) \leq 0 \qquad \forall i \in [m] \tag{8}$$

A subset of decision variables is binary, i.e. $x_i \in \mathbb{Z}$ for all $i \in I$ with $I \subseteq [n]$. Functions f_i are defined by $f_i : \mathbb{R}^n \to \mathbb{R}$, $x \mapsto x^T P_i x + q_i^T x + r_i$ with matrices $P_i = (p_{jk}) \in \mathbb{R}^{n \times n}$, vectors $q_i \in \mathbb{R}^n$ and $r_i \in \mathbb{R}^n$. For OTIMES, the matrices P_i have the following form:

$$\begin{aligned} P_i \;&= (p_{jk}) \in \mathbb{Z}_2^{n \times n} = \begin{pmatrix} \tilde{P}_i & O \\ O & O \end{pmatrix} \\ \tilde{P}_i \;&= (\tilde{p}_{jk}) \in \mathbb{Z}_2^{\tilde{n} \times \tilde{n}} \qquad\qquad \text{with} \;\; \tilde{P}_i = \tilde{P}_i^T \;\; \text{and} \;\; \tilde{p}_{jj} = 0 \quad \forall j \in [\tilde{n}] \end{aligned} \tag{9}$$

We can approximate the ratio $\tilde{n}/n$ to $(|S| + |C|) \;/\; m(G)$. Accordingly, the quadratic part depends on the number of device *candidates*. Furthermore, submatrices $\tilde{P}_i$ are *sparse for all* $i \in [m]$. Matrices P_i are not positive definite and thus the problem as-is can not be reduced to Semi-Definite Programming (SDP).

6 Perspective

In this extended abstract, we demonstrated the modeling of OTIMES and formalized the problem definition. The problem is NP hard [1]. We will discuss the concrete problem class X, which is $0-1$ MILP $\subseteq$ X $\subseteq$ $0-1$ MIQCP in the talk. Based on the structural properties, we will present approximation approaches.

References

[1] Samuel Burer and Adam N Letchford. Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, 17(2):97–106, 2012.

[2] Akomeno Omu, Ruchi Choudhary, and Adam Boies. Distributed energy resource system optimisation using mixed integer linear programming. *Energy Policy*, 61:249–266, 2013.

This work is part of the research project ES–Flex–Infra; funded by the European Regional Development Fund (EFRE–0800106).

Hardness of k-anonymous microaggregation

Florian Thaeter[1]

[1]Institut für Theoretische Informatik, Universität zu Lübeck, 23562 Lübeck, Germany,
thaeter@tcs.uni-luebeck.de

Abstract

We show optimal k-anonymous microaggregation is NP-hard for $k \geq 26$, extending a previous result showing the NP-hardness only for $k = 3$. This lower bound already holds for 2 attributes. Our construction uses similarities between microaggregation and the k-means problem. A known reduction of Planar 3-SAT to the k-means problem is adapted to k-anonymous clustering.

1 Introduction

The problem of k-anonymous microaggregation [DF09] arises in the context of utility-preserving microdata anonymization. It describes the clustering of a database consisting of n elements with m numerical attributes into clusters of size at least k. After the clustering, all elements of a cluster are replaced with their cluster centroids in order to obtain anonymized data. The goal of k-anonymity is to prevent an attacker with some information about an element of the database to obtain full information about his target. The privacy of elements is increased, because information on specific elements is limited by hiding each one in a group of k identical-appearing elements. In order not to destroy the quality of the data one would like to generate as little distortion as possible.

Definition 1. ***(k-anonymous microaggregation problem)*** *Let $d(\cdot,\cdot)^2$ be the squared Euclidean distance between two m-dimensional vectors and $c(\mathbf{x}_i) = {}^1/_{|C(\mathbf{x}_i)|} \cdot \sum_{\mathbf{x} \in C(\mathbf{x}_i)} \mathbf{x}$ the centroid of the cluster $C(\mathbf{x}_i)$ of $\mathbf{x}_i$ in a partitioning $\mathcal{C}$. Given a sequence of n vectors $\mathcal{X} = \mathbf{x}_1, \ldots, \mathbf{x}_n$ of dimension m, i.e. $\mathbf{x}_i \in \mathbb{R}^m$, find a partitioning $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ of $\mathcal{X}$ with centroids $c(C_i)$ such that*

$$\forall i \in \{1, \ldots, \ell\} : |C_i| \geq k, \text{ and } \mathit{Cost}(\mathcal{C}) := \sum_{i=1}^{n} d\left(\mathbf{x}_i, c(\mathbf{x}_i)\right)^2 \text{ is minimized.}$$

In a multiset-respecting clustering all elements $\mathbf{x}_i$, $\mathbf{x}_j$ with identical attributes are clustered together.

The k-anonymous microaggregation problem in a way is dual to the k-means problem. Whereas the k-means problem asks for a clustering into at most k clusters of arbitrary size, the k-anonymous microaggregation seeks to cluster elements in an arbitrary amount of clusters with each having at least k elements. Similar to the k-means problem [Llo82], we measure the distortion by the sum of squared errors (SSE) between elements and their cluster representatives. Optimal k-anonymous microaggregation is achieved if the SSE of a clustering is minimal over all k-anonymous clusterings.

In 2001, Oganian and Domingo-Ferrer claimed that the anonymization problem is NP-hard [ODF01]. However, they only give a proof for $k = 3$ and their reduction does not work for other values of k. The most we could see is that allowing arbitrary duplications of vectors the proof can be extended to multiples of 3. For k-anonymization we do not know of any technique to translate a hardness result for some value of the parameter k to larger values contrary to most classical

decision or optimization problems. On the one hand, this problem seems intuitively more difficult if the minimal size of a cluster is increased, on the other hand having to deal with less clusters may make things easier. Thus, the status for most values of k is still open. In this paper we will take another approach and are able to show that the problem is NP-hard for all values of $k \geq 26$.

2 From Planar 3-SAT to k-anonymous microaggregation

Our reduction is inspired by a reduction from Planar 3-SAT to optimal Planar k-means presented by Mahajan, Nimbhorkar and Varadarajan in 2009 [MNV09]. We use a slightly modified reduction function to take into consideration the differences between k-means and k-anonymous clustering.

Definition 2. ***(Planar 3-SAT problem)*** *Let F be a 3-CNF formula with variables $\{v_1, \ldots, v_n\}$ and clauses $\{c_1, \ldots, c_m\}$. We call $G(F) := (V, E)$ the (undirected) graph of F, where*

$$V = \{v_i \mid 1 \leq i \leq n\} \cup \{c_j \mid 1 \leq j \leq m\}$$
$$E = E_1 \cup E_2 \text{ where}$$
$$E_1 = \{(v_i, c_j) \mid v_i \in c_j \text{ or } \overline{v}_i \in c_j\}$$
$$E_2 = \{(v_j, v_{j+1}) \mid 1 \leq j < n\} \cup \{(v_n, v_1)\}.$$

If $G(F)$ is a planar graph, F is called a planar 3-CNF formula. The planar 3-SAT problem is to determine whether a given planar 3-CNF formula F is satisfiable.

The Planar 3-SAT problem has been shown to be NP-hard by Lichtenstein in 1982 [Lic82].

Theorem. *The optimal k-anonymous microaggregation problem is NP-hard for $k \geq 26$.*

The theorem is proven by creating a database $\mathcal{X}$ out of a Planar 3-SAT instance F. To show correctness, it is necessary to argue that the constructed microaggregation instance has an optimal solution $\mathcal{C}$ with $Cost(\mathcal{C})$ not higher than a predefined threshold if and only if F is satisfiable.

2.1 Construction

Let F be a Planar 3-SAT instance with n variables v_i and m clauses c_j. We use the graph $G(F)$ to construct a 2-dimensional instance $(\mathcal{X}, k)$ for the k-anonymous microaggregation problem.

Consider $G(F)$ with all E_2 edges removed, i.e. $G'(F) = (V, E_1)$.

1. Compute a planar embedding $\mathcal{E}$ of $G'(F)$ and for each vertex, assign numbers κ to incident edges according to a cyclic ordering.

2. Replace every variable vertex v_i by a cyclic subgraph with m vertices $v_i^1, \ldots, v_i^m$. Reroute all edges e_κ incident to v_i to v_i^κ. Observe that it is still possible to obtain a planar embedding $\mathcal{E}'$ of the resulting graph $G''(F)$.

3. Place every vertex of $G''(F)$ on an integer grid and route edges along grid lines.

4. Inflate the grid by a factor of $b \geq 14$, which guarantees that every vertex or bend point u of an edge has no other bend point or vertex inside a box B_u of size $b \times b$ grid lengths and inside a smaller box S_u of size 6×6 grid lengths.

5. Replace every edge between a variable vertex v_i^κ and a clause vertex c_j by a pair of parallel rectilinear paths separated by two grid squares. At the clause vertex end, connect both paths by a straight line on the border of S_{c_j} and on the variable vertex end, connect one path with the edge to $v_i^{\kappa-1}$ and the other path with the edge to $v_i^{\kappa+1}$ to form a continuous circuit s_i.

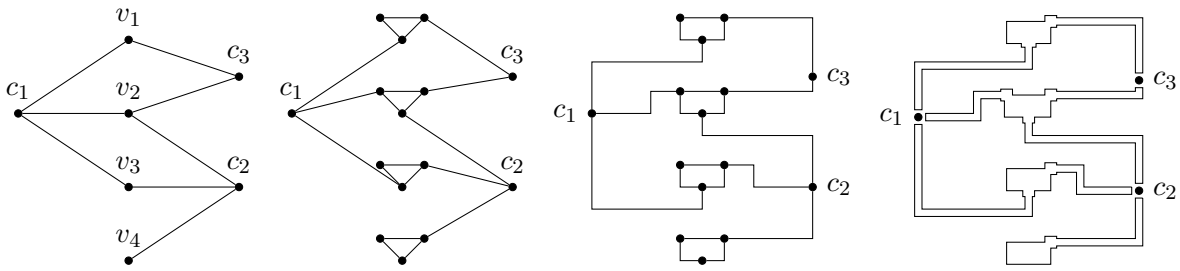


Figure 1: Step 1, 2, 3 and 5 of the construction for $F = (\overline{v}_1 \vee v_2 \vee v_3) \wedge (v_2 \vee \overline{v}_3 \vee v_4) \wedge (v_1 \vee \overline{v}_2)$

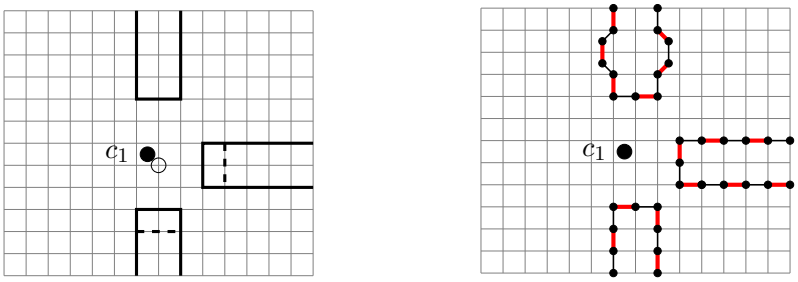


Figure 2: Step 6 and 7 of the construction around vertex c_1. Edges of true matchings are highlighted.

6. Move every clause vertex c_j to the center of the north-west grid square touching it. Extend all circuits incident to a clause vertex c_j so that they all are in distance $5/2$ grid lengths to c_j. Observe that all circuits are of even length and do not intersect.

7. There are two possible perfect matchings on the grid points on any circuit s_i. Fix one of them and call it *true matching* from now on (the other one is subsequently called *false matching*). If a clause c_j contains a variable v_i in non-negated form, and the clause vertex c_j is closer to a true matching edge of s_i, nothing needs to be changed. This holds analogously for negated variable and false matching edge. If however a matching is not according to the sign of a variable in a clause, the variable circuit is modified near the clause vertex, to correct this.

8. To conclude the construction, database elements are created out of the resulting structure: Let β be the squared grid length. Around every circuit at squared distance β from each other, place M database elements at the same position, so that these points are placed on top of the vertices used for the circuit matchings. Note that there is an even number of such vertices and therefore an even number of element positions on any circuit. Clause vertices are represented by single elements. The anonymization parameter k is set to be $2M$.

2.2 Correctness

We can observe the squared distance between two adjacent circuit points is β and between any two non-adjacent circuit points is at least 2β. The squared distance between a circuit and a clause vertex is $\alpha := (5/2)^2\beta$. The squared distance between a clause vertex and both endpoints of a circuit's edge closest to it is $\alpha + \beta/4$, all other circuit points are at least at squared distance $\alpha + 5/4\beta$ from the circuit vertex. Clause vertices have at least squared distance $\theta := b^2\beta$ from each other.

Lemma 3. *Consider a database without clause points. Let ℓ be half the number of circuit points. An optimal $2M$-anonymous clustering has ℓ clusters and any cluster contains all elements from two adjacent circuit points. There is a minimal distance between optimal and non-optimal clustering:*

1. *Clustering the circuit points into consecutive pairs has cost $\frac{\ell M\beta}{2}$.*
2. *Any other clustering of circuit points has cost at least $\frac{\ell M\beta}{2} + \frac{M\beta}{2}$.*

Proof of lemma 3 (sketch). Statement 1. can be easily verified. To show that any other clustering has costs as stated in 2. we need to consider several cases: (a) The biggest cluster contains elements from 2 circuit points. (b) The biggest cluster contains elements from 3 circuit points and is multiset-respecting. (c) The biggest cluster contains elements from 3 circuit points and is not multiset-respecting. (d) The biggest cluster contains elements from more than 3 circuit points. By using the minimal cluster size of $k = 2M$ it can be shown, that in every case, the cost is indeed as stated. □

Lemma 4. *A formula is satisfiable if and only if the associated database has a k-anonymous clustering of cost at most $\frac{\ell M\beta}{2} + \frac{2M}{2M+1}\alpha m$.*

Proof of lemma 4 (sketch). $\Rightarrow$: If the formula is satisfiable, every clause vertex can be clustered with two circuit points nearby. $\Leftarrow$: Uses the result of lemma 3. Holds if $M > 1/2(25m-1)$, resulting in a minimum M of 13 and therefore a minimum k of 26. □

So far the construction can handle only instances with even $k \geq 26$. It is however possible to adjust the construction, so that on any circuit, points with multiplicity M and $M + 1$ are alternating. Together with $k = 2M + 1$ and analogous calculations to those in lemma 3 and lemma 4 it is possible to extend the claim to *odd* values ≥ 26.

3 Conclusion

An obvious question is what about smaller values of k. It seems unlikely that for values of $4 \leq k \leq 25$ anonymization becomes easier. We tried to tune this construction, but geometrical constraints seem to prevent further decrease. Is there another hard problem more suited for the reduction?

References

[DF09] Josep Domingo-Ferrer. *Encyclopedia of Database Systems*, chapter Microaggregation, pages 1736–1737. Springer US, 2009.

[Lic82] David Lichtenstein. Planar formulae and their uses. *SIAM journal on computing*, 11(2):329–343, 1982.

[Llo82] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[MNV09] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *International Workshop on Algorithms and Computation*, pages 274–285. Springer, 2009.

[ODF01] Anna Oganian and Josep Domingo-Ferrer. On the complexity of optimal microaggregation for statistical disclosure control. *Statistical Journal of the United Nations Economic Commission for Europe*, 18(4):345–353, 2001.

Exact Solution Approaches to Competitive Hub Location Problem with Attraction Function

Richa Tiwari[1], Sachin Jayaswal[1], and Ankur Sinha[1]

[1]IIM Ahmedabad, Vastrapur, Gujarat, India, 380015, richat@iima.ac.in

Abstract

In this paper, we study hub location problems in the presence of competition caused by the presence of already existing airlines in the network. For this, we model the market share captured by the entrant airline as a proportional gravity based attraction function. This leads to a non-linear integer program, for which we propose four customised exact solution methods. From our extensive computational experiments using two of the publicly available data-sets, namely the Civil Aeronautics Board (CAB) data-set and the Australian Post (AP) data-set, we suggest the method which performs the best in terms of computation time. We further provide insights into the computational performance of the four methods.

1 Solution Methods

Through this paper, we make the following contributions to the literature on hub location problems. To the best of our knowledge, our paper is the first to solve a competitive hub location problem with attraction functions exactly. We propose four different exact approaches for the problem, based on the model proposed by [Eiselt and Marianov(2009)]. Since this model is a non-linear IP, it can't be solved using an off-the-shelve MIP solver like CPLEX.

The first approach is based on approximating the non-linear terms using supporting hyper-planes, which are generated as needed. This method is called a constraint generation method . We also provide a proof of the finiteness of this method along the lines of [Elhedhli(2005)]. We will refer to this method as CPA. While solving many of the instances of the problem, the system was running out of memory. To solve this issue, we reformulated the problem as an MISOCP (Mixed Integer Second order conic program) which is non-trivial in itself. MISOCPs are a special class of non-linear problems that can be handled directly using solvers like CPLEX and GUROBI. To improve the computational results even further, we propose the third and the fourth method. These methods decompose the problem into two sub-problems, one of which is an IP and the other is a non-linear program with continuous variables. The IP is solved exactly using a sorting algorithm and the other sub-problem can be solved using two approaches which gives rise to the third and fourth approach. This type of decomposition has advantages as the network size increases. The third method is referred to as LR-SOCP since we solve one of the sub-problems using SOCP. The fourth method is referred to as LR-CPA since we solve one of the sub-problems using aforementioned CPA.

2 Results and Conclusion

We compare the computational time for all the four proposed methods within a time frame of 2 hours. We find that using CPA we are able to solve 76 % of the 120 benchmark instances to

optimality at 1% optimality gap . Also we were are to solve 81% of the instances to optimality by reformulating the problem as MISOCP. This percentage significantly increases to 95% and 88 % for LR-SOCP and LR-CPA respectively.
We subsequently show, through rigorous experimentation that LR-SOCP and MISOCP are the fastest of the four methods. Also for smaller instances of upto network size of 20 nodes, MISOCP has the best computational performance. As the network size increases, LR-SOCP performs better. This is expected of any lagrangian relaxation based method. We also provide insights into which method is performs better if the practical requirement is to close optimality gap at 2%. From the numerical experiments across various network sizes, it is clear that the LR-SOCP reaches optimality gap of 2% from 5% at a very fast rate but starts tailing to reach to a gap of 1%. As opposed to this, LR-CPA steadily approaches optimality gap of 5% to 2% and then 1%. With these results we conclude our work.

References

[Eiselt and Marianov(2009)] Eiselt, H. A., Marianov, V., 2009. A conditional p-hub location problem with attraction functions. Computers & Operations Research 36 (12), 3128–3135.

[Elhedhli(2005)] Elhedhli, S., 2005. Exact solution of a class of nonlinear knapsack problems. Operations research letters 33 (6), 615–624.

Graphical Methods for Finding Instrumental Variables

Benito van der Zander[1], Johannes Textor[2], and Maciej Liśkiewicz[1]

[1]Universität zu Lübeck, Germany
[2]Radboud University Nijmegen, The Netherlands

Abstract

Instrumental variables (IVs) are a popular approach to identify causal effects. For valid inference, IVs must not be direct causes of any variable in the model except the explanatory variable X. Such variables do not exist in many model instances, so the approach has been generalized to *conditional* IVs. However, a barrier for application of this method is of algorithmic nature: So far, it was not clear whether such conditional IVs can be tested and found efficiently. We prove that it is indeed an NP-complete problem to test if a given variable is a conditional IV. However, if the covariates are restricted to ancestors, this test can be performed in linear time. This implies a new definition of IVs, which we term ancestral IVs. It turns out that an ancestral IV exists if and only if a conditional IV exists in a graph. We use this definition to obtain efficient algorithms to find conditional IVs.

Introduction. Discovering and quantifying causal relationships is a key goal of empirical sciences. Analyzing causes of diseases, economic crises and other complex phenomena is of great social and economic importance. However, for ethical or economic reasons questions such as "Does smoking cause lung cancer?" or "What are the major causes of economic crises?" can be difficult to examine through direct experimentation. On the other hand, there are often available large amounts of observed data that can provide relevant information about these issues.

Causal inference is a rapidly growing field involving mathematical statistics, machine learning and some sub-fields of artificial intelligence and computer science. The goal is to explore from observed data and phenomena the causal relationships between different objects and actions, like e.g., between medical treatment and recovery. The theory of causality allows a rigorous characterization of the circumstances under which, given sufficient knowledge about a system, experimental data can be substituted by observational data for causal inference [4]. In this paper, we study algorithmic and complexity aspects involving instrumental variables (IVs) a widely used approach to infer cause-effect relationships [1, 2, 3]. We mostly present our results from [5] and some extensions improving time complexities of our algorithms.

Graphical Concepts. Graphical causal models represent a data-generating causal process as a graph, most commonly as a directed acyclic graph (DAG) of n nodes and m edges. Every node in the DAG corresponds to a random variable and every edge $X \to Y$ represents a direct causal effect of X on Y. The concept of d-*separation* links conditional independences between these random variables to paths in the DAG. Let $An(\mathbf{W})$ denote all ancestors of nodes $\mathbf{W}$. A node C on a path π is called a *collider* if two arrowheads of π meet at C, i.e., if π contains $X \to C \leftarrow Y$. A collider C is *open* given a set $\mathbf{W}$ if C is in $\mathbf{W} \cup An(\mathbf{W})$. A path π is d-*connecting* given a set $\mathbf{W}$, if every collider on π is open given $\mathbf{W}$ and every non-collider is not in $\mathbf{W}$. Two nodes X, Y are called d-*connected* by a set $\mathbf{W}$ if there is a d-connecting path π between them. If X, Y are not d-connected by $\mathbf{W}$, $\mathbf{W}$ d-*separates* them. A path that is not d-connecting is *blocked*. When the data is consistent with the graphical model, variables X, Y in the data are conditionally independent given a set of variables $\mathbf{W}$, if nodes X, Y are d-separated given nodes $\mathbf{W}$ [4].

The Identification Problem. In a structural equation model (SEM) it is assumed that the random variables influence each other linearly, i.e., the expected value of each variable is a linear function of some other

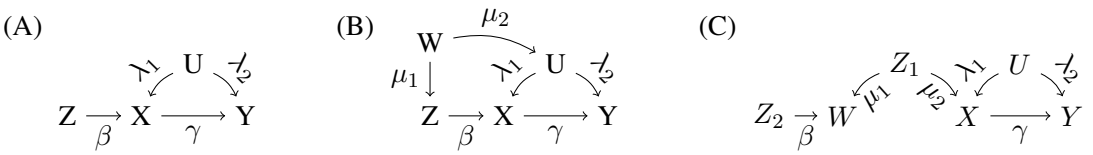


Figure 1: (A) The classic IV model. (B) Z is not an IV, but is a *conditional instrument* given W. (C) Z_1 is an IV and Z_2 is a CIV given W but not an *ancestral* IV. In all cases, U is treated as unobserved variable.

variables, namely of its parents in the DAGs. Every edge is also given a weight, the *direct causal effect*. We assume that all variables are normally distributed and normalized to have unit variance. Moreover, we distinguish variables that are considered as *observed* (measured; we denote the set of these variables as $\mathbf{M}$) and such that are *unobserved* (meaning they are absent from the data or cannot be measured). Figure 1A represents example causal equations $Z = \epsilon_Z$, $X = \beta Z + \lambda_1 U + \epsilon_X$, and $Y = \gamma X + \lambda_2 U + \epsilon_Y$, with random, normally, independently distributed error terms ϵ_X, ϵ_Y, and ϵ_Z, assuming Z, X, Y are observed and U is an unobserved variable. A unit change of Z, for example, changes the value of X by β. However, any change of X leaves the expected value of Z unchanged, since X is not a cause or ancestor of Z.

The *identification problem* in SEMs consists in recovering the weights (i.e., direct causal effects) between observed variables given the graphical structure of the model and data for observed variables. For a SEM in Fig. 1A the task would be to estimate β and γ. This problem plays a fundamental role in the theory and practice of SEMs. However, though some partial solutions are given, in general, the problem remains still unsolved. A widely used approach to identify parameters in SEMs is based on IVs.

The IV-based Method. If we know the causal effects at each edge, we can easily calculate the correlations between all random variables from the model. Assuming all variables are normalized to variance 1, the correlation (covariance) $\mathrm{Cov}(Z, Y)$ between two variables Z, Y is equal to $\sum_\pi \prod_{e \in \pi} ce(e)$ where $ce(e)$ is the causal effect of edge e, the product goes over every edge e on path π and the sum goes over every path π between Z and Y that is d-connected. In the example Fig. 1A there is exactly one open path $Z \to X \to Y$ between Z and Y, so the correlation $\mathrm{Cov}(Z, Y)$ is $\beta\gamma$. Analogously, $\mathrm{Cov}(Z, X) = \beta$ and $\mathrm{Cov}(X, Y) = \gamma + \lambda_1\lambda_2$. In Fig. 1B we have $\mathrm{Cov}(Z, Y) = \beta\gamma + \mu_1\mu_2(\lambda_1\gamma + \lambda_2)$. By regressing Y on Z, W we can also identify the product $\beta\gamma$.

On the other hand, if we know the correlations between variables in $\mathbf{M}$ and the (presence and directions) of edges, but not the causal effects of the edges, we can calculate the causal effects from the correlations (if the model is true). Such identities are of great importance since correlations are readily provided by observed data. In theory one can construct polynomial equation systems that link the causal effects to the correlations and solve these for the causal effects. However, in practice the resulting equation systems can only be solved for trivially small models. Nevertheless, in certain situations one can find a simple expression. For example in Fig. 1A the causal effect of Z on X is easily obtained as $\beta = \mathrm{Cov}(Z, X)$. Knowing that $\mathrm{Cov}(Z, Y) = \beta\gamma$, we can estimate the causal effect $\gamma = \beta\gamma/\beta = \mathrm{Cov}(Y, Z)/\mathrm{Cov}(X, Z)$. In Fig. 1B one can use $\gamma = \beta\gamma/\beta$, where both β and $\beta\gamma$ can be obtained by regression, and in Fig. 1C similarly $\gamma = \beta\mu_1\mu_2\gamma/\mu_1\mu_2\beta$.

The graphical language used in causal modeling allows one to express sufficient conditions for a causal effect of X on Y that can be obtained by this method from the covariance matrix in an elegant way as (a) Z is d-connected to X by $\mathbf{W}$, (b) Z is d-separated from Y in $\mathcal{G}_c = \mathcal{G} \setminus (X \to Y)$ by $\mathbf{W}$, and (c) $\mathbf{W}$ consists of non-descendants of Y. An additional constraint is that not all variables in the DAG are observed, so one does not know the correlations between every pair of variables. We assume correlations are known for variables in the set $\mathbf{M}$ of observed variables thus we require $X, Y, Z \in \mathbf{M}$, and $\mathbf{W} \subseteq \mathbf{M}$. If Z fulfills these conditions, Z is called a *conditional instrument* (CIV) (relative to $X \to Y$). If $\mathbf{W} = \emptyset$, Z is just an IV. If $\mathbf{W} \subseteq An(Y, Z)$, we call Z an *ancestral instrument* (AIV) [5]. Fig. 1 (A) shows an IV, (B) an AIV and (C) a CIV. Of course every IV is an AIV and every AIV is a CIV. Main results of this paper are the following:

Theorem 1. *Determining if, for given $X, Y, Z \in \mathbf{M}$, node Z is a conditional IV relative to $X \to Y$ is an NP-complete problem.*

Theorem 2. *Determining if, for given $X, Y, Z \in \mathbf{M}$, node Z is an ancestral IV relative to $X \to Y$ can be done in $\mathcal{O}(n+m)$ time.*

Theorem 3. *For given variables $X, Y \in \mathbf{M}$, an ancestral IV $Z \in \mathbf{M}$ relative to $X \to Y$ exists iff a conditional IV $Z' \in \mathbf{M}$ relative to $X \to Y$ exists.*

Corollary 4. *Finding, for given $X, Y \in \mathbf{M}$, a conditional IV Z relative to $X \to Y$ or verifying that no conditional IV exists can be done in $\mathcal{O}(n(n+m))$ time.*

Theorem 1 is proved by reducing 3SAT to the testing of a CIV using the construction of Fig. 2. The only observed nodes are the V nodes in rows 2 to 4, so only these nodes might occur in $\mathbf{W} \subseteq \mathbf{M}$. Z is a CIV iff there is a d-connected path between Z and X, and no d-connected path between Z and Y.

Every clause of the 3SAT instance corresponds to a collider C_i on the path between X and Z at the top. Each clause-node has three children V_j^k (or $\overline{V}_j^k$) corresponding to literals x_j ($\overline{x}_j$) in the clause with the index j representing the variable and k an occurrence indicator (so the number of V nodes in the 2nd row of Fig. 2 is exactly three times the number of clauses). The path between X and Z is d-connected given $\mathbf{W}$ if and only if each collider C_i has a child in $\mathbf{W}$. Thus to make Z a CIV, one needs to choose nodes V_j^k, $\overline{V}_j^k$ to include in $\mathbf{W}$, which corresponds to choosing a literal in each clause to set to true in the 3SAT instance.

The graphical structure below the 2nd row ensures that this choice is consistent, i.e., one cannot include both V_j^k and $\overline{V}_j^{k'}$ in $\mathbf{W}$. For each variable there is a path $Y' \to P_j \leftarrow F_j \to N_j \leftarrow Y''$, such that collider P_j is an ancestor of all positive V_j^k nodes and N_j an ancestor of all negative $\overline{V}_j^{k'}$ nodes. Choosing both V_j^k and $\overline{V}_j^{k'}$ for inclusion in $\mathbf{W}$ would open these two colliders and open the d-connected path $Y \leftarrow Y'' \ldots Y' \to C_1' \ldots Z$ between Y and Z, preventing Z from being a CIV.

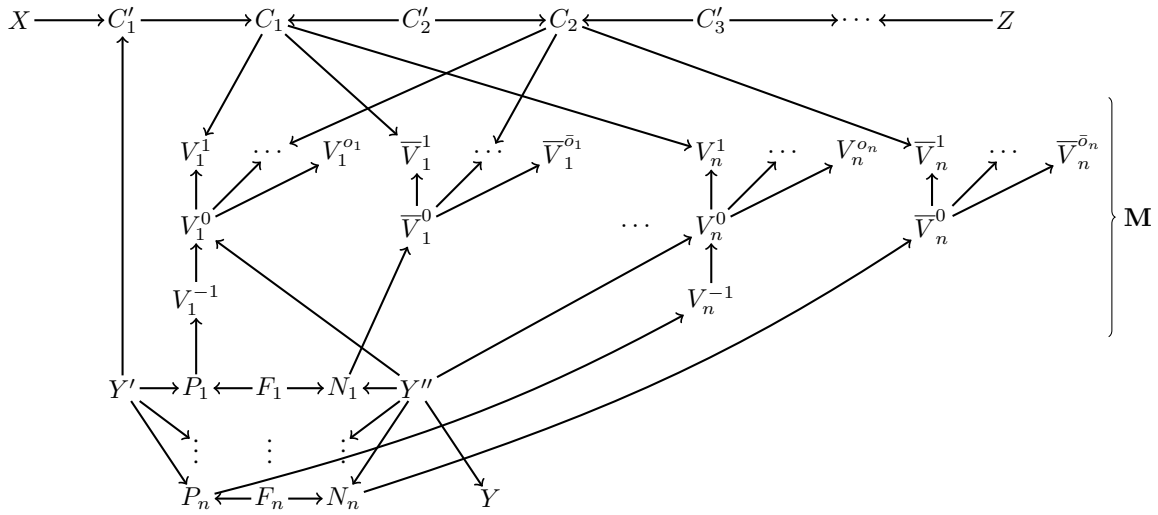


Figure 2: Reduction of 3SAT to the instrumentalization problem (edge $X \to Y$ is omitted). Each variable C_i stands for a clause in the input formula. Here n stands for the number of variables in the 3SAT instance.

We show Theorem 2 with an efficient algorithm that finds the set $\mathbf{W}$ for an AIV under the constraint $\mathbf{W} \subseteq An(Y, Z)$ in $\mathcal{O}(n+m)$. First note that we can ignore X and the condition that $\mathbf{W}$ should not d-separate X and Z, if we choose $\mathbf{W}$ as small as possible, such that any other set d-separates more nodes.

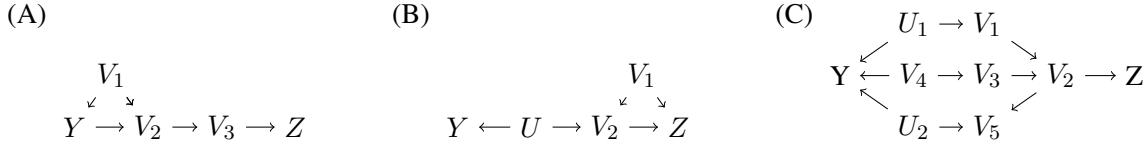


Figure 3: Three DAGs with unobserved variables $\{U, U_1, U_2\}$. The nearest separators for Y and Z are $\{V_2\}$ and $\{V_1, V_2\}$ in (A), $\{V_1, V_2\}$ in (B) and $\{V_1, V_4\}$ in (C).

A set $\mathbf{W} \subseteq \mathbf{M} \setminus \{Y, Z\}$ is called a *nearest separator* if (a) $\mathbf{W}$ d-separates Y and Z, and (b) for any $W \in \mathbf{W}$ and any set $\mathbf{W}' \subseteq \mathbf{M} \setminus \{W, Y, Z\}$ it holds: if $\mathbf{W}'$ does not d-separate W and Z, then $\mathbf{W}'$ does not d-separate Y and Z. For example in Fig. 3A (the only) nearest separators are the sets $\{V_2\}$ and $\{V_1, V_2\}$. V_1 can be included, because any d-connecting path from Z to V_1 can be extended to a d-connecting path to Y, but is unnecessary, since such a path is already blocked at V_2. In Fig. 3B the only nearest separator is $\{V_1, V_2\}$. Since node U is unobserved, the bottom path is d-connecting without V_2, but V_2 alone d-connects the path through V_1. Similarly in (C) the nearest separator is $\{V_1, V_4\}$. V_5 must not be included, since it is a collider that would lead to more d-connecting paths.

In principle a nearest separator can be found by a greedy algorithm that searches a (d-connecting) path from Y to Z and adds the first node of the path to $\mathbf{W}$, until $\mathbf{W}$ d-separates Y and Z. Two complications arise, first unobserved nodes cannot be added to $\mathbf{W}$, and secondly, when a collider from the path is added to $\mathbf{W}$, the path remains d-connecting. It can be shown that this is fine. One can always use the first observed node not yet in $\mathbf{W}$ and the colliders will help to block other paths, e.g., like V_2 on the path $Y \leftarrow U \to V_2 \leftarrow V_1 \to Z$ in Fig. 3B. The nearest separator can also be found in linear time $\mathcal{O}(n + m)$ by a reachability search that starts at Y, stops on observed non-colliders and continues through unobserved nodes or colliders in $An(Y, Z)$. All (observed) nodes reached by this search form a nearest separator.

Once $\mathbf{W}$ has been found it is easy to verify, whether it satisfies the conditions of an AIV.

To prove Theorem 3 we start with a CIV Z that is not an AIV and search another node in the DAG that is an AIV. It is a well-known fact of causal models that if there exists any set that d-separates Y and Z, there exists a subset of $An(Y, Z)$ that d-separates Y and Z. Thus the purpose of the additional nodes $\mathbf{W} \setminus An(Y, Z)$ is not to d-separate Y and Z, but to d-connect X and Z, i.e., to open a collider C on a path between X and Z. This collider is clearly d-connected to X. C might be an AIV, but it might also be unobserved. But the node $W \in \mathbf{W}$ that opens C is observed, and d-connected to X given $\mathbf{W} \setminus W$. It can be shown that at least one such W is also d-separated from Y and, when $\mathbf{W}$ is minimal, an AIV.

Conclusion. We can find an AIV or CIV Z for given X, Y in time $\mathcal{O}(n(n + m))$. However, testing if a given Z is a CIV is NP-complete, which is surprising, since finding is usually harder than testing.

Acknowledgment. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) grant LI 634/4-2.

References

[1] J. D. Angrist, G. W. Imbens, and D. B. Rubin. Identification of causal effects using instrumental variables. *Journal of the American Statistical Association*, 91(434):444–455, 1996.

[2] S. Greenland. An introduction to instrumental variables for epidemiologists. *Int J Epidemiol*, 29(4):722–729, Aug 2000.

[3] G. Imbens. Instrumental variables: An econometrician's perspective. *Statistical Science*, 29(3):323–358, 2014.

[4] J. Pearl. *Causality*. Cambridge University Press, 2009.

[5] B. van der Zander, J. Textor, and M. Liśkiewicz. Efficiently finding conditional instruments for causal inference. In *Proc. Int. Joint Conference on Artificial Intelligence (IJCAI)*, pages 3243–3249, 2015.

Toughness and forbidden subgraphs for hamiltonian-connected graphs

Wei Zheng[1,2], Hajo Broersma[1], and Ligong Wang[2]

[1]Faculty of EEMCS, University of Twente, Enschede, The Netherlands
[2]Department of Applied Mathematics, Northwestern Polytechnical University, Xi'an, China

Abstract

A graph G is called hamiltonian-connected if for every pair of distinct vertices $\{u, v\}$ of G there exists a Hamilton path in G that connects u and v. A graph G is said to be t-tough if $t \cdot \omega(G-X) \leq |X|$ for all $X \subseteq V(G)$ with $\omega(G-X) > 1$. The toughness of G, denoted $\tau(G)$, is the maximum value of t such that G is t-tough (taking $\tau(K_n) = \infty$ for all $n \geq 1$). It is known that a hamiltonian-connected graph G has toughness $\tau(G) > 1$, but that the reverse statement does not hold in general. In this presentation, we investigate all possible forbidden subgraphs H such that every H-free graph G with $\tau(G) > 1$ is hamiltonian-connected. Except for one open case $H = K_1 \cup P_4$, we characterize all possible graphs H with this property.

1 Introduction

We use standard graph terminology and notation adopted from the textbook [4], and consider simple graphs only. Let G be a graph with vertex set $V(G)$. For a given graph H, we say G is *H-free* if G does not contain an induced copy of H. Let $\omega(G)$ denote the number of components of the graph G. As introduced by Chvátal in [7], we say that a connected graph G is *t-tough* if $t \cdot \omega(G-X) \leq |X|$ for all $X \subseteq V(G)$ with $\omega(G-X) > 1$. The *toughness* of G, denoted $\tau(G)$, is the maximum value of t such that G is t-tough (taking $\tau(K_n) = \infty$ for all $n \geq 1$).

A cycle in a graph G is called a *Hamilton cycle* if it contains all vertices of G, and a graph is said to be *hamiltonian* if it contains a Hamilton cycle. A *Hamilton path* in a graph G is a path that contains all vertices of G, and a graph G is *hamiltonian-connected* if every pair of vertices of G occurs as the end vertices of a Hamilton path of G. It is easy to verify and a well-known fact that a hamiltonian graph is 1-tough, and that a hamiltonian-connected graph has toughness strictly larger than one. It is also known that the reverse statements do not hold, i.e., there exist infinitely many nonhamiltonian 1-tough graphs, and there exist infinitely many graphs with toughness strictly larger than one that are not hamiltonian-connected. It is natural and interesting to investigate under which additional conditions the reverse statements do hold. In other words, under which additional conditions are the properties of being 1-tough and being hamiltonian equivalent, and similarly for the stronger properties of having toughness strictly larger than one and being hamiltonian-connected. The type of additional conditions we focus on here are forbidden subgraph conditions. For hamiltonicity this type of problem was addressed by the authors of [10].

In [10], the aim was to characterize all possible graphs H such that every 1-tough H-free graph is hamiltonian. The almost complete answer was given there by the conclusion that every proper induced subgraph of $K_1 \cup P_4$ (the graph on five vertices consisting of a path on four vertices and an additional vertex with degree 0) can act as a forbidden subgraph to ensure that every 1-tough

graph is hamiltonian, and that there is no other forbidden subgraph with this property, except possibly for the graph $K_1 \cup P_4$ itself. This was left as an open case for hamiltonicity, and it seems to be a very hard case. To date it is even unknown whether there exists some constant t such that every t-tough $K_1 \cup P_4$-free graph is hamiltonian.

Instead of researching this open case, we consider the stronger property of being hamiltonian-connected under the same additional forbidden subgraph conditions, assuming the toughness to be strictly larger than one. We find that the results are completely analogous to the hamiltonian case: every graph H such that any 1-tough H-free graph is hamiltonian also ensures that every H-free graph with toughness larger than one is hamiltonian-connected. And similarly, there is no other forbidden subgraph having this property except possibly for the open case $K_1 \cup P_4$.

2 Preliminaries

Since its introduction by Chvátal [7] in the 1970s, the toughness notion has received a lot of attention, mainly inspired by Chvátal's Conjecture which states that there exists a constant t_0 such that every t_0-tough graph on $n \geq 3$ vertices is hamiltonian. This conjecture is still wide open, although many results have been obtained since, inspired by the results and open problems in [7]. For several years, it was believed that $t_0 = 2$ should be the correct value, because this would imply several other conjectures, including the conjecture in [11] that every 4-connected claw-free graph is hamiltonian. Since 2000 we know that $t_0 \geq 9/4$, because in [3] the authors constructed an infinite family of nonhamiltonian graphs with toughness arbitrarily close to 9/4 from below. The survey paper [2] deals with a large number of results related to toughness that have been established until more than ten years ago. A more recent survey of results and open problems appeared a few years ago [5].

Because of the difficulty of proving or refuting Chvátal's Conjecture, researchers considered the toughness condition restricted to several different classes of graphs. In some cases, the necessary condition of being 1-tough turns out to be also sufficient for hamiltonicity, as in the following result. Here a path on k vertices is denoted by P_k, a complete graph on k vertices by K_k, and we use $G \cup H$ to denote the disjoint union of two disjoint graphs G and H, and we use kG to denote the graph consisting of k disjoint copies of the graph G.

Theorem 1 ([10]). *Let R be an induced subgraph of P_4, $K_1 \cup P_3$ or $2K_1 \cup K_2$. Then every R-free 1-tough graph on at least three vertices is hamiltonian.*

Note that every induced subgraph of P_4, $K_1 \cup P_3$ or $2K_1 \cup K_2$ is also an induced subgraph of $K_1 \cup P_4$, and that $K_1 \cup P_4$ is the only induced subgraph of $K_1 \cup P_4$ that is not an induced subgraph of P_4, $K_1 \cup P_3$ or $2K_1 \cup K_2$. The following complementary result shows that there is no graph H that can ensure every 1-tough H-free graph is hamiltonian other than the induced subgraphs of $K_1 \cup P_4$.

Theorem 2 ([10]). *Let R be a graph on at least three vertices. If every R-free 1-tough graph on at least three vertices is hamiltonian, then R is an induced subgraph of $K_1 \cup P_4$.*

The two theorems together clearly leave $K_1 \cup P_4$ as the only open case in characterizing all the graphs H such that every H-free 1-tough graph is hamiltonian.

A hamiltonian graph is 1-tough, and hence 2-connected, so a hamiltonian-connected graph on at least three vertices is also 2-connected. It is even clearly 3-connected: if there exists a cut set $\{u, v\}$, then u and v cannot be connected by a Hamilton path, because only the vertices of one component can be picked up. It is almost equally easy to show that a hamiltonian-connected graph

has toughness strictly larger than one. This can be seen by considering an arbitrary cut set S in a hamiltonian-connected graph G, and a Hamilton path P between two distinct vertices u and v of S (noting that $|S| \geq 3$ since G is 3-connected). Now, obviously $\omega(G - S) \leq \omega(P - S) \leq |S| - 1$, hence $\tau(G) > 1$.

In 1978, Jung [8] obtained the following result, in which he shows that for P_4-free graphs, the necessary condition $\tau(G) > 1$ is also a sufficient condition for hamiltonian-connectivity.

Theorem 3 ([8])**.** *Let G be a P_4-free graph. Then G is hamiltonian-connected if and only if $\tau(G) > 1$.*

3 Our results

In a paper of 2000 [6], Chen and Gould concluded that if $\{S, T\}$ is a pair of graphs such that every 2-connected $\{S, T\}$-free graph is hamiltonian, then every 3-connected $\{S, T\}$-free graph is hamiltonian-connected. Following up on this idea, we considered the following question. Suppose R is a graph such that every 1-tough R-free graph is hamiltonian. Is then every R-free graph G with $\tau(G) > 1$ hamiltonian-connected? For the purpose of answering this question, we tried to prove each of the forbidden subgraph cases analogous to the statement in Theorem 1. Of course Theorem 3 has already given us a partial positive answer. And indeed, we get a positive answer for each of these cases, as indicated in the following result.

Theorem 4. *Let R be an induced subgraph of $K_1 \cup P_3$ or $2K_1 \cup K_2$. Then every R-free graph G with $\tau(G) > 1$ on at least three vertices is hamiltonian-connected.*

We note here that from the proof of this result, it can be observed that the toughness condition $\tau(G) > 1$ in the above result cannot be weakened to the condition that the graph is 3-connected. We also proved the following analogue of Theorem 2, showing that except for the induced subgraphs of $K_1 \cup P_4$, there are no other forbidden induced subgraphs that can ensure every graph with toughness larger than one is hamiltonian-connected.

Theorem 5. *Let R be a graph on at least three vertices. If every R-free graph G with $\tau(G) > 1$ on at least three vertices is hamiltonian-connected, then R is an induced subgraph of $K_1 \cup P_4$.*

We conclude this extended abstract with the left unknown case as an open problem.

Problem 6. *Is every $K_1 \cup P_4$-free graph G with $\tau(G) > 1$ on at least three vertices hamiltonian-connected?*

As remarked earlier, we do not even know whether such graphs are hamiltonian, even if the condition on the toughness is replaced by $\tau(G) > t$ for any constant $t \geq 1$. Relations between different hamiltonian properties and toughness conditions have been studied in [1], leading to several equivalent conjectures, some seemingly stronger and some seemingly weaker that Chvátal's Conjecture. In the presentation, we will reflect on these aspects and sketch the key ingredients of the proofs of Theorem 4 and Theorem 5.

References

[1] D. Bauer, H.J. Broersma, J.P.M. van den Heuvel and H.J. Veldman, On hamiltonian properties of 2-tough graphs, *J. Graph Theory*, **18** (1994) 539–543.

[2] D. Bauer, H.J. Broersma and E. Schmeichel, Toughness in graphs - a survey, *Graphs Combin.,* **22** (2006) 1–35.

[3] D. Bauer, H.J. Broersma and H. J. Veldman, Not every 2-tough graph is hamiltonian, *Discrete Appl. Math.*, **99** (2000), 317–321.

[4] J.A. Bondy and U.S.R. Murty, *Graph Theory*, Springer Graduate Texts in Mathematics, vol. 244 (2008).

[5] H.J. Broersma, How tough is toughness?, *Bulletin of the EATCS*, **117** (2015).

[6] G. Chen and R. J. Gould, Hamiltonian connected graphs involving forbidden subgraphs, *Bull. Inst. Combin. Appl.*, **20** (2000) 25–32.

[7] V. Chvátal, Tough graphs and hamiltonian circuits, *Discrete Math.*, **5** (1973) 215–228.

[8] H.A. Jung, On a class of posets and the corresponding comparability graphs, *J. Combin. Theory, Ser. B*, **24** (1978) 125–133.

[9] D. Kratsch, J. Lehel and H. Müller, Toughness, hamiltonicity and split graphs, *Discrete Math.*, **150** (1996) 231–245.

[10] B. Li, H.J. Broersma and S. Zhang, Forbidden subgraphs for hamiltonicity of 1-tough graphs, *Discuss. Math. Graph Theory*, **36** (2016) 915–929.

[11] M.M. Matthews and D.P. Sumner, Hamiltonian results in $K_{1,3}$-free graphs, *J. Graph Theory*, **8** (1984) 139–146.

On sufficient spectral radius conditions for hamiltonicity

Qiannan Zhou[1,2], Hajo Broersma[2], Ligong Wang[1], and Yong Lu[3]

[1]Department of Applied Mathematics, Northwestern Polytechnical University, China
[2]Faculty of EEMCS, University of Twente, The Netherlands
[3]School of Mathematics and Statistics, Jiangsu Normal University, China

Abstract

During the last decade several research groups have published results on sufficient conditions for the hamiltonicity of graphs in terms of their spectral radius and their signless Laplacian spectral radius. Here we extend some of these results. All of our results involve the characterization of the exceptional graphs, i.e., all the nonhamiltonian graphs that satisfy the condition. The proofs of our main results are based on the Bondy-Chvátal closure, a degree sequence condition due to Chvátal, and an operation on the edges that is known as the Kelmans transformation.

1 Introduction

It is well-known that the problem of deciding whether a given graph is hamiltonian or not is an NP-complete problem. Many scholars have focussed on finding sufficient conditions for graphs to be hamiltonian. Nowadays, spectral graph theory is an important research area within the field of algebraic graph theory. It mainly deals with all kinds of spectral properties of the matrices of graphs (like the adjacency matrix, Laplacian matrix, signless Laplacian matrix, distance matrix, distance signless Laplacian matrix, and so on). During the last decade, many scholars have applied spectral graph theory to the problem of hamiltonicity of graphs. In this way, one can hope to judge whether a graph has a Hamilton cycle or not based on some spectral property. This is usually achieved by a combination of algebraic methods and the exploration of the graph structure.

We firstly introduce some basic terminology and notation. Let G be a graph with vertex set $\{v_1, v_2, \ldots, v_n\}$. Then the adjacency matrix $A(G)$ of G is the symmetric $n \times n$-matrix with entries $A(i, j) = 1$ if and only if $v_i v_j \in E(G)$ and zeros elsewhere. The diagonal degree matrix $D(G)$ of G is the $n \times n$-matrix with entries $D(i, i) = d(v_i)$ and zeros elsewhere. The matrix $Q(G) = D(G) + A(G)$ is known as the signless Laplacian matrix of G. The largest eigenvalue of $A(G)$, denoted by $\rho(A(G))$ or $\rho(G)$, is called the spectral radius of G. The largest eigenvalue of $Q(G)$, denoted by $q(Q(G))$ or $q(G)$, is called the signless Laplacian spectral radius of G.

Next we introduce Kelmans transformation [4] and the notion of an equitable partition, which are both helpful to prove our main results.

Let G be a graph and let $u, v \in V(G)$. We construct a new graph G^* by replacing all edges vx by ux for every $x \in N(v) \setminus N[u]$, which is known as Kelmans transformation.

Suppose M is a symmetric real $n \times n$-matrix whose rows and columns are indexed by $X = \{1, \ldots, n\}$. Let $\pi = \{X_1, \ldots, X_m\}$ be a partition of X. Let M be partitioned according to $\{X_1, \ldots, X_m\}$, i.e.,

$$M = \begin{pmatrix} M_{11} & \ldots & M_{1m} \\ \vdots & & \vdots \\ M_{m1} & \ldots & M_{mm} \end{pmatrix},$$

where M_{ij} denotes the block of M formed by the rows in X_i and the columns in X_j. Let $b_{ij} = \frac{\mathbf{1}^T M_{ij} \mathbf{1}}{|X_i|}$, i.e., the average row sum of M_{ij}, where $\mathbf{1}$ is the column vector (of the correct dimension) with all entries equal to 1. Then the matrix $M/\pi = (b_{ij})_{m\times m}$ is called the quotient matrix of M. If the row sum of each block M_{ij} is a constant, then the partition is called equitable.

To put our results in the right context, we first introduce a number of sufficient conditions for hamiltonicity in terms of the spectral radius, and signless Laplacian spectral radius, respectively, that were obtained in the last decade. Our aim is to extend these results.

2 Sufficient conditions in terms of spectral radii

We start with the results that are based on the spectral radius. In 2010, Fiedler and Nikiforov [2] presented the following sufficient condition for hamiltonicity.

Theorem 1 ([2]). *Let G be a graph of order n. If $\rho(G) > n-2$, then G is hamiltonian unless $G = K_1 \vee (K_{n-2} + K_1)$.*

Note that the graph $G = K_1 \vee (K_{n-2} + K_1)$ is clearly not hamiltonian: it has a vertex with degree 1, and is also not 1-tough. The work of [2] spurred the interest of several research groups. In 2015, Ning and Ge [9] obtained the following closely related sufficient condition for a graph G with minimum degree $\delta(G) \geq 2$ to be hamiltonian.

Theorem 2 ([9]). *Let G be a graph of order $n \geq 14$ with $\delta(G) \geq 2$. If $\rho(G) \geq \rho(K_2 \vee (K_{n-4}+2K_1))$, then G is hamiltonian unless $G = K_2 \vee (K_{n-4} + 2K_1)$.*

Note that $K_2 \vee (K_{n-4} + 2K_1)$ is another example of a graph that is not 1-tough. Excluding a family of four classes of graphs that are not 1-tough, Benediktovich [1] obtained the following extension of the result of Ning and Ge.

Theorem 3 ([1]). *Let G be a graph of order $n \geq 9$ with $\delta(G) \geq 2$. If $\rho(G) \geq n-3$, then G is hamiltonian unless $G \in \{K_4 \vee 5K_1, K_3 \vee (K_{1,4} + K_1), K_1 \vee (K_{n-3} + K_2), K_2 \vee (K_{n-4} + 2K_1)\}$.*

By imposing the minimum degree condition $\delta(G) \geq k$, for general integers $k \geq 1$, Li and Ning [5] established the following result.

Theorem 4 ([5]). *Let k be an integer, and let G be a graph of order n. If $\delta(G) \geq k \geq 1$ and $\rho(G) \geq \rho(K_k \vee (K_{n-2k} + kK_1))$, where $n \geq \max\{6k+5, (k^2+6k+4)/2\}$, then G is hamiltonian unless $G = K_k \vee (K_{n-2k} + kK_1)$.*

Recently, Nikiforov [8] extended and strengthened Theorem 4 in the following sense.

Theorem 5 ([8]). *Let G be a graph of order n with $\delta(G) \geq k$. If $k \geq 2$, $n \geq k^3 + k + 4$ and $\rho(G) \geq n-k-1$, then G is hamiltonian unless $G = K_1 \vee (K_{n-k-1} + K_k)$ or $G = K_k \vee (K_{n-2k} + kK_1)$.*

More recently, Ge and Ning [3] showed that the statement in the above theorem due to Nikiforov also holds for $k \geq 1$ and $n \geq \max\{\frac{1}{2}k^3 + k + \frac{5}{2}, 6k+5\}$.

For the signless Laplacian radius, we start with the following results due to Yu and Fan [10], and Liu et al. [7], respectively.

Theorem 6 ([10]). *Let G be a graph of order $n \geq 3$. If $q(G) > 2n-4$, then G is hamiltonian unless $G = K_2 \vee 3K_1$ or $G = K_1 \vee (K_{n-2} + K_1)$.*

Theorem 7 ([7]). *Let G be a graph of order $n \geq 4$ with $\delta(G) \geq 2$. If $q(G) \geq 2n - 5 + \frac{3}{n-1}$, then G is hamiltonian unless $G = K_3 \vee 4K_1$ or $G = K_2 \vee 3K_1$.*

By imposing the minimum degree condition $\delta(G) \geq k$, for general integers $k \geq 1$, Li and Ning [5] established the following counterpart of Theorem 4.

Theorem 8 ([5]). *Let k be an integer, and let G be a graph of order n. If $\delta(G) \geq k \geq 1$ and $q(G) \geq q(K_k \vee (K_{n-2k} + kK_1))$, where $n \geq \max\{6k + 5, (3k^2 + 5k + 4)/2\}$, then G is hamiltonian unless $G = K_k \vee (K_{n-2k} + kK_1)$.*

The most recent result in this area that we are aware of is the following result due to Li et al. [6].

Theorem 9 ([6]). *Assume $k > 1$ and $n \geq k^4 + k^3 + 4k^2 + k + 6$. Let G be a connected graph with n vertices and $\delta(G) \geq k$. If $q(G) \geq 2(n - k - 1)$, then G is hamiltonian unless $G \in \mathcal{M}_1(n, k)$ or $G \in \mathcal{L}_1(n, k)$.*

This result involves the two exceptional classes $\mathcal{M}_1(n, k)$ and $\mathcal{L}_1(n, k)$ that we will not define here. We refer the interested reader to [6] for the definitions.

3 Our results

Following up on the above results in terms of the spectral radius, we obtain the following two results. We will sketch the key ingredients of the proofs of these results in the presentation.

Theorem 10. *Let G be a graph of order $n \geq 6k^2 + 4k + 2$ with $\delta(G) \geq k \geq 1$. If*

$$\rho(G) > \frac{k-1}{2} + \sqrt{n^2 - (3k+1)n + \frac{(k+1)^2 - 4}{4}},$$

then G is hamiltonian unless $cl_n(G) = K_1 \vee (K_{n-k-1} + K_k)$ or $cl_n(G) = K_k \vee (K_{n-2k} + kK_1)$.

In the above statement, $cl_n(G)$ denotes the Bondy-Chvátal closure of G, obtained from G by recursively joining pairs of non-adjacent vertices by an edge whose degree sum is at least n (in the current graph) until no such pair remains.

It is easy to check that $n - k - 1 > \frac{k-1}{2} + \sqrt{n^2 - (3k+1)n + \frac{(k+1)^2-4}{2}}$. Hence Theorem 10 extends Theorem 5, in the sense that the lower bound on $\rho(G)$ is generally better.

We also obtained the following result for graphs on $n \geq 5$ vertices with $\delta(G) \geq 1$.

Theorem 11. *Let G be a graph of order $n \geq 5$ with $\delta(G) \geq 1$. If*

$$\rho(G) > \sqrt{n^2 - 4n},$$

then G is hamiltonian unless $G \in \mathcal{G}_1 = \{G_1, G_1^1, G_1^2, G_3^6, G_3^9, G_3^{17}, G_3^{18}, G_3^{21}, G_3^{22}, G_3^{24}, G_3^{26}\}$.

Here $\mathcal{G}_1$ is a set of exceptional graphs, which will be depicted in our presentation. Note that our result is an improvement of Ge and Ning's result in [3], i.e., the aforementioned statement of Theorem 5 in the case $k = 1$.

For the results in terms of the signless Laplacian spectral radius, we obtain the following two results.

Theorem 12. *Let G be a graph of order $n \geq 6k^2 + 4k + 3$ with $\delta(G) \geq k \geq 1$. If*

$$q(G) > 2n - 2k - 2 - \frac{2k+1}{n-1},$$

then G is hamiltonian unless $cl_n(G) = K_1 \vee (K_{n-k-1} + K_k)$ or $cl_n(G) = K_k \vee (K_{n-2k} + kK_1)$.

It is obvious that the lower bound condition on $q(G)$ of Theorem 12 is weaker than that of Theorem 9. Hence Theorem 12 strengthens Theorem 9 in that sense.

Theorem 13. *Let G be a graph of order $n \geq 6$ with $\delta(G) \geq 1$. If*

$$q(G) > 2n - 4 - \frac{3}{n-1},$$

then G is hamiltonian unless $G \in \mathcal{G}_2 = \{G_1, G_1^1, G_1^2, G_3^6, G_3^8, G_3^9, G_3^{17}, G_3^{21}, G_3^{23}, G_3^{26}\}$.

Here $\mathcal{G}_2$ is a set of exceptional graphs, which will also be depicted in our presentation. Noting that K_{n-2} is a proper subgraph of G_1, we obtain that $q(G_1) > q(K_{n-2}) = 2n - 4$. Therefore our result improves Theorem 8 in the case $k = 1$.

References

[1] V. I. Benediktovich, Spectral condition for Hamiltonicity of a graph, Linear Algebra Appl. 494 (2016) 70–79.

[2] M. Fiedler, V. Nikiforov, Spectral radius and Hamiltonicity of graphs, Linear Algebra Appl. 432 (2010) 2170–2173.

[3] J. Ge, B. Ning, Spectral radius and Hamiltonicity of graphs and balanced bipartite graphs with large minimum degree, Preprint available at arXiv:1606.08530v3.

[4] A. K. Kelmans, On graphs with randomly deleted edges, Acta Math. Acad. Sci. Hung. 37 (1981) 77–88.

[5] B. L. Li, B. Ning, Spectral analogues of Erdős' and Moon–Moser's theorems on Hamilton cycles, Linear and Multilinear Algebra 64 (2016) 2252–2269.

[6] Y. W. Li, Y. Liu, X. Peng, Signless Laplacian spectral radius and Hamiltonicity of graphs with large minimum degree, Linear Multilinear Algebra 66 (2018) 2011–2023.

[7] R. F. Liu, W. C. Shiu, J. Xue, Sufficient spectral conditions on Hamiltonian and traceable graphs, Linear Algebra Appl. 467 (2015) 254–266.

[8] V. Nikiforov, Spectral radius and Hamiltonicity of graphs with large minimum degree, Czechoslovak Math. J. 66 (2016) 925–940.

[9] B. Ning, J. Ge, Spectral radius and Hamiltonian properties of graphs, Linear and Multilinear Algebra 63 (2015) 1520–1530.

[10] G. D. Yu, Y. Z. Fan, Spectral conditions for a graph to be Hamilton-connected, Applied Mechanics and Materials, 336-338 (2013) 2329–2334.

Alphabetical list of authors