15th Cologne-Twente Workshop on Graphs and Combinatorial Optimization

Cologne, Germany, June 6-8, 2017

Extended Abstracts

Bert Randerath, Heiko Röglin, Britta Peis, Oliver Schaudt, Rainer Schrader,
Frank Vallentin, Vera Weil (eds.)

Scientific Committee

Ali Fuat Alkaya (U Marmara)

Albert Ceselli (U Milano)

Roberto Cordone (U Milano)

Ekrem Duman (U Ozyegin)

Ulrich Faigle (U Köln)

Johann L. Hurink (U Twente)

Leo Liberti (CNRS France)

Bodo Manthey (U Twente)

Gaia Nicosia (U Roma Tre)

Andrea Pacifici (U Roma Tor Vergata)

Stefan Pickl (UBw München)

Bert Randerath (TH Köln)

Giovanni Righini (U Milano)

Heiko Röglin (U Bonn)

Britta Peis (RWTH Aachen)

Oliver Schaudt (U Köln)

Rainer Schrader (U Köln)

Rüdiger Schultz (U Duisburg-Essen)

Organizing Committee

Bert Randerath

Heiko Röglin

Britta Peis

Oliver Schaudt

Rainer Schrader

Frank Vallentin

Vera Weil

Local Organization

Alexander Apke

Toni Böhnlein

Jun-Gyu Kim

Roland Mainka

Martin Olschewski

Andrea Oversberg

Fabian Senger

Table of Contents

Heuristics for a Green Orienteering Problem

Saverio Basso[1], Marco Casazza[1], and Alberto Ceselli[1]

[1]Università degli Studi di Milano, Dipartimento di Informatica, Italy.

We address a routing problem where a vehicle with limited time, loading capacity and battery autonomy can optionally serve a set of customers, each providing a profit. Such a problem is of particular relevance both because of its practical implications in sustainable transportation and its use as a sub-problem in *Green Vehicle Routing* column generation algorithms. We propose a dynamic programming approach to obtain both primal and dual bounds to the value of the optimal solutions, a fast greedy heuristics and a very large scale neighbourhood search procedure.

1 Introduction

Pushed by the constant increase in gasoline costs, and eco-sustainability awareness of consumers, the market share of vehicles powered by alternative fuels is increasing steadily. In traditional routing problems, fuel autonomy is typically assumed sufficient to serve all customers because of both the big size of the tank and the short refuelling time. However, such conditions do not hold for vehicles running on alternative fuels such as electric batteries, whose full recharge can take up to several hours. Therefore, an explicit planning of the refuelling stops is required to satisfy time constraints, such as in [3].

Within this context we address the *Green Orienteering Problem (GOP)*, a variant of the *Orienteering Problem* [2] involving a single vehicle running on electric batteries only. Each time the vehicle visits a customer it collects a profit. However, to travel between customers the vehicle consumes both battery and time resources. Recharge stations are available on the network, which are equipped with different recharging technologies, offering different trade-off between recharge time and cost. During its route, the vehicle can stop at recharge stations, selecting a particular technology and the amount of energy to be recharged. A fixed cost is always paid at each recharge. The aim is to find a route that maximizes the difference between collected profits and recharge costs, in such a way that (a) a time limit is not exceeded, (b) the loading capacity of the vehicle is not exceeded when serving customer demands, and (c) the vehicle never runs out of battery.

Our GOP is of particular relevance since it arises as a sub-problem in *Green Vehicle Routing Problems* [1] when they are solved exploiting column generation techniques. We propose a methodology based on dynamic programming to obtain both primal and dual bounds to the value of optimal solutions, a fast greedy algorithm, and a very large scale neighbourhood search procedure.

2 Modelling

The GOP can be formulated as follows: let $G = (V \cup R, E)$ be an undirected graph, where V is the set of customer vertices, R is the set of station vertices, and $E = \{(i,j) \mid i,j \in V \cup R\}$ is a set of edges connecting them. To perform customer visits we are given a single vehicle of limited loading capacity Q and limited time availability T, equipped with a battery of maximum charge B. The vehicle starts and ends at a depot $o \in R$. Each time the vehicle visits a customer i, it serves a demand q_i and collects a profit p_i using s_i units of time. Also, when the vehicle travels along an edge $(i,j) \in E$ it consumes $d_{(i,j)}$ units of battery charge and $t_{(i,j)}$ units of time. We assume $q_i = p_i = 0$ for each $i \notin V$.

The vehicle cannot travel if the battery charge reaches zero, but it can be recharged at each station vertex $r \in R \setminus \{r_0\}$ using one out of a set K of technologies. Each technology $k \in K$ provides b_k battery charge units for each unit of time at a cost c_k for each unit of battery recharged. Also, a fixed cost f is paid at each recharge. Mixing technologies during the same recharge stop is forbidden.

A route $\rho = ((i_1, \delta_1, k_1), \ldots, (i_n, \delta_n, k_n))$ is a sequence of triplets describing the customers visited, the order of visits, and charge information, where $i \in V \cup R$ is a vertex and δ is the time spent recharging the vehicle at vertex i using technology k. When i is a customer, recharge is forbidden: δ is fixed to 0 and k is set to a dummy value. A route is feasible if:

- there is an edge between vertices of two following triplets: $\exists (i_\sigma, i_{\sigma+1}) \in E, \forall \sigma = 1 \ldots n-1$;
- it does not exceed the time limit: $\sum_{\sigma=1}^{n-1} t_{(i_\sigma, i_{\sigma+1})} + \sum_{\sigma=1}^{n} s_{i_\sigma} + \sum_{\sigma=1}^{n} \delta_\sigma \leq T$;
- it does not exceed the capacity: $\sum_{\sigma=1}^{n} q_{i_\sigma} \leq Q$;
- the battery level is always between 0 and B: $\sum_{\sigma=1}^{\sigma'-1} \delta_\sigma b_{k_\sigma} - d_{(i_\sigma, i_{\sigma+1})} \geq 0, \forall \sigma' = 2 \ldots n$ and $\delta_{\sigma'} b_{k'_\sigma} + \sum_{\sigma=1}^{\sigma'-1} \delta_\sigma b_{k_\sigma} - d_{(i_\sigma, i_{\sigma+1})} \leq B, \forall \sigma' = 2 \ldots n$.

A route is optimal if it is feasible and its value, computed as the difference between collected profits and recharge costs, is maximum: max $\sum_{\sigma=1 | i_\sigma \in V}^{n} p_{i_\sigma} - \sum_{\sigma=1 | i_\sigma \in R}^{n} f + \delta_\sigma b_{k_\sigma} c_{k_\sigma}$.

3 Algorithms

We propose a methodology to obtain both primal and dual bounds to the value of the optimal solutions of GOP. We start from a simple observation:

Observation 1. *If a utopia technology having recharge speed $\hat{b} = \max_{k \in K} b_k$ and recharge cost $\hat{c} = \min_{k \in K} c_k$ exists, it would always be profitable to select it.*

That is, all the other technologies would be dominated. We accordingly define the *Utopia Technology GOP (UT-GOP)*, where we suppose that at each station, such an additional utopia technology is available (and therefore always selected), and we prove that:

Observation 2. *any sequence of visits yielding a feasible UT-GOP route, and in particular an optimal one, yields a feasible GOP route as well.*

Proposition 1. *The value of an optimal UT-GOP solution is a dual bound to GOP.*

To solve to optimality the UT-GOP we define a label correcting algorithm having the following structure:

Label structure: partial routes starting from the depot r_0 and ending in vertex i are encoded as labels $\lambda = (i, \tilde{V}, p, x, y, z)$ where p is the profit of the partial route computed as the difference of the collected profits and the recharge costs, $\tilde{V}$ is the set of customer vertices visited in the partial route, x is the potential battery level, y is the residual time, and z is the residual capacity.

Initialization: an initial label $\lambda = (r_0, \emptyset, 0, B, T, Q)$ is created and marked as 'open'.

Extension: at each iteration an open label $\lambda = (i, \tilde{V}, p, x, y, z)$ having maximum p value is selected and for each edge $(i, j) \in E$ a new label $\lambda' = (j, \tilde{V}', p', x', y', z')$ is created. If vertex j is a customer, that is $j \in V$, we set $p' = p + p_j - d_{(i,j)}\hat{c}$, $\tilde{V}' = \tilde{V} \cup \{j\}$, $x' = x - d_{(i,j)}$, $y' = y - t_{(i,j)} - s_j - d_{(i,j)}/\hat{b}$, and $z' = z - q_j$. Otherwise, if $j \in R$, $p' = p - f - d_{(i,j)}\hat{c}$, $\tilde{V}' = \tilde{V}$, $x' = x - d_{(i,j)}$, $y' = y - t_{(i,j)} - d_{(i,j)}/\hat{b}$, and $z' = z$. Label λ is marked as 'closed' while λ' as 'open'. Extension is not performed when $x' < 0$, $y' < 0$, or $z' < 0$, which encode an infeasible route.

Recharge: when a label $\lambda = (i, \tilde{V}, p, x, y, z)$ is created at a vertex $i \in R$, the potential battery level is fully charged by setting $x = B$.

Dominance: after extension, if two labels $\lambda' = (i, \tilde{V}', p', x', y', z')$ and $\lambda'' = (i, \tilde{V}'', p'', x'', y'', z'')$ are found, having $p' \geq p''$, $\tilde{V}' \subseteq \tilde{V}''$, $x' \geq x''$, $y' \geq y''$, and $z' \geq z''$, and at least one of these inequalities is strict, then label λ'' is deleted being sub-optimal.

Stopping criteria: we stop when no 'open' label is left.

According to Observation 2 by optimizing the UT-GOP we obtain a sequence of visits allowing to build a feasible GOP route, thereby obtaining a primal bound. In particular, we need to adjust the technology selection and amount of recharge done at each station. To perform such an adjustment we formulate a new optimization problem where given an incomplete route ρ, we have to find δ and k values in such a way that the route is still feasible and it minimizes the recharge costs.

Let $\tilde{R}$ be the set of recharge stations visited in ρ. W.l.o.g. if a station r is visited more than once, $\tilde{R}$ contains its copies r', r'', and so on. The problem of adjusting recharge quantities is

formulated as follows:

$$\min \quad \sum_{r\in\tilde{R}}\sum_{k\in K} c_k b_k y_{rk} \tag{1}$$

$$\text{s.t.} \quad \sum_{k\in K} x_{rk} \leq 1 \qquad \forall r \in \tilde{R} \tag{2}$$

$$\sum_{r\in\tilde{R}}\sum_{k\in K} y_{rk} \leq T - \sum_{\sigma=1}^{n-1} t_{(i_\sigma, i_{\sigma+1})} \tag{3}$$

$$\sum_{\sigma=1|i_\sigma\in\tilde{R}}^{\sigma'-1}\sum_{k\in K} b_k y_{i_\sigma k} - \sum_{\sigma=1}^{\sigma'-1} d_{(i_\sigma, i_{\sigma+1})} \geq 0 \qquad \forall \sigma' = 2\dots n \tag{4}$$

$$\sum_{\sigma=1|i_\sigma\in\tilde{R}}^{\sigma'}\sum_{k\in K} b_k y_{i_\sigma k} - \sum_{\sigma=1}^{\sigma'-1} d_{(i_\sigma, i_{\sigma+1})} \leq B \qquad \forall \sigma' = 2\dots n \tag{5}$$

$$b_k y_{rk} \leq B x_{rk} \qquad \forall r \in \tilde{R}, k \in K \tag{6}$$

$$y_{rk} \geq 0, x_{rk} \in \mathbb{B} \qquad \forall r \in \tilde{R}, k \in K \tag{7}$$

where y_{rk} is the time spent at station r recharging the vehicle with technology k and x_{rk} is a binary variable that is 1 if technology k is used at station r, 0 otherwise. The objective function (1) minimizes recharge costs. Constraints (2) forbid mixing technologies in a single station. Constraint (3) enforces the time limit. Constraints (4) and (5) ensure that the vehicle does not travel with empty battery and the maximum battery level is never exceeded, respectively. Constraints (6) enforce that variables x_{rk} are set to 1 when a technology is used in a station.

Greedy heuristic and very large scale neighbourhood search. We also propose a greedy heuristic for GOP that iteratively moves between vertices until the vehicle runs out of battery or time resources. At each step, the algorithm selects one out of three possible operations: (a) move the vehicle to the neighbour customer maximizing the difference between the collected profit and the travelling cost, (b) take a detour to a recharge station if no customer can be visited with current resources, or (c) go back to depot if the previous operations invalidate route feasibility.

Once a route is built, we re-optimize the technology selection and energy recharge amounts by optimizing a MIP sub-problem similar to (1) – (7), that corresponds to exploring a neighbourhood whose size is exponential in the number of stations. Such an optimization is performed by means of general purpose MIP solvers.

References

[1] A. Ceselli, A. Felipe, M.T. Ortuño, G. Righini, and G. Tirado, *A branch-and-cut-and-price algorithm for the green vehicle routing problem with partial recharge and multiple technologies*, Odysseus 2015, Ajaccio (2015).

[2] P. Vansteenwegen, W. Souffriau, and D. Oudheusden, *The orienteering problem: A survey*, European Journal of Operational Research (2011).

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

A Characterization of Affine Oriented Matroids

Abstract for CTW 2017

Andrea Baum[1] and Yida Zhu[2]

[1]Department of Mathematics, University of Hamburg, 20146 Hamburg, Germany
[2]RTG Algorithmic Optimization, Department of Mathematics, University of Trier, 54296 Trier, Germany

Oriented matroids can be thought of as combinatorial abstractions of real hyperplane arrangements, which arise as fundamental objects in various mathematical theories such as inequality systems in linear programming, facets of convex polytopes and so on. *Affine oriented matroids* are the corresponding abstraction of *affine* hyperplane arrangements.

The standard axiomatization of oriented matroids requires a common intersection of all hyperplanes in the arrangement and thus does not extend naturally to the affine context. In an unpublished manuscript [2], Johan Karlander has given an axiomatization of affine oriented matroids by identifying every parallel class of affine hyperplanes with a sign vector (parallel vector), resembling the point at infinity from projective geometry.

Arrangement of hyperplanes and sign vector system

A finite family $\mathcal{H} = \{H_e : e \in E\}$ of hyperplanes in $\mathbb{R}^d$ is called an *arrangement of hyperplanes*. The arrangement is *affine* if its hyperplanes are. We focus on the combinatorial structure of such arrangements, that is, how those hyperplanes partition the space.

Every point $x \in \mathbb{R}^d$ will be assigned to a *sign vector* given by $(X_i)_{i\in E} \in \{+,-,0\}^E$, where X_i tells the position of x regarding H_i, see Figure 1. The system of all different sign vectors (SVS) induced by $\mathcal{H}$ forms the covector system of an oriented matroid [1, chapter 2]. In general, [1, 4.1.1] give an axiomatization of such SVS. One of this axioms requires that all hyperplanes have a common intersection. Certainly, this axiom is no longer valid by generalizing the concept to affine arrangements: two affine hyperplanes may be parallel to each other.

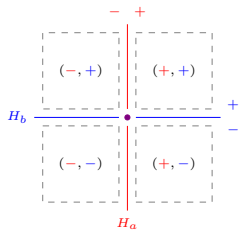


Figure 1: Partition of $\mathbb{R}^2$.

Affine arrangement and affine oriented matroid

In the case $\bigcap_{e\in E} H_e = \emptyset$, one has to raise the dimension by 1 and add a hyperplane into the embedded arrangement, such that all axioms in [1, 4.1.1] are fulfilled in the embedded system (which forms an oriented matroid), as illustrated in Figure 2. Note that the partition of $\mathbb{R}^2$ can be found in any affine hyperplane $\{(x, y, z) \in \mathbb{R}^3 : z = c\}$ in $\mathbb{R}^3$. From this perspective, the SVS induced by some affine arrangement is actually a subset of some oriented matroid in higher dimension, more precisely: If $\mathcal{W} \subseteq \{+,-,0\}^E$ is a SVS induced by some affine arrangement, then

(a) Arrangement of parallel affine hyperplanes in $\mathbb{R}^2$. (b) Embedded arrangement of hyperplanes in $\mathbb{R}^3$.

Figure 2: Embedding process to avoid $\bigcap_{e \in E} H_e = \emptyset$.

there exists some oriented matroid $\mathcal{O} \subseteq \{+, -, 0\}^{E+z}$ such that $\mathcal{O} = \{(X, +) : X \in \mathcal{W}\}$. We call such SVS *affine oriented matroid*.

Sign vectors corrsponding to points at infinity

In order to demonstrate the idea of parallel vectors, we consider the real space $\mathbb{R}^3$ as the interior of a unit ball (Figure 3b). The point at infinity of the parallel class $\{H_1, H_2\}$ is presented by the intersection of their embedded image, which does not belong to $\mathbb{R}^3$ (i.e., right on the shell). The axiom of Karlander states a closure property of the affine oriented matroid $\mathcal{W}$ together with the system $\mathcal{P}(\mathcal{W})$ of all parallel vectors.

(a) A parallel class $\{H_1, H_2\}$ of two hyperplanes in $\mathbb{R}^2$. (b) Corresponding parallel vectors P and $-P$.

Figure 3: Sketch for the concept of parallel vectors.

In the original work [2], there was a small gap in the main proof. However, the axiomatization itself remains true. We offer an alternative construction to fix this issue and simplify the structure of $\mathcal{P}(\mathcal{W})$ by including the entire horizon (the gray cycle on the shell in Figure 3b) into $\mathcal{P}(\mathcal{W})$.

References

[1] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, G.M. Ziegler, *Oriented Matroids*, Cambridge Univ. Press, 1993.

[2] J. Karlander, A characterization of affine sign vector systems, in *Zero-one matrices matroids and characterization problems*, pp. 67-91, Ph.D. Thesis KTH Stockholm, 1992.

Simplicial Decomposition for Large-Scale Quadratic Convex Programming

Enrico Bettiol[1], Lucas Létocart[1], Francesco Rinaldi[2], and Emiliano Traversi[1]

[1]LIPN UMR CNRS 7030 Université Paris 13, Sorbonne Paris Cité, 99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France

[2]Università di Padova, Dipartimento di Matematica, Via Trieste, 63, 35121 Padova, Italia

We consider the following problem

$$\begin{array}{ll} \min & f(x) \;=\; x^\top Q x + c^\top x + d \\ \text{s.t.} & Ax \;=\; b \\ & x \geq 0 \end{array} \tag{1}$$

with $Q \in \mathbb{R}^{n\times n}$, $c \in \mathbb{R}^n$, $d \in \mathbb{R}$, $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. When the size of the problem is large, very often it is more convenient to take advantage of smart or ad-hoc strategies to tackle the problem. Column generation, described for example in [3], represents one of the most important ways to deal with large-scale problems. In thi work we present a column generation algorithm called Simplicial Decomposition. We develop new techniques in order to make it more efficient and we compare our algorithm against the state-of-the-art software *CPLEX*. We present our algorithm and show our results, obtained on portfolio optimization problems and on general convex quadratic problems.

1 Simplicial Decomposition

The idea behind the Simplicial Decomposition (SD) algorithm is described in [5]. The original problem is decomposed into two simpler subproblems, which are called pricing and master problems, and are solved alternatively and repeatedly. The pricing solves the original problem with a linear objective function and the master problem, instead, is a problem with the original objective function, but with lower dimension and simplified constraints.

More specifically, starting from a single point, the domain of each master problem is the convex hull of a finite set of (hopefully few) affinely independent points, i.e. a simplex, and these points are the solution of the previous pricing problems: if $B_k := \{x_1, \dots x_k\}$ is the set of points after the k^{th} iteration, the simplex S_k generated by these points can be represented as a convex combination of the generators, so the dimension of the master is $k \ll n$.

On the other hand, each pricing problem is linear because it minimizes the gradient of the original objective function in the optimal point of the previous master. In this way, it obtains a descent direction, so the new master will be able to find points with lower cost; otherwise, if no new points are found, the algorithm terminates.

2 Setting

We compare the SD algorithm with the *state-of-the-art* solver *CPLEX*. Furthermore, we introduced some modifications in the SD algorithm, both for the pricing and for the master problems, in order to improve the convergence.

2.1 Strategies for Efficiently Solving the Master

The master problem can be solved in various ways: a first one is directly using *CPLEX* itself (namely *SD - Cplex*). We propose two other methods in order to speed up the computational time of the master problem, based respectively on the Conjugate Directions and on the Projected Gradient Methods.

We introduce a new Adaptation of the (unconstrained) Conjugate Directions Method (SD - ACDM) in order to exploit the particular structure of the simplices that are generated at each iteration of the algorithm. The informations used to solve each master problem is reused to solve the next one in a single iteration. In this way, we introduced a significant warm start, that is not possible with *CPLEX*.

More specifically, we exploit the property that, for every master problem, the search for the optimum starts from a point in the interior of a facet of its domain. So we develop and prove the convergence of a method that proceeds in this way: it starts from the previous optimum, it finds the new conjugate direction, it determines the optimum along this new line with unconstrained techniques (described, for example, in [2]) and, if needed, it projects the point onto the simplex, in order to get the constrained optimal point. In this case the intersection with the boundary of the simplex is found and the method finds the optimal point in the intersection face.

The second approach is a Fast Gradient Projection Method (FGPM) and belongs to the family of gradient projection approaches.

2.2 Strategies for Efficiently Solving the Pricing

With respect to the pricing, it is always solved by *CPLEX*, however, we introduce some features in order to improve its performance.

We introduce a family of cuts called *shrinking cuts*, in order to speed-up the solution of the pricing and to generation more useful columns. The second strategy tested is the use of an early stopping in the pricing problem: this reduces the computational time of the pricing and is still sufficient to provide a descent direction at each cycle, which is enough to guarantee the convergence of the algorithm. Finally, we tested the use of the sifting solver for *CPLEX*, instead of the default settings.

3 Computational Results

We use two set of instances as test-bed: Portfolio Instances and Generic Quadratic Instances. For lack of space, we report the results concerning only the Portfolio Instances. The results related to Generic Quadratic instances are, however, similar.

In this section we report the performances of our algorithm in base of the different settings used. Resuming, We use three methods for solving the master programs: Cplex, ACDM,

FGPM, which have been described before. Regarding the pricing problems, we evaluate the following combinations of settings:

- Default
- Cuts
- Early stopping
- Cuts + early stopping
- Sifting
- Sifting + Cuts
- Sifting + Early stopping
- Sifting + Cuts + Early stopping

In Figure 1 we report the performance for the faster master solver, which is ACDM, for these instances.

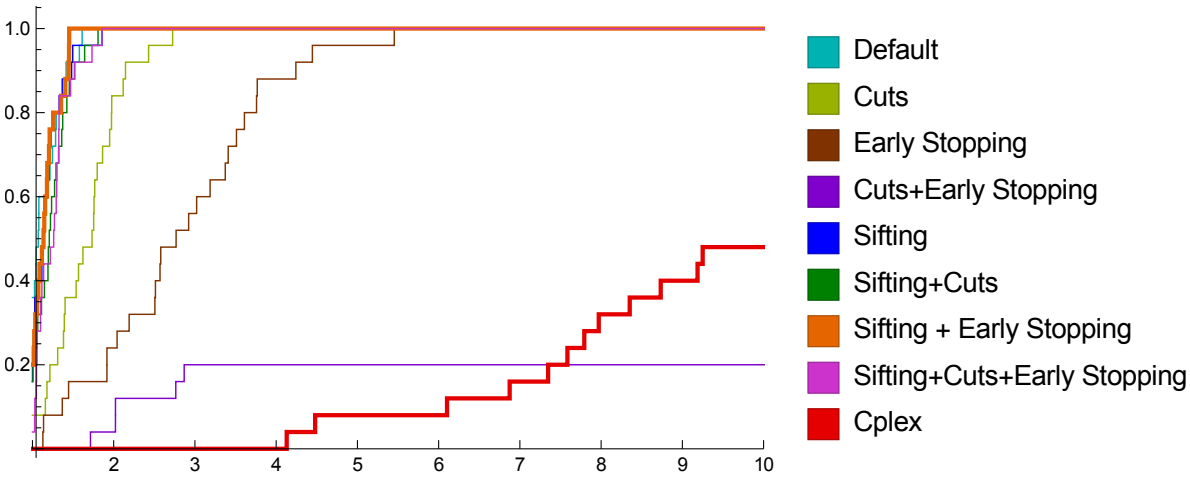


Figure 1: Performance profile for Portfolio, master solved with ACDM.

The best option for the pricing is Sifting + Early Stopping.

Now, Figure 2 shows the comparison among the three different SD master solvers and the only-Cplex solver. For each SD method, the best pricing option (found analogously as for the previous one) is used. For the Portfolio instances, the best setting for the pricing is

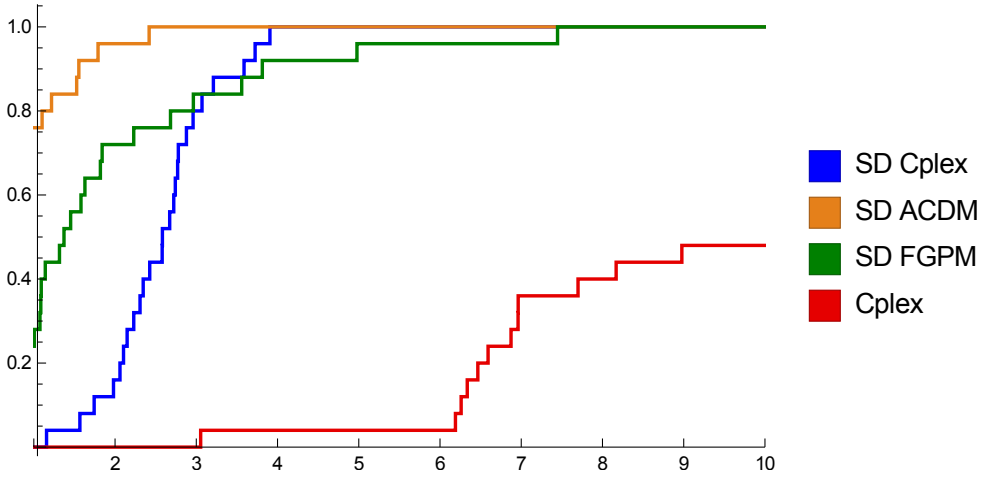


Figure 2: Performance profile for Portfolio, master solvers.

Sifting+Early Stopping.

4 Conclusions

PORTFOLIO	LOW M	LARGE M
SD ACDM		SD FGPM
SIFTING		
EARLY STOPPING		
		CUTS

Table 1: Overview.

We presented an efficient approach to solve continuous quadratic problems based on Simplicial Decomposition. We experimentally prove that our algorithm is more efficient than a generic solver on the instances tested. To achieve this result, we proposed two ad-hoc techniques for solving the master problem: the Conjugate Directions based Method (ACDM) and the Fast Gradient Projection Method (FGPM); and three additional improvements on the standart Simplicial Decomposition method: an Early Stopping techniques, a Shrinking method based on additional cuts on the pricing and a specific way to solve the pricing problem via the sifting algorithm. In Table 1, we summarize which setting is recommended in relation to the instances we want to solve. The instances are sorted from left to right according to an increasing value of $M = \frac{m}{n}$. For portfolio instances, the best master optimizer is ACDM and the best pricing options are sifting and sifting with early stopping. The cuts play an important role and are the best pricing option for the largely constrained problems. Moreover, for this class of problems the best master solver is the Projected gradient.

References

[1] E.G. Birgin, J.M. Martinez and M. Raydan. *Nonmonotone spectral projected gradient methods for convex sets.* SIAM Journal on Optimization 10, pp. 1196-1211, 2000.

[2] L. Grippo, M. Sciandrone. *Metodi di ottimizzazione non vincolata.* Springer, 2011.

[3] Larsson, Torbjörn, Athanasios Migdalas, and Michael Patriksson. *A generic column generation scheme.* 2006.

[4] Markowitz, H. *Portfolio Selection.* The Journal of Finance, Vol. 7, No. 1, pp. 77-91. March. 1952.

[5] B. Von Hohenbalken. *Simplicial decomposition in nonlinear programming algorithms.* Mathematical Programming 13 (1977) 49-68.

Robust Bipartite Matching Augmentation

Viktor Bindewald[1] and Moritz Mühlenthaler[1]

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

We investigate the complexity of making a given perfect matching in a bipartite graph robust against the deletion of at most k edges by adding at most ℓ edges to the graph. We show that for $k = 1$ the problem is NP-complete and it is W[2]-hard when parametrized by ℓ. We further show that the problem can be solved in time $O(mn + n^2)$ if $k = \ell = 1$. In addition, we show that the problem remains hard if budget restrictions are imposed on repairing the matching, unless the budget is very small.

1 Introduction

An *augmentation problem* asks how many edges need to be added to a graph in order to have the resulting graph satisfy some desired property, such as bridge- or biconnectivity [5], or hamiltonicity [6, GT34]. We consider an augmentation problem related to robust perfect matchings in a bipartite graph: We call a perfect matching M in a graph G *k-robust* if, after the deletion of at most k M-edges, G still admits a perfect matching.

Problem 1 (k-Robust Bipartite Matching ℓ-Augmentation)**.** *Given a simple bipartite graph $G = (X + Y, E)$ and a perfect matching M in G, can we make M k-robust by adding at most ℓ XY-edges to G?*

Robust decision problems are often phrased as follows: The task is to determine if some infrastructure, say a graph G, has some desired property, no matter what goes wrong. For example, after removing at most k edges from G, does the resulting graph contain some structure of interest such as a spanning tree [2], a perfect matching [1, 4], or an st-path [3]. Such problems and their optimization variants are often hard even if the underlying nominal problems can be solved in polynomial time. In our setting, we are given a graph and a fixed structure of interest, i.e., a perfect matching, and would like to harden the infrastructure by adding a certain number of edges to the graph. In other words we are addressing the following question: Is it easier to harden existing infrastructure rather than designing a robust one from scratch? However, our results indicate that in general the answer is no, as implied by our main result.

Theorem 2. 1-Robust Bipartite Matching ℓ-Augmentation *is* NP*-complete.*

In order to prove Theorem 2, we use a reduction from Set Cover. In addition to NP-hardness, the reduction implies that 1-Robust Bipartite Matching ℓ-Augmentation is W[2]-hard when parametrized by ℓ. Furthermore, there is no polynomial time constant-factor

approximation for the problem of minimizing ℓ. On the positive side, we show that a 1-Robust Bipartite Matching ℓ-Augmentation instance with $\ell = 1$ can be solved in time $O(mn + n^2)$, where m is the size of the input graph and n is its order.

In [4], a related robust decision problem is considered, where the matching is not fixed and there is an additional budget restriction for repairing a matching after removing at most k edges from the input graph. A perfect matching M in a graph $G = (V, E)$ is *k-robust s-recoverable*, if for each set $F \subseteq E$ of cardinality at most r, $G - F$ has a perfect matching N such that the symmetric difference $M \triangle N$ contains at most s edges. Dourado et al. proved the following hardness result by a reduction from 3Sat:

Theorem 3 ([4, Theorem 3])**.** *For any $s \geq 2$ it is* NP*-complete to decide if a given bipartite graph G admits a 1-robust $2s$-recoverable perfect matching.*

The proof of our main result also implies that the following augmentation problem is NP-complete for any $s \geq 3$:

Problem 4 (1-Robust s-Recoverable Bipartite Matching ℓ-Augmentation)**.** *Given a simple bipartite graph $G = (X + Y, E)$ and a perfect matching M in G, can make M 1-robust $2s$-recoverable by adding at most ℓ XY-edges edges to G?*

Quite surprisingly, it turns out, that the problem can be solved in polynomial time if $s = 2$, by a reduction to the Minimum Cost Edge Cover problem.

2 Our Results

If k and ℓ are fixed, then k-Robust Bipartite Matching ℓ-Augmentation can be solved in polynomial time using a brute-force approach: Let $\overline{G} = (V, \overline{E})$ be the bipartite complement of the input graph $G = (V, E)$. Check for each $U \subseteq \overline{E}$ of cardinality at most ℓ if the matching M is k-robust in $G + U$. In order to test if $G + U$ is k-robust, check for each subset M' of M of size at most k, if $G - M'$ has a perfect matching. It follows that k-Robust Bipartite Matching ℓ-Augmentation is in XP. For instance, for $k = \ell = 1$, the brute-force approach has a running time of $O(m^2 \cdot T_{\text{pm}}(m, n))$, where m is the size of G, n is the order of G and $T_{\text{pm}}(m, n)$ is the time required for computing a perfect matching in a bipartite graph of size m and order n (e.g., the Hopcroft-Karp algorithm yields $T_{\text{pm}}(m, n) = O(\sqrt{n}(m + n))$ [7]). It turns out that for $k = \ell = 1$ we can significantly improve upon the brute-force approach:

Theorem 5. 1-Robust Bipartite Matching 1-Augmentation *can be decided in time $O(mn + n^2)$.*

We give a brief overview of the algorithmic ideas. Let $I = \langle G, M \rangle$ be an instance of 1-Robust Bipartite Matching 1-Augmentation. We call an M-edge *1-redundant* if it is contained in an M-alternating cycle in the graph. Clearly, I is a Yes-instance if and only if we can enforce that each M-edge is 1-redundant after adding at most one edge to G. The following simple proposition tells us that we cannot consider M-edges in isolation when checking this property:

Proposition 6. *Let $v, w \in V$ be two non-adjacent vertices joined by an M-alternating path of odd length that starts with an M-edge. Then each M-edge on any M-alternating vw-path is 1-redundant in $G + vw$.*

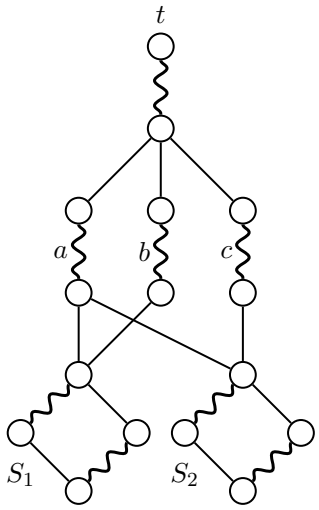


Figure 1: Illustration a graph and a perfect matching obtained by the reduction from Set Cover to 1-Robust Bipartite Matching ℓ-Augmentation. The corresponding Set Cover instance has two sets $\{S_1, S_2\}$, where $S_1 = \{a, b\}$ and $S_2 = \{a, c\}$.

In order to efficiently identify those M-edges which are not 1-redundant, we consider an *directed* auxiliary graph $G' = (V', A)$ whose construction somewhat similar that of the *alternating tree* used in Edmond's blossom algorithm. Let x be an arbitrary vertex of G. Note that this can be shown that for our purposes the choice of x is irrelevant if G is bipartite. We say that a vertex is *even* (*odd*), if it can be reached from x by an M-alternating path of even (odd) length. The vertex set V' of G' is the set of all even vertices of G. The arc set A is given by

$$A := \{uw \mid \exists u, w \in V',\, v \in V :\, uv \in M, vw \in E \setminus M\}.$$

It is not hard to see that each trivial strong component of G' corresponds to an M-edge that is not 1-redundant. Let us consider the condensation $C(G')$ of G', that is, the graph of strongly connected components of G'. If $C(G')$ contains no trivial strong components then M is 1-robust and therefore we have a Yes instance. Otherwise, our task is to identify a source s and a sink t of $C(G')$ such that every trivial strong component is on some st-path. This can be done performing two passes over the nodes of $C(G')$ in (reverse) topological order. By performing $O(n)$ of bookkeeping for each edge, we can identify sources and sinks that are connected to each trivial strong component. If there is at least one source and one sink that is connected to each trivial component then we have a Yes-instance, otherwise we have a No-instance. Due to our main result, Theorem 2, it is unlikely that we can extend our technique to arbitrary choices of ℓ.

We now illustrate by example how the reduction works that we used in the proof of Theorem 2. Given the set family $\{S_1, S_2\}$ on the ground-set $\{a, b, c\}$, where $S_1 = \{a, b\}$ and $S_2 = \{a, c\}$, we construct the graph and the perfect matching shown in Figure 1. The matching edges are the wiggly ones. Essentially, each of the edges labeled a, b, c corresponds to the item with the same label in the ground-set $\{a, b, c\}$. The 4-cycle containing the nodes labeled S_1 (S_2) corresponds to set S_1 (S_2) and is wired to the edges a, b, and c correspondingly. In the light of Proposition 6, for $i \in \{1, 2\}$, adding the edge tS_i corresponds to the selection of the set

S_i. It can be proved that any selection of ℓ additional edges that makes each matching edge 1-redundant corresponds to a set cover of size at most ℓ in the original instance. Therefore, the hardness results mentioned in the introduction follow. Note that adding an edge tS_i creates an alternating cycle of length six. This implies that the reduction from Set Cover can also be used in order prove to the hardness of 1-Robust s-Recoverable Bipartite Matching ℓ-Augmentation for any $s \geq 3$.

3 Conclusions and Open Problems

We have shown that, in the setting of perfect matchings in bipartite graphs, making a given perfect matching robust by augmentation is in general as hard as deciding if a graph admits a robust matching. Here is a selection of interesting topics for future investigations:

- Can we improve upon the brute-force approach to k-Robust Bipartite Matching ℓ-Augmentation for $k \neq 1$ and $\ell \neq 1$?
- Is 1-Robust Bipartite Matching ℓ-Augmentation W[2]-complete?
- How hard is k-Robust Bipartite Matching 0-Augmentation?

References

[1] David Adjiashvili, Viktor Bindewald, and Dennis Michaels. Robust Assignments via Ear Decompositions and Randomized Rounding. In *ICALP 2016*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 71:1–71:14, Dagstuhl, Germany, 2016.

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

[3] Christina Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012.

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

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

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

[7] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

Revenue maximization in Stackelberg Pricing Games: Beyond the combinatorial setting

Toni Böhnlein[1], Stefan Kratsch[2], and Oliver Schaudt[1]

[1]Universität zu Köln, Institut für Informatik, Weyertal 80, 50321 Köln,,
boehnlein@zpr.uni-koeln.de, schaudto@uni-koeln.de
[2]Universität Bonn, Institut für Informatik, Friedrich-Ebert-Allee 144, 53113 Bonn,
kratsch@cs.uni-bonn.de

In a Stackelberg Pricing Game a distinguished player, the *leader*, chooses prices for a set of items, and the other players, the *followers*, each seek to buy a minimum cost feasible subset of the items. The goal of the leader is to maximize her revenue, which is determined by the sold items and their prices. Most previously studied cases of such games can be captured by a combinatorial model where we have a base set of items, some with fixed prices, some priceable, and constraints on the subsets that are feasible for each follower. In this combinatorial setting, Briest et al. and Balcan et al. independently showed that the maximum revenue can be approximated to a factor of $H_k \sim \log k$, where k is the number of priceable items.

Our results are twofold. First, we strongly generalize the model by letting the follower minimize any continuous function plus a linear term over any compact subset of $\mathbb{R}^n_{\geq 0}$; the coefficients (or *prices*) in the linear term are chosen by the leader and determine her revenue. In particular, this includes the fundamental case of linear programs. We give a tight lower bound on the revenue of the leader, generalizing the results of Briest et al. and Balcan et al. Besides, we prove that it is strongly NP-hard to decide whether the optimum revenue exceeds the lower bound by an arbitrarily small factor. Second, we study the parameterized complexity of computing the optimal revenue with respect to the number k of priceable items. In the combinatorial setting, given an efficient algorithm for optimal follower solutions, the maximum revenue can be found by enumerating the 2^k subsets of priceable items and computing optimal prices via a result of Briest et al., giving time $O(2^k|I|^c)$ where $|I|$ is the input size. Our main result here is a W[1]-hardness proof for the case where the followers minimize a linear program, ruling out running time $f(k)|I|^c$ unless $\mathsf{FPT} = \mathsf{W}[1]$ and ruling out time $|I|^{o(k)}$ under the Exponential-Time Hypothesis.

1 Introduction

Pricing problems are fundamental in both economics and mathematical optimization. In this paper we study such pricing problems formulated as games, which are usually called *Stackelberg*

Pricing Games [10]. In our setting, in order to maximize her revenue one player chooses prices for a number of items and one or several other players are interested in buying these items. Following the standard terminology, the player to choose the prices is called the *leader* while the other players are called *followers*. Depending on the follower's preferences, computing optimal prices can be a computational non-trivial problem. In a setting where followers have valuations over individual items only, the problem is simple. If, however, valuations become more complex, e.g., over whole subsets of items, pricing problems become much harder–also in a formal sense.

Largely the literature has focused on what we call the *combinatorial setting*: there is a set Y of items and one follower seeks to buy a feasible subset. Some of the items have fixed costs, the others have prices that are chosen by the leader. If the follower buys a feasible subset $S \subseteq Y$ of the items, he has to pay the sum of the fixed costs of the elements of S, plus the leader's prices of the bought elements. The leader's revenue is the sum of the prices of the priceable items in S. This can also be captured by defining a solution space X containing 0/1-vectors corresponding to the feasible subsets S of Y. The goal of the follower is then to minimize a given additive function $f\colon X \to \mathbb{R}$ that depends on both fixed and leader-chosen prices.

So-called Stackelberg Network Pricing Games became popular when Labbé et al. [8] used them to model road toll setting problems. In this game, the leader chooses prices for a subset of *priceable* edges in a network graph while the remaining edges have fixed costs. Each follower has a pair of vertices (s, t) and wants to buy a minimum cost path from s to t, taking into account both the fixed costs and the prices chosen by the leader. The work of Labbé et al. led to a series of studies of the Stackelberg Shortest Path Game. Roche et al. [9] showed that the problem is NP-hard, even if there is only one follower, and it has later been shown to be APX-hard [2, 7]. More recently, other combinatorial optimization problems were studied in their Stackelberg pricing version. For example, Cardinal et al. [4, 5] studied the Stackelberg Minimum Spanning Tree Game, proving APX-hardness and giving some approximation results. Moreover, a special case of the Stackelberg Vertex Cover Game in bipartite graphs has been shown to be polynomially solvable by Briest et al. [3].

An important contribution to the study of Stackelberg Games was the discovery by Briest et al. [3]. They show that the optimal revenue can be approximated surprisingly well using a single-price strategy. For a single-price strategy the leader sets the same price for all of her priceable items. Basically, their result says the following: In any Network Pricing Game with k priceable items, there is some $\lambda \in \mathbb{R}_{\geq 0}$ such that, when assigning the price of λ to all priceable items at once, the obtained revenue is only a factor of H_k away from the optimal revenue. Here, $H_k = \sum_{i=1}^{k} 1/i$ denotes the *k-th harmonic number*. This discovery has been made independently, in a slightly different model, by Balcan et al. [1]. Actually, in both papers [1, 3] a stronger fact is proven: The single-price strategy yields a revenue that is at least R/H_k, where R is a natural upper bound on the optimal revenue. The definition of R was sketched in the example above, and is formally laid out later.

Our results. Our work focuses on pushing the knowledge on Stackelberg Pricing Games beyond the well-studied combinatorial setting, in order to capture more complex problems of the leader. This is motivated by the simple fact that the combinatorial setting is too limited to even model, e.g., a leader that has a minimum cost flow problem—a crucial problem in both, combinatorial optimization and algorithmic game theory. More generally, we might want to be able to give bounds and algorithms in the case when the follower has an arbitrary linear or even convex program. For example, the follower might have a production problem in which he needs

to buy certain materials from the leader, but such pricing problems haven't been discussed in the literature so far.

We prove an approximation result that applies even to a setting generalizing linear and convex programs. In our model, the follower minimizes a continuous function f over a compact set of feasible solutions $x \in X \subseteq \mathbb{R}^n_{\geq 0}$. For some of the variables, say x_1 up to x_k, the leader can choose a price vector $p \in \mathbb{R}^k$. Now the follower chooses a vector $x \in X$ that minimizes his objective function $f(x) + \sum_{i=1}^{k} p_i x_i$. We remark that if X is a set containing 0/1-vectors only, then we are back to the classical combinatorial. The result of Briest et al. can be transferred to the case when f is non-additive, in view of their original proof. Moreover if X is a polytope and f is additive, the follower minimizes a linear program, which is an important special case.

In the first part of the paper, we formally introduce this more general model and prove the following results.

(i) The maximum revenue obtainable by the leader can be approximated to a logarithmic factor using a single-price strategy. This generalizes the above mentioned result of Briest et al. [3] not only to linear programs but to any kind of follower that is captured by our model.

(ii) The analysis of point (i) is tight. There is a family of instances for which the single-price strategy yields maximum revenue. And this revenue meets the bound of point (i).

(iii) It is strongly NP-hard to decide whether one can achieve a revenue that is only slightly larger than the one guaranteed by the single-price strategy. This holds true even in a very restricted combinatorial setting.

The second part of the paper deals with the parameterized complexity of Stackelberg Pricing Games.To the best of our knowledge, the only result in this direction is an XP-algorithm by Cardinal et al. [5] for the Stackelberg Minimum Spanning Tree Game in graphs of bounded treewidth.

In contrast to structural parameters like the treewidth of the input graph, we consider the complexity of the pricing problem when parameterized by the number of priceable variables (or items in the combinatorial setting). Our main result in this part is a $\mathsf{W}[1]$-hardness proof for the case that the optimization problem of the follower is a linear program, which is arguably one of the most interesting cases that does not fit into the combinatorial setting. This rules out algorithms of running time $f(k)|I|^c$ unless $\mathsf{FPT} = \mathsf{W}[1]$ for any function f and polynomial $|I|^c$ of the input size; it also rules out running time $|I|^{o(k)}$ under the Exponential-Time Hypothesis of Impagliazzo et al. [6]. This intractability result is complemented by a fairly simple FPT-algorithm with running time $O(2^k|I|^c)$ for any Stackelberg Game that fits into the combinatorial model, when provided with an efficient algorithm for finding optimal follower solutions. The algorithm enumerates all subsets of priceable items and applies a separation argument of Briest et al. [3] to compute optimal leader prices and revenue.

References

[1] Maria-Florina Balcan, Avrim Blum, and Yishay Mansour, *Item Pricing for Revenue Maximization*, Proceedings 9th ACM Conference on Electronic Commerce (EC-2008), Chicago, IL, USA, June 8-12, 2008, 2008, pp. 50–59.

[2] Patrick Briest, Parinya Chalermsook, Sanjeev Khanna, Bundit Laekhanukit, and Danupon Nanongkai, *Improved Hardness of Approximation for Stackelberg Shortest-Path Pricing*, Internet and Network Economics, Springer, 2010, pp. 444–454.

[3] Patrick Briest, Martin Hoefer, and Piotr Krysta, *Stackelberg Network Pricing Games*, Algorithmica **62** (2012), no. 3-4, 733–753.

[4] J. Cardinal, E.D. Demaine, S. Fiorini, G. Joret, S. Langerman, I. Newman, and O. Weimann, *The Stackelberg Minimum Spanning Tree Game*, Algorithmica **59** (2011), 129–144.

[5] Jean Cardinal, Erik D Demaine, Samuel Fiorini, Gwenaël Joret, Ilan Newman, and Oren Weimann, *The Stackelberg Minimum Spanning Tree Game on Planar and Bounded-Treewidth Graphs*, Journal of combinatorial optimization **25** (2013), no. 1, 19–46.

[6] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane, *Which problems have strongly exponential complexity?*, J. Comput. Syst. Sci. **63** (2001), no. 4, 512–530.

[7] Gwenaël Joret, *Stackelberg Network Pricing is Hard to Approximate*, Networks **57** (2011), no. 2, 117–120.

[8] M. Labbé, P. Marcotte, and G. Savard, *A Bilevel Model of Taxation and Its Application to Optimal Highway Pricing*, Management Science **44** (1998), 1608–1622.

[9] S. Roche, G. Savard, and P. Marcotte, *An approximation algorithm for Stackelberg network pricing*, Networks **46** (2005), 57–67.

[10] H. von Stackelberg, *Marktform und Gleichgewicht*, Springer, 1934.

Independent Domination in Bipartite Cubic Graphs

Christoph Brause[1,2] and Michael A. Henning[2]

[1]TU Bergakademie Freiberg
[2]University of Johannesburg

A set S of vertices in a graph G is an independent dominating set of G if S is an independent set and every vertex not in S is adjacent to a vertex in S. The independent domination number of G, denoted by $i(G)$, is the minimum cardinality of an independent dominating set. In this talk, we study the following conjecture posed by Goddard and Henning [Discrete Math. 313 (2013), 839–854]: If $G \neq K_{3,3}$ is a connected, cubic, bipartite graph on n vertices, then $i(G) \leq \frac{4}{11}n$. As a consequence of a more general result due to Dorbec et al. [J. Graph Theory 80(4) (2015), 329–349], it is known that if G is a bipartite, connected, cubic, graph of order n that does not have an induced subgraph isomorphic to a supergraph of $K_{2,3}$, then $i(G) \leq \frac{3}{8}n$. In this talk, we improve this $\frac{3}{8}$-upper bound to a $\frac{4}{11}$-upper bound, thereby proving the Goddard-Henning conjecture when the graph does not have an induced subgraph isomorphic to a supergraph of $K_{2,3}$. This strengthens a result in [Discrete Applied Math. 162 (2014), 399–403] which proves the Goddard-Henning conjecture when the girth is at least 6.

1 Introduction

A *dominating set* of a graph G with vertex set $V(G)$ is a set S of vertices of G such that every vertex in $V(G) \setminus S$ is adjacent to a vertex in S; that is, every vertex outside S has a neighbor in S. The *domination number* of G, denoted by $\gamma(G)$, is the minimum cardinality of a dominating set. A set is *independent* in G if no two vertices in the set are adjacent. An *independent dominating set*, abbreviated ID-*set*, in G is a set that is both dominating and independent. Equivalently, an independent dominating set is a maximal independent set. The *independent domination number* of G, denoted by $i(G)$, is the minimum cardinality of an ID-set, and an ID-set of cardinality $i(G)$ in G is called an $i(G)$-*set*. Independent dominating sets have been studied extensively in the literature. (see, for example the so-called domination books [3, 4]). A recent survey on independent domination in graphs can be found in [2].

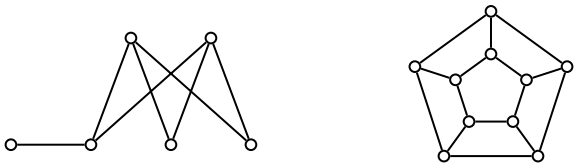
For notation and graph theory terminology we generally follow [3]. The *order* of G is denoted by $n(G) = |V(G)|$, and the *size* of G by $m(G) = |E(G)|$. We denote the *degree* of a vertex v in the graph G by $d_G(v)$. If every vertex in G has degree r, then G is called an *r-regular graph*. A 3-regular graph is commonly referred to as a *cubic graph* in the literature. The maximum (minimum) degree among the vertices of G is denoted by $\Delta(G)$ ($\delta(G)$, respectively).

The *girth* of G, denoted $g(G)$, is the length of a shortest cycle in G. A component of a graph G isomorphic to a graph F we call an *F-component* of G. For a set $S \subseteq V$, the subgraph induced by S is denoted by $G[S]$. Further, the subgraph of G obtained from G by deleting all vertices in S and all edges incident with vertices in S is denoted by $G - S$; that is, $G - S = G[V(G) \setminus S]$. A graph G is said to be *subcubic* if its maximum degree is at most 3. Let $n_j(G)$ denote the number of vertices of degree j in G. A vertex of degree 0 is called an *isolated vertex*.

For a simpler notation, let $[k]$ denote the set $\{1, 2, \ldots, k\}$ for some positive integer k.

1.1 Special Graphs

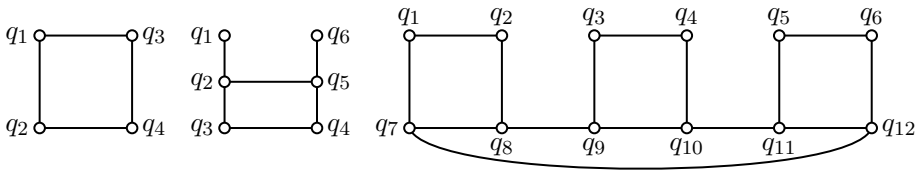
A *cycle* on n vertices is denoted by C_n and a *path* on n vertices by P_n. The *complete bipartite graph* with one partite set of size n and the other of size m is denoted by $K_{n,m}$. We denote the graph obtained from $K_{2,3}$ by adding a pendant edge to a vertex of degree 2 in $K_{2,3}$ by $K^+_{2,3}$. The 5-prism, $C_5 \,\square\, K_2$, is the Cartesian product of a 5-cycle with a copy of K_2. The graphs $K^+_{2,3}$ and $C_5 \,\square\, K_2$ are shown in Figure 1(a) and 1(b), respectively.



(a) $K^+_{2,3}$ (b) $C_5 \,\square\, K_2$

Figure 1: The graphs $K^+_{2,3}$ and $C_5 \,\square\, K_2$.

Let B_4, B_6 and B_{12} be the three graphs shown in Figure 2. We call these three graphs, and the graph $K_{3,3}$, "*bad graphs*." Further, we define a *bad component* of a graph G to be a component of G isomorphic to $K_{3,3}$, B_4, B_6 or B_{12}. For notational simplicity, we denote the vertices of a bad graph different from $K_{3,3}$ as in Figure 2.



(a) B_4 (b) B_6 (c) B_{12}

Figure 2: The "bad graphs" B_4, B_6 and B_{12}

Given a graph G, let $b_1(G)$ be the number of components of G isomorphic to B_6 or B_{12}, let $b_2(G)$ be the number of components of G isomorphic to B_4, and let $b_3(G)$ be the number of components of G isomorphic to $K_{3,3}$. Further, we let $b(G)$ denote the number of bad components of G, and so $b(G) = b_1(G) + b_2(G) + b_3(G)$.

2 Main Result

In this talk, we shall present an upper bound on the independent domination number of a subcubic, bipartite graph that does not have an induced subgraph isomorphic to $K_{2,3}^+$. We shall prove the following result.

Theorem 1. *If G is a subcubic, bipartite graph that does not have an induced subgraph isomorphic to $K_{2,3}^+$, then*

$$11i(G) \leq 11n_0(G) + 7n_1(G) + 5n_2(G) + 4n_3(G) + b_1(G) + 2b_2(G) + 9b_3(G).$$

As an immediate consequence of Theorem 1, we have our main result.

Theorem 2. *If $G \neq K_{3,3}$ is a connected, cubic, bipartite graph of order n that does not have an induced subgraph isomorphic to $K_{2,3}^+$, then $i(G) \leq \frac{4}{11}n$.*

2.1 Motivation and Known Results

As remarked in several papers [1, 5], the question of best possible bounds on the independent domination number of connected, cubic graphs remains unresolved. If we restrict our attention to bipartite, cubic graphs, we have the following result which was first observed in [2] (see also [5]).

Theorem 3. ([2]) *If G is a connected, cubic, bipartite graph of order n, then $i(G) \leq \frac{1}{2}n$ with equality if and only if $G = K_{3,3}$.*

If we relax the bipartite condition, then Lam, Shiu, and Sun [6] established the following upper bound on the independent domination number of a connected cubic graph. We remark that equality in Theorem 4 holds for the 5-prism, $C_5 \,\square\, K_2$.

Theorem 4. ([6]) *If $G \neq K_{3,3}$ is a connected, cubic graph on n vertices, then $i(G) \leq \frac{2}{5}n$.*

Dorbec et al. [1] proved the following result.

Theorem 5. ([1]) *If $G \neq C_5 \,\square\, K_2$ is a connected, subcubic graph of order n that does not have a subgraph isomorphic to $K_{2,3}$, then $8i(G) \leq 8n_0(G) + 5n_1(G) + 4n_2(G) + 3n_3(G)$.*

Our improvement of Theorem 5 consists of forbidding induced subgraphs isomorphic to $K_{2,3}^+$ instead of (not necessarily induced) subgraphs isomorphic to $K_{2,3}$.

Theorem 6. *If $G \notin \{C_5 \,\square\, K_2, K_{3,3}\}$ is a connected, subcubic graph of order n that does not have an induced subgraph isomorphic to $K_{2,3}^+$, then $8i(G) \leq 8n_0(G)+5n_1(G)+4n_2(G)+3n_3(G)$.*

As a special case of Theorem 6, every connected, cubic graph $G \notin \{C_5 \,\square\, K_2, K_{3,3}\}$ of order n that does not have an induced subgraph isomorphic to $K_{2,3}^+$ satisfies $i(G) \leq \frac{3}{8}n$. In this talk, our main result, namely Theorem 2, improves this $\frac{3}{8}$-upper bound to a $\frac{4}{11}$-upper bound for bipartite graphs. As motivation for our work the following conjecture on the independent domination number of a cubic graph is posed in [2].

Conjecture 7. (Goddard-Henning [2]) *If $G \neq K_{3,3}$ is a connected, cubic, bipartite graph of order n, then $i(G) \leq \frac{4}{11}n$.*

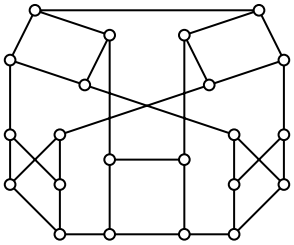


Figure 3: The bipartite cubic graph G_{22} with $i(G_{22}) = \frac{4}{11}n$.

Justin Southey [7] has confirmed by computer search that Conjecture 7 is true when $n \leq 26$. The upper bound in Conjecture 7 is achieved, for example, by the bipartite, cubic graph G_{22} of order $n = 22$ with $i(G_{22}) = 8$ shown in Figure 3.

The Goddard-Henning Conjecture was shown in [5] to be true if the girth is at least 6.

Theorem 8. ([5]) *If G is a cubic, bipartite graph of order n and of girth at least 6, then $i(G) \leq \frac{4}{11}n$.*

Theorem 1 strengthens the result of Theorem 8, and proves the Goddard-Henning Conjecture when the graph does not have an induced subgraph isomorphic to $K^+_{2,3}$.

References

[1] P. Dorbec, M. A. Henning, M. Montassier, and J. Southey, Independent domination in cubic graphs. *J. Graph Theory* **80**(4) (2015), 329–349.

[2] W. Goddard and M. A. Henning, Independent domination in graphs: A survey and recent results. *Discrete Math.* **313** (2013), 839–854.

[3] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, *Fundamentals of Domination in Graphs*, Marcel Dekker, Inc. New York, 1998.

[4] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, *Domination in Graphs: Advanced Topics*, Marcel Dekker, Inc. New York, 1998.

[5] M. A. Henning, C. Löwenstein, and D. Rautenbach, Independent domination in subcubic bipartite graphs of girth at least six. *Discrete Applied Math.* **162** (2014), 399–403.

[6] P. C. B. Lam, W. C. Shiu, and L. Sun, On independent domination number of regular graphs. *Discrete Math.* **202** (1999), 135–144.

[7] J. Southey, Domination results: Vertex partitions and edge weight functions. Ph.D Thesis, University of Johannesburg, May 2012.

A metaheuristic for the Minimum Gap Graph Partitioning Problem

Maurizio Bruglieri[1], Roberto Cordone[2], and Vincenzo Caurio[2]

[1]Politecnico di Milano
[2]Università degli Studi di Milano

We address the *min-sum* version of the *Minimum Gap Graph Partitioning Problem* through a Tabu Search metaheuristic. We also develop a Mixed Integer Linear Programming formulation to evaluate the quality of the solutions found by the Tabu Search on a set of benchmark instances properly built.

1 Introduction

The *Minimum Gap Graph Partitioning Problem* (*MGGPP*) is a graph partitioning problem [1, 2] introduced by [4]. Let $G = (V, E)$ be an undirected connected graph, w_v an integer *weight* coefficient defined on each vertex $v \in V$, and $p \leq |V|$ a positive integer number. Given a vertex subset $U \subseteq V$, we denote by $m_U = \min_{u \in U} w_u$ and $M_U = \max_{u \in U} w_u$ the minimum and maximum weight in U, respectively, and by *gap* their difference $\gamma_U = M_U - m_U$. The *MGGPP* consists in partitioning G into p vertex-disjoint connected subgraphs $G_s = (V_s, E_s)$, for $s = 1, \ldots, p$, with at least two vertices each. Its *min-sum* version consists in minimizing the sum of the gaps $f^{MS} = \sum_{s=1}^{p} \gamma_{V_s}$ over all subgraphs. The *MGGPP* can find applications, for example, in agriculture (divide a land into parcels with bounded height difference [3]) and in social network analysis (identify connected clusters of members with homogeneous features).

In [4], the computational complexity and the approximability of different versions of the *MGGPP* is investigated, also considering some special cases. In this work, we focus on the *min-sum* version of the *MGGPP* and, due to its NP-hardness, we propose a Tabu Search (*TS*) metaheuristic. Moreover, we develop a Mixed Integer Linear Programming (MILP) formulation to evaluate the quality of the solutions produced by the *TS* on a set of benchmark instances.

2 A Mixed Integer Linear Programming formulation

In this section we introduce a MILP formulation of the *MGGPP* where the solutions are represented as a spanning tree on an auxiliary directed graph $\tilde{G} = (V \cup \{r\}, A)$, being r a super-root node and $A = \{(i, j) : [i, j] \in E \text{ or } [j, i] \in E\} \cup \{(i, r) : i \in V\}$. Each connected subgraph of the solution (i.e., each component of the partition of G) is identified by a subtree appended to r, where the predecessor of r is the minimum weight node in each subtree. Therefore, the MILP consists in a multi-commodity flow formulation based on flow binary variables y_{ijk} $\forall (i, j) \in A$ and $k \in V$ equal to 1 if the flow of commodity k passes along arc (i, j), 0 otherwise;

binary variables x_{ij} equal to 1 if arc (i, j) belongs to the spanning tree, 0 otherwise; continuous variables γ_i, $\forall i \in V$ modeling the gap of the subgraph linked to r through node i, 0 otherwise.

$$\min \sum_{i \in V} \gamma_i \tag{1}$$

$$\sum_{(i,j) \in \delta^+(i)} y_{ijk} = \sum_{(i,j) \in \delta^-(i)} y_{jik} \quad \forall i \in V, k \in V : i \neq k \tag{2}$$

$$\sum_{(i,j) \in \delta^+(i)} y_{ijk} = 1 \quad \forall i \in V, k \in V : i = k \tag{3}$$

$$\sum_{i \in V} y_{irk} = 1 \quad \forall k \in V \tag{4}$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} = 1 \quad \forall i \in V \tag{5}$$

$$x_{ij} + x_{ji} \leq 1 \quad \forall (i, j) \in A \tag{6}$$

$$y_{ijk} \leq x_{ij} \quad \forall i \in V, (i, j) \in \delta^+(i), k \in V \tag{7}$$

$$y_{irk} = 0 \quad \forall i \in V, k \in V : w_k < w_i \tag{8}$$

$$w_k y_{irk} - w_i x_{ir} \leq \gamma_i \quad \forall i \in V, k \in V \tag{9}$$

$$\sum_{(j,i) \in \delta^-(i)} x_{ji} \geq x_{ir} \quad \forall i \in V \tag{10}$$

$$\sum_{i \in V} x_{ir} = p \tag{11}$$

where $\delta^+(i)$ and $\delta^-(i)$ denote the forward and the backward star of vertex i, respectively.

The objective function (1), to be minimized, is given by the sum of the gaps of each subgraph in the solution. Constraints (2) guarantee the flow conservation for each commodity k in each node $i \neq k$; (3) ensure that for each commodity k the total flow of this commodity going out from node k is 1; (4) impose that for each commodity k the super-root r can be reached from one and only one node $i \in V$. Constraints (5) guarantee that for each node $i \in V$ one and only one arc of its forward star is selected. Constraints (6) prevent the two-cycles. Coherency constraints (7) impose that if arc (i, j) is not in the solution then for no commodity k there can be a flow on this arc. Constraints (8) guarantee for each commodity k that the final node, the super-root r is reached through, has weight lower than w_k and thus is the minimum weight node of the subgraph. Constraints (9) enforce variables γ_i to model the gap of the subgraph linked to r through node i and (10) ensure that no subgraph in the solution is a singleton. Finally, constraint (11) guarantees that the solution is made up of p subgraphs.

3 A Tabu Search solution approach

The solution approach is based on two phases: in the first an initial solution is built through a *construction procedure*, while in a second it is improved by *Tabu Search* [6]. We apply this approach several times considering different initializations in the first phase. The construction procedure is inspired by Prim's algorithm for the minimum spanning tree problem [5] and it returns a spanning forest of p trees identifying a feasible solution for the *MGGPP*:

1. we order the edges $[i, j] \in E$ by non decreasing value of the gaps $|w_i - w_j|$;

2. we build an initial greedy matching X of cardinality p, selecting the edges in the given order and forbidding those adjacent to the already selected ones;

3. we repeat the following steps until all vertices are covered by X:
   - for each edge of the cut generated by the vertices covered by X we compute the total gap obtained if it were added to X;
   - we select the edge that minimizes the total gap and we add it to X.

Tabu Search is a well known metaheuristic approach introduced in [6] to enhance the performance of local search. Our implementation for the *MGGPP* exploits a very simple neighborhood, based on moving a single vertex from a subgraph of the current solution to another one. This neighborhood consists of at most $n(p-1)$ solutions. The feasibility requires three conditions: 1) the moved vertex must not be an articulation vertex for the original subgraph, i.e. its removal should not disconnect the subgraph; 2) the destination subgraph must be adjacent to the vertex; 3) the original subgraph must include at least three vertices. The feasible moves can be identified in $O(m)$ time for the whole neighborhood in the worst-case, due to the identification of the articulation vertices and of the subgraphs adjacent to each vertex. We test this kind of neighborhood since it is a natural building-block for more sophisticated moves.

The attribute used to characterize the tabu moves is the inclusion of a vertex v in a subgraph s for $s = 1, \ldots, p$. A suitable matrix L stores the most recent iteration l_{vs} in which vertex v has left subgraph s. A move that transfers vertex v into subgraph s is tabu until iteration $l_{vs} + l_{in}$, where l_{in} is a parametric value known as *tabu tenure*. At the beginning, all elements of matrix L are set to a very large negative value ($l_{vs} = -\infty$), so that all moves are nontabu. In order to intensify or diversify the search, the tenure parameter changes in an adaptive way based on the value assumed by the objective in the previous iteration: l_{in} decreases by 1 if the objective has improved, increases by 1 otherwise, remaining inside a prescribed range $\{l_{\min}, ..., l_{\max}\}$. The rationale is to intensify the search in more promising regions of the solution space and to diversify it in less promising ones. This algorithm is applied several times, replacing the initial greedy matching built at step 2 of the construction procedure with random ones.

4 Some numerical results

We built instances with four different sizes (n = 10, 20, 50, 100 vertices) and three different structures with respect to the density and topology: random graphs with $m = n(n-1)/6$ and $m = n(n-1)/3$ edges, and planar graphs obtained with a greedy triangulation of n points uniformly distributed at random in a square. The vertex weights are integer numbers uniformly distributed in $\{1, \ldots, 100\}$. The number of subgraphs p is set equal to $\ln(n)$, $\sqrt{n}$ or $n/\ln(n)$ rounded to the closest integer, in order to obtain solutions with few large subgraphs, balanced subgraphs, and with many small subgraphs, respectively. For each combination we consider 5 samples. In this way, we obtain 180 instances of the *MGGPP*.

Table 1 reports our results on the smaller instances (n = 10 and n = 20), most of which can be solved to optimality with a MILP solver. The first two columns report the number of nodes and the graph topology. The following four columns provide the average percent gap of the results, obtained on 15 instances (3 values of p and 5 samples), by the greedy algorithm (*Greedy*), the local search (*LS*), the tabu search algorithm without any restart (*TS*) and the tabu search algorithm restarted every 125 iterations (*TSr*), respect to the optimal MILP value computed with GUROBI 6.5.2. Both tabu search algorithms have $l_{\min} = 5$ and $l_{\max} = 10$

and are run for one minute on an Intel Core i3-3110M CPU with 2.40GHz and 4 GB. The greedy and the local search algorithm take at most 1 and 3 msec for the largest instances. The tabu search with restart finds all optimal solutions on the instances with $n = 10$ and on the denser ones with $n = 20$, while it strongly reduces the gap on the sparser instances. The restart seems to be a fundamental component to achieve this performance. The planar instances prove harder than the other ones, and this behavior is more evident as the instance size increases.

n	Topology	Greedy	LS	TS	TSr
10	$d = 0.3$	26.75%	14.73%	5.81%	0.00%
	$d = 0.6$	22.01%	22.01%	14.97%	0.00%
	planar	26.05%	13.14%	4.84%	0.00%
20	$d = 0.3$	35.68%	13.21%	11.64%	2.13%
	$d = 0.6$	13.31%	7.62%	6.34%	0.00%
	planar	42.74%	20.86%	18.73%	1.60%

Table 1: Results on the smaller instances

Table 2 reports our results on the larger instances ($n = 50$ and $n = 100$). Since for most of them GUROBI is unable to find either the optimum or a lower bound, we compare the results of the three algorithms listed above only with the starting greedy solution, reporting their percent improvement respect to the greedy solution value in the columns *LS*, *TS* and *TSr*. These results confirm that *TSr* outperforms the other algorithms. On the planar instances, the improvement is higher, probably due to a worse quality of the starting greedy solution.

n	Topology	LS	TS	TSr
50	$d = 0.3$	6.99%	7.60%	9.20%
	$d = 0.6$	1.56%	2.69%	5.94%
	planar	15.03%	20.10%	27.62%
100	$d = 0.3$	6.76%	8.20%	9.58%
	$d = 0.6$	2.72%	3.30%	3.75%
	planar	18.95%	25.52%	43.59%

Table 2: Improvements with respect to the greedy algorithm on the larger instances

References

[1] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, eds. *Graph Partitioning and Graph Clustering*, v. 588 of *Contemporary Mathematics*. AMS, 2013.

[2] C.-E. Bichot and P. Siarry, editors. *Graph Partitioning*. Wiley-ISTE, 2013.

[3] Li Xiao, Li Hongpeng, Niu Dongling, Wang Yan, and Liu Gang. Optimization of GNSS-controlled land leveling system and related experiments. *Transactions of the Chinese Society of Agricultural Engineering*, 31(3):48–55, 2015.

[4] M. Bruglieri, R. Cordone Partitioning a graph into minimum gap components. *Electronic Notes in Discrete Mathematics*, vol. 55: 33–36, 2016.

[5] R. C. Prim Shortest Connection Networks and some Generalizations. *Bell System Technical Journal*, vol. 36: 1389–1401, 1957.

[6] F. Glover Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operations Research*, vol. 13: 533–549, 1986.

Complexity of K-Adaptable Stochastic Programming*

Christoph Buchheim[1] and Jonas Pruente[1]

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

We address optimization problems with uncertain objective functions, given by discrete probability distributions. Within this setting, we investigate the so-called K-adaptability approach: the aim is to calculate a set of k feasible solutions such that the objective value of the best of these solutions, calculated in each scenario independently, is optimal in expectation. We show that this problem is NP-hard even if the underlying certain problem is trivial, and present further complexity results concerning approximability and fixed-parameter tractability.

1 Introduction

Consider an optimization problem of the form

$$\begin{array}{ll} \min & \xi^\top x \\ \text{s.t.} & x \in X \end{array} \tag{P}$$

where $X \subseteq \mathbb{Q}^n$ describes the set of feasible solutions. In practice, the objective function coefficients ξ in Problem (P) are often unknown or uncertain. As an example, consider the shortest path problem in a road network. In this case, the time needed to traverse an edge depends on the traffic situation, which is not know exactly in advance.

In the standard stochastic programming approach, one would thus replace the objective function $\xi^\top x$ of Problem (P) by its expected value $E(\xi^\top x)$, considering ξ a random variable. By the linearity of the expected value, this is equivalent to replacing every entry of ξ by its expected value, the problem is thus reduced to a certain problem, so that this approach is of limited theoretical interest. As a consequence, in stochastic programming, one is usually interested in uncertain constraints rather than uncertain objective.

In this paper, we investigate a new approach, motivated by the so-called K-adaptable robust optimization paradigm [1, 2]: we search for a set of k feasible solutions such that the expected objective value of the best of these solutions is optimal, where the best solution is selected independently in each scenario. The idea is thus to determine k feasible solutions $x_1, ..., x_k \in X$ before the scenario materializes, and then choose the best of them afterwards. Formally, this leads to a problem of the form

$$\begin{array}{ll} \min & E\big(\min\{\xi^\top x_1, \dots, \xi^\top x_k\}\big) \\ \text{s.t.} & x_1, \dots, x_k \in X\ . \end{array}$$

*This work has been supported by the German Research Foundation within the Research Training Group 1855.

In the following, we assume that the random variable ξ is discrete, i.e., we define a finite set of scenarios $S_1, \dots, S_l$ with probabilities $p_1, \dots, p_l \geq 0$ satisfying $p_1 + \dots + p_l = 1$, and corresponding objective vectors $\xi_1, \dots, \xi_l \in \mathbb{Q}^n$. The problem then becomes

$$\begin{array}{ll} \min & \sum_{j=1}^{l} p_j \min\{\xi_j^\top x_1, \dots, \xi_j^\top x_k\} \\ \text{s.t.} & x_1, \dots, x_k \in X\ . \end{array} \tag{mEm}$$

For a practical application, consider a supplier who serves the same companies every day. The time for using different routes depends on the current traffic situation. So he has to solve a vehicle routing problem every day, which due to its complexity is not possible quickly enough. Knowing the possible scenarios and how they effect his arrival time, and their probabilities, he could use the Min-E-Min idea to simplify his decision: for this, he has to solve Problem (mEm) only once, and every morning he just has to choose the best route for the day out of a small set of potential solutions. From a human user's point of view, it might be desirable to keep the number k small, even when a larger k could improve the objective value.

2 Preliminaries

Our main objective is to investigate the complexity of Problem (mEm), where the input consists of the probabilities $p_1, \dots, p_l$ and the corresponding objective vectors $\xi_1, \dots, \xi_l \in \mathbb{Q}^n$. By integrating the probability p_i into the vector ξ_i, we may assume that $p_i = 1$ for $i = 1 \dots, l$ in the following. We will consider both the situation where the number k of solutions is fixed and where it is part of the input.

For the set X of feasible solutions, we do not make any specific assumptions. The idea is that it is specified by means of an oracle for solving the certain problem (P), for any objective vector ξ. When proving hardness results, we will show that they hold even for very particular sets X, in these cases we always choose $X \subseteq \{0,1\}^n$. We address the following decision variant of Problem (mEm):

Definition 1. *The* Discrete Min-E-Min Decision Problem *is defined as follows: Given $X \subseteq \mathbb{Q}^n$ and objective vectors $\xi_1, ..., \xi_l \in \mathbb{Q}^n$ as well as $k, l \in \mathbb{N}$ and $b \in \mathbb{Q}$, decide whether there exist $x_1, \dots, x_k \in X$ such that $\sum_{j=1}^{l} \min\{\xi_j^\top x_1, \dots, \xi_j^\top x_k\} \leq b$.*

In this problem, each scenario ξ_j is "covered" by one of the solutions $x_1, \dots, x_k \in X$, namely the one where $\min\{\xi_j^\top x_1, \dots, \xi_j^\top x_k\}$ is attained. This leads to a partition of the set of scenarios into at most k parts. Conversely, if such a partition $\{1, \dots, l\} = \cup_{i=1}^{k} I_i$ is given, one can construct a solution $x_1, \dots, x_k$ by computing

$$x_i \in \operatorname{argmin}_{x \in X} \Big(\textstyle\sum_{j \in I_i} \xi_j \Big)^\top x$$

using the oracle for the certain problem (P). This shows that Problem (mEm) can be solved by calling the oracle for each possible k-partition of $\{1, \dots, l\}$. Moreover, it is easy to verify that it suffices to consider partitions without empty subsets. In particular, we obtain

Theorem 2. *The Discrete Min-E-Min Decision Problem can be polynomially reduced to the certain problem* (P) *if either the number of scenarios l or the difference $l - k$ is bounded.*

Proof. The number of partitions of $\{1, \dots, l\}$ into k non-empty subsets is the Sterling number of the second kind,

$$S_{l,k} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^l \ .$$

The problem is trivial if $k \geq l$, so when l is bounded, we may assume that k is bounded as well, and hence also $S_{l,k}$. For the second assertion, note that $S_{l,k} \in O(l^{2(l-k)})$. □

For the general case, we have

Theorem 3. *The Discrete Min-E-Min Decision Problem belongs to NP, if the underlying problem* (P) *belongs to NP.*

3 Hardness Results

We first consider the case where k is part of the input. We use the following NP-complete problem for our first hardness proof:

Definition 4. *Let $G(V, E)$ be a graph with node weights $w \in \mathbb{N}_0^V$ and edge lengths $l \in \mathbb{N}_0^E$ and $b, p \in \mathbb{N}_0$ integers with $p \leq |V|$. The* p-Median Decision Problem *asks whether there is a set $P \subseteq V$ of cardinality at most p such that $\sum_{v \in V} w(v) \cdot d_P(v) \leq b$. Here, $d_P(v)$ denotes the length of a shortest path from v to any point in P with respect to l.*

The NP-hardness of the p-Median problem (sometimes called *Min-Sum Multicenter Problem*) was proven by Kariv and Hakimi [4], using a reduction from Dominating Set. This reduction is a parameterized reduction showing that p-Median is actually $W[2]$-hard for the parameter p, and hence most likely not fixed-parameter tractable. The same then follows for the Discrete Min-E-Min Problem:

Theorem 5. *If k is part of the input, the Discrete Min-E-Min Decision Problem is NP-hard and $W[2]$-hard for parameter k. This remains true even if $|X|$ is polynomial.*

Proof. We construct a parametrized reduction from p-Median. The set X consists of all unit vectors in $\mathbb{Q}^V$. For every $v \in V$, we define a scenario $\xi_v \in \mathbb{Q}^V$ by $(\xi_v)_u := w(v) \cdot d(u, v)$. Finally, we set $k = p$. Now for any $P = \{u_1, \dots, u_k\} \subseteq V$ and $v \in V$ we have

$$w(v) \cdot d_P(v) = w(v) \cdot \min\{d(u_1, v), \dots, d(u_k, v)\} = \min\{(\xi_v)^\top e_{u_1}, \dots, (\xi_v)^\top e_{u_k}\} \ ,$$

from which the result follows. □

As shown by this proof, the hardness of the problem is not due to the structure of X, but due to the exponential number of possible partitions when k is not fixed. For fixed k and polynomial $|X|$, the problem can be solved in polynomial time by enumeration. However, if $|X|$ is not polynomial, the problem turns out to be hard again.

Theorem 6. *The Discrete Min-E-Min Decision Problem is NP-hard for each fixed $k \geq 3$, even if $X = \{0, 1\}^n$ and $\xi_j \in \{-1, 0, 1\}^n$ for every $j \in \{1, ..., l\}$.*

Proof. We use a reduction from k-Vertex Coloring, which is NP-complete for any $k \geq 3$ [3]. Given a graph $G(V, E)$, we define a scenario $\xi_v \in \mathbb{Q}^V$ for each node $v \in V$ by $(\xi_v)_u = -1$ if $u = v$, $(\xi_v)_u = 1$ if $\{u, v\} \in E$, and $(\xi_v)_u = 0$ otherwise. Now for given $x_1, \ldots, x_k \in \{0,1\}^V$ we have $\sum_{v \in V} \min\{\xi_v^\top x_1, \ldots, \xi_v^\top x_k\} \leq b := -|V|$ if and only if the sets $V_i := \{v \in V \mid (x_i)_v = 1\}$ for $i \in \{1, \ldots, k\}$ are independent sets covering V. This implies the result. □

Theorem 7. *The Discrete Min-E-Min Decision Problem with $k = 2$ fixed is NP-hard, even if $X = \{0,1\}^n$.*

Proof. We reduce the Max Cut Problem by defining the scenarios ξ_v as in the last proof, except that $(\xi_v)_v$ is now defined as $-\deg_G(v)$. Then $\sum_{v \in V} \min\{\xi_v^\top x_1, \xi_v^\top x_2\} \leq -2|E| + 2b$ if and only if the sets V_1 and V_2, defined as in the last proof, form a partition of V inducing a cut of cardinality at least $|E| - b$. □

As the certain problem (P) is tractable for $X = \{0,1\}^n$, the two latter results show that no oracle-polynomial time algorithm based on the underlying certain problem can exists for the Discrete Min-E-Min Problem, even if $k \geq 2$ is fixed, unless $P = NP$.

We finally note that the Min-E-Min Problem cannot even be approximated up to a constant factor unless $P = NP$, starting again with the case of unbounded k.

Theorem 8. *If k is part of the input, the Min-E-Min Problem is not in APX unless $P = NP$, even if X has polynomial size.*

Proof. Assume that there exists an α-approximation algorithm for the Min-E-Min Problem, for some $\alpha > 1$. Then we claim that Vertex Cover can be solved in polynomial time. Given $G(V, E)$ and $k \in \mathbb{N}$, we let X be the set of unit vectors in $\mathbb{Q}^V$ and define a scenario ξ_e for each $e \in E$ by $(\xi_e)_v = \frac{1}{\alpha}$ if $v \in e$ and $|E| + 1 - \frac{1}{\alpha}(|E| - 1)$ otherwise. Now if $U \subseteq V$ is a vertex cover of G, we have $\sum_{e \in E} \min_{u \in U}(\xi_e)_u \leq \frac{1}{\alpha}|E|$, while otherwise $\sum_{e \in E} \min_{u \in U}(\xi_e)_u \geq |E| + 1$, for large enough α. This shows the results. □

Adapting the proof of Theorem 6, one can show that k-Vertex Coloring can be decided by using any constant-factor approximation algorithm for the Min-E-Min Problem for the same k. This leads to

Theorem 9. *The Min-E-Min Problem with fixed $k \geq 3$ is not in APX unless $P = NP$, even for $X = \{0,1\}^n$.*

References

[1] D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010.

[2] C. Buchheim and J. Kurtz. Min-max-min robust combinatorial optimization. *Mathematical Programming (Series A)*. To appear. DOI: 10.1007/s10107-016-1053-z.

[3] M.R. Garey, D.S. Johnson, and L.J. Stockemeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[4] O. Kariv and S.L. Hakimi. An algorithmic approach to network location problems 2: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560, 1979.

Research on the Price of Connectivity for the vertex cover problem and the dominating set problem, with the help of the system GraphsInGraphs

Eglantine Camby[1,2] and Gilles Caporossi[2]

[1]Université Libre de Bruxelles (Belgium), ecamby@ulb.ac.be
[2]GERAD & HEC Montréal (Canada), gilles.caporossi@hec.ca

The vertex cover problem and the dominating set problem are two well-known problems in graph theory. Both hold a connected version, which imposes that the vertex subset must induce a connected component. To study the interdependence between the connected version and the original version of a problem, the Price of Connectivity (PoC) was introduced by Cardinal and Levy [7, 10] as the ratio between invariants from the connected version and the original version of the problem.

Some classes of PoC-Near-Perfect graphs, hereditary classes of graphs in which the Price of Connectivity is bounded by a fixed constant, have been already studied [5, 6]. To go further, we present for the vertex cover problem conjectures on these graphs with the help of the computer software GraphsInGraphs [4].

Moreover, Camby, Cardinal, Fiorini and Schaudt [5] introduced, for the vertex cover problem, the notion of critical graphs, graphs that can appear in the list of forbidden induced subgraphs characterization. By definition, the Price of Connectivity of a critical graph is strictly greater than that of any proper induced subgraph. In this paper, we prove that for the vertex cover problem, every critical graph is either isomorphic to a cycle on 5 vertices or bipartite. Moreover, for the dominating set problem, we investigate critical trees and we show that every minimum dominating set of a critical graph is independent.

1 Introduction

A *vertex cover* is a vertex subset X such that every edge of G has at least one endpoint in X. A *connected vertex cover* is a vertex cover X such that the induced subgraph $G[X]$ is connected. A *dominating set* of a graph G is a vertex subset X such that every vertex either is in X or has a neighbor in X. A *connected dominating set* of G is a dominating set X of G that induces a connected subgraph. In both cases, when G is not connected, we require that $G[X]$ has the same number of connected components as G. Table 1 fixes all notations about these notions.

In 1972, Karp identified 21 NP-hard problems, among which finding a minimum vertex cover of a graph. In 2008, Cardinal and Levy [7, 10] introduced the Price of Connectivity for the vertex cover problem. Lately, Camby, Cardinal, Fiorini and Schaudt [5] studied more in depth this new graph invariant. Besides, several researchers studied the interdependence between other graphs invariants.

Original version	VERTEX COVER PROBLEM	DOMINATING SET PROBLEM
value of the minimum size	$\tau(G)$	$\gamma(G)$
name of this value	vertex cover number	domination number
Connected version	VERTEX COVER PROBLEM	DOMINATING SET PROBLEM
value of the minimum size	$\tau_c(G)$	$\gamma_c(G)$
name of this value	connected vertex cover number	connected domination number
Price of Connectivity	VERTEX COVER PROBLEM	DOMINATING SET PROBLEM
value	$\frac{\tau_c(G)}{\tau(G)}$	$\frac{\gamma_c(G)}{\gamma(G)}$

Table 1: Notations for the vertex cover problem and the dominating set problem.

Zverovich [11] characterized, in terms of list of forbidden induced subgraphs, *PoC-Perfect graphs*, graphs for which the connected domination number and the domination number are equal for all induced subgraphs. Some years ago, Camby and Schaudt [6] translated the Price of Connectivity from the vertex cover problem to the dominating set problem and investigated it.

Recently, Belmonte, vant Hof, Kamiński and Paulusma [1, 2, 3] studied the Price of Connectivity for the feedback vertex set while Hartinger, Johnson, Milanič and Paulusma [8, 9] investigated the Price of Connectivity for cycle transversals.

2 Our results

2.1 The vertex cover problem

2.1.1 Conjectures on PoC-Near-Perfect graphs and critical graphs

With the help of the computer software GraphsInGraphs [4], we establish two new conjectures on *PoC*-Near-Perfect graphs and critical graphs. As a generalization of *PoC*-Perfect graphs, *PoC-Near-Perfect graphs* are graphs such that the Price of Connectivity of all induced subgraphs is bounded by a fixed constant, whereas, by definition, a *critical graph* has a Price of Connectivity strictly greater than that of all proper induced subgraphs.

Conjecture 1. *The following assertions are equivalent for every graph G:*

(i) For every induced subgraph H of G it holds that $\tau_c(H) \leqslant \frac{5}{3}\ \tau(H)$.

(ii) G is $(H_i)_{i=1}^{10}$-free, where graphs $H_1, \ldots, H_{10}$ are depiced in Figure 1.

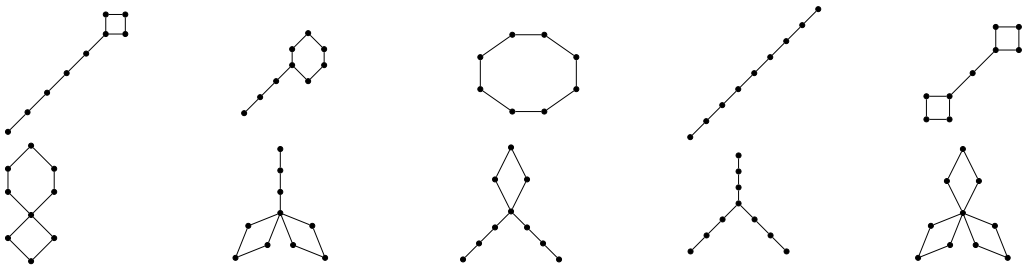


Figure 1: Graphs $H_1, \ldots, H_{10}$ from Conjecture 1.

Conjecture 2. *Every critical graph is a cactus.*

2.1.2 Critical graphs

Camby *& al.* [5] proved that every strongly critical graph, graph whose its price of connectivity is strictly greater than that of all proper (not necessarily induced) subgraphs, is bipartite. Here, we extend the result to the class of critical graphs, expect the cycle C_5 on 5 vertices.

Theorem 1. *A critical graph G is either isomorphic to C_5, or bipartite. Moreover, when G is bipartite, every minimum vertex cover of G is independent.*

2.2 Dominating set problem

2.2.1 Critical trees

Let T be a tree. We call T *special* if T is obtained from another tree (filled circle vertices in the example of Figure 2) by subdividing each edge either once or twice (hollow circle vertics) and then attaching a pendent vertex to every leaf of the resulting graph (square vertices).

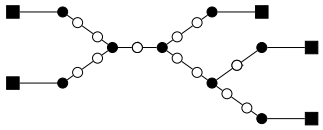


Figure 2: A special tree

The next result gives a partial characterization of the class of critical trees. However, the class of special trees turns out to be too restricted. We need a new definition.

We call a tree T *peculiar* if the neighbor of every leaf has degree 2, every minimum dominating set D of T is independent and every vertex $v \in V(T) \setminus D$ with degree at least 3 has only one neighbor in D, i.e. $|N_T(v) \cap D| = 1$. See Figure 3 for an example of peculiar tree where a minimum dominating set contains vertices indicated by filled circles and leaves are indicated by squares.

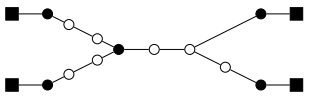


Figure 3: A peculiar tree.

Theorem 2. *For a tree G, the following assertions are equivalent:*

(i) G is a peculiar critical tree.

(ii) G is strongly critical.

(iii) G is critical.

Moreover, if G is critical and if the degree of any $v \in V(G) \setminus D$, where D is an arbitrary minimum dominating set of G, is at most 2, then G is a special tree built on an initial tree H, where $V(H)$ is a minimum dominating set.

Figure 4 illustrates the relations between graph classes: special trees, peculiar trees and critical trees. Not all peculiar trees are critical. For instance, the graph depicted in Figure 3 (whose Price of Connectivity is 12/5) is not critical because it contains as an induced subgraph the graph, with a higher Price of Connectivity, obtained from $K_{1,3}$ by subdividing each edge exactly thrice. Furthermore, the graph illustrated by Figure 5 is a peculiar critical tree which is not special. Also, we point out that not all special trees are critical, for instance P_8 contains an induced

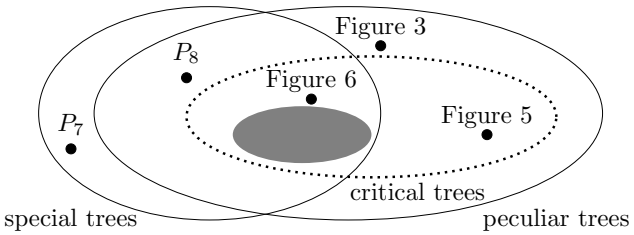


Figure 4: The situation around critical trees.

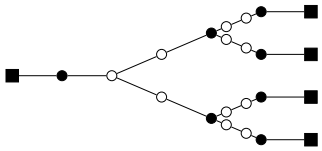


Figure 5: A peculiar critical tree, not special.

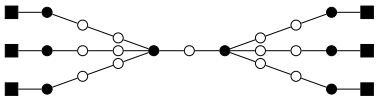


Figure 6: A special critical tree.

P_6 with the same Price of Connectivity.

Moreover, by Proposition 3, every special tree built on the initial tree H, where all edges of H are subdivided exactly twice in G, is critical. These graphs are represented by the gray area in Figure 4. However, the converse in the class of special trees is not true because the graph illustrated by Figure 6 is critical.

Proposition 3. *Let G be a special tree built on the initial tree H. If all edges of H are subdivided twice in G, then G is critical.*

2.2.2 Critical graphs

Theorem 4. *Every minimum dominating set of a critical graph is independent.*

Acknowledgements

This work was partially supported by a post-doc grant "Bourse d'Excellence WBI.WORD" from Fédération Wallonie-Bruxelles (Belgium).

References

[1] R. Belmonte, P. vant Hof, M. Kamiński, D. Paulusma, The price of connectivity for feedback vertex set, *The Seventh European Conference on Combinatorics, Graph Theory and Applications* (2013), pp. 123–128.

[2] R. Belmonte, P. vant Hof, M. Kamiński, D. Paulusma, Forbidden induced subgraphs and the price of connectivity for feedback vertex set, *International Symposium on Mathematical Foundations of Computer Science,* (2014), pp. 57–68.

[3] R. Belmonte, P. vant Hof, M. Kamiński, D. Paulusma, The price of connectivity for feedback vertex set, *Discrete Applied Mathematics 217*, (2017), pp. 132–143.

[4] E. Camby, G. Caporossi, Studying graphs and their induced subgraphs with the computer : GraphsInGraphs, *Cahiers du GERAD G-2016-10* (2016).

[5] E. Camby, J. Cardinal, S. Fiorini, O. Schaudt, The price of connectivity for vertex cover, *Discrete Mathematics & Theoretical Computer Science 16* (2014), pp. 207–224.

[6] E. Camby, O. Schaudt, The price of connectivity for dominating sets: upper bounds and complexity, *Discrete Applied Mathematics 177* (2014), pp. 53–59.

[7] J. Cardinal, E. Levy, Connected vertex covers in dense graphs, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques,* (2008), pp. 35–48.

[8] T.R. Hartinger, M. Johnson, M. Milanič, D. Paulusma, The price of connectivity for cycle transversals, *International Symposium on Mathematical Foundations of Computer Science* (2015), pp. 395–406.

[9] T.R. Hartinger, M. Johnson, M. Milanič, D. Paulusma, The price of connectivity for cycle transversals, *European Journal of Combinatorics 58* (2016), pp. 203–224.

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

[11] I.E. Zverovich, Perfect connected-dominant graphs, *Discussiones Mathematicae Graph Theory 23* (2003), pp. 159-162.

Inventory rebalancing in bike-sharing systems

Marco Casazza[1], Alberto Ceselli[1], and Roberto Wolfler Calvo[2]

[1]Università degli Studi di Milano, Dipartimento di Informatica, Italy.
[2]Université Paris 13, Laboratoire d'Informatique Paris Nord, France.

We address an optimization problem arising in rebalancing operations of inventory levels in bike-sharing systems. Such systems are public services where bikes are available for shared use on a short term basis. To ensure the availability of bikes in each station and avoid disservices, the bike inventory level of each station must met a forecast value. This is achieved through the use of a fleet of vehicles moving bikes between stations. Our problem can be classified as a Split Pickup and Split Delivery Vehicle Routing Problem. We propose a formulation in which routes are decomposed in smaller structures and we exploit properties on the structure of the optimal solutions, to design an exact algorithm based on branch-and-price.

1 Introduction

A bike-sharing system is a public service where bikes are available for shared use to individuals on a short term basis. Typically, users can pick up bikes at a cost and drop them back at designated stations widespread around the city. Such systems have been implemented around the world and are now present in hundreds of cities, as documented in [1].

Indeed the organization and management of the logistics of a bike-sharing system is challenging: for example, user behaviour results in an imbalance of the bike inventory in the stations over time, leading to undesired disservices such as empty departure stations or full destinations. To ensure the availability of service, one of the solutions chosen by many operators is to rebalance the bike inventory level of the stations by means of a fleet of dedicated trucks: the target inventory level of each station is forecast, and bikes are picked up from stations where congestion is expected, and delivered to those expected to become empty. Due to the high costs of running trucks in a urban environment, efficient rebalancing operations are a key factor for the success of the whole system.

Unfortunately, such operations rise very hard optimization problems. That is the case of the *Split Pickup and Split Delivery Vehicle Routing Problem (SPSDVRP)*: we assume that the network of the bike stations is given with both travel time and cost between each pair of stations. For each station we also know both the current and the target bike inventory levels. A fleet of homogeneous vehicles of limited capacity is given to perform rebalancing operations in such a way that the target inventory level is met for each station of the network. However, during the rebalancing process each station can be visited multiple times, even by the same vehicle. The SPSDVRP requires to find a route for each vehicle, that is a pattern

defining which stations are visited, the order of visits, and the amount of bikes loaded or unloaded at each station. Each vehicle always starts and ends at a depot with no bikes on board. Furthermore, each route cannot exceed a given time limit, that is the operator shift duration. We finally assume that no station is used as a temporary unloading location, meaning that during rebalancing operations no bikes are loaded from a delivery station or unloaded to a pickup station and therefore the number of bikes in each station is monotone. A feasible solution to the SPSDVRP consists of a set of routes respecting the above conditions. A solution is also optimal when the sum of the travelling costs of all vehicles is minimum.

From a methodological point of view, our SPSDVRP is NP-hard and belongs to the wide class of *Pickup and Delivery Vehicle Routing Problems (PDVRPs)* [2] and generalises the *Split Delivery Vehicle routing Problem (SDVRP)*. The problem of rebalancing bikes with a single vehicle has been addressed in [3], while a first mathematical approach for the SPSDVRP on a bike-sharing system has been proposed in [4]: the authors model the problem as a set partitioning extended formulation in which each variable represents a full vehicle route. However, such a formulation has two main drawbacks: first, it is not designed to handle travelling times, that are instead approximated by a limit on the number of visits in each route, and second, solving the continuous relaxation by means of column generation techniques is time consuming due to the complexity of the pricing procedure. We propose instead a new formulation that overcomes these two limitations, new theoretical properties on the structure of the optimal solutions, and a branch-and-price approach that solves to optimality instances with up to 20 nodes.

2 Model

The SPSDVRP on a bike-sharing system can be formalized as follows: a set of station nodes $N = \{1 \dots n\}$ is given, each with both the current and the target bike inventory levels $stock_i$ and $target_i$, respectively. When $stock_i > target_i$ we say that i is a *pickup* node, when $stock_i < target_i$ it is a *delivery* node, and when $stock_i = target_i$ i it is already balanced. The demand of each node $d_i = |stock_i - target_i|$ is the quantity of bikes to pick up from (resp. deliver to) that node.

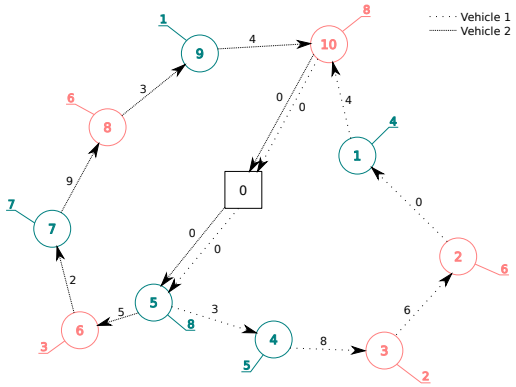
Let $G = (N_0, A)$ be a directed graph in which $N_0 = N \cup \{0\}$ is the set of nodes including the depot 0, and $A = \{(i, j) \mid i, j \in N_0\}$ is the set of arcs. Let c_{ij} and t_{ij} be the travelling cost and time of arc $(i, j) \in A$, respectively. W.l.o.g we assume that both costs and times satisfy triangular inequality.

A homogeneous fleet of vehicles $M = \{1 \dots m\}$ is given to satisfy station node demands. Each vehicle has a capacity C and a time resource T.

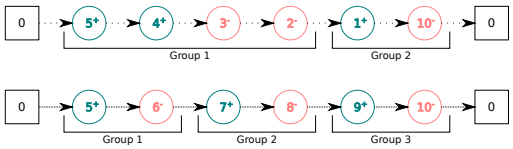
The SPSDVRP on a bike-sharing system is the problem of redistributing bikes in the network at minimum travelling cost, satisfying node demands while not exceeding neither vehicles capacity nor their time resource. Pickup and delivery nodes can be visited several times, either by the same vehicle or by different ones, and therefore their demand can be split.

3 Groups formulation and properties

The approach to the SPSDVRP proposed in [4] revealed that solving the continuous relaxation of such formulation was very challenging due to the structure of the pricing problem. That



(a) Example of solution on a graph having 10 nodes and 2 vehicles: each nodes is either a pickup ($^+$) or a delivery ($^-$). At each node it is attached a label reporting its demand. Each arc traelled by a vehicle has a label indicating the current amount of bikes on board.



(b) Group partitioning of the routes in solution in Figure 1(a): for each vehicle, each sequence of pickups and delivieries corresponds to a single group. Groups are concatenated to compose a route.

Figure 1: Example of SPSDVRP solution and its group representation.

motivated us to elaborate on a different approach, identifying particular regularities and properties of combinatorial substructures of the routes, and trying to reduce the complexity of the pricing problem by exploiting these properties.

We start from a simple observation:

Observation 1. *A route always starts with a sequence composed only by pickup nodes, always ends with a sequence composed only by delivery nodes, and in general always interleaves sequences of pickups followed by sequences of deliveries.*

Our intuition is therefore that the structure of a route can be much simplified by explicitly encoding such an interleaved behaviour. We formalize such an intuition denoting as *group* a sequence of one or more pickup nodes followed by a sequence of one or more delivery nodes and define a route as a sequence of concatenated groups. We then express a route cost and time in function of its groups. An example of routes and groups is depicted in Figure 1(b).

We then provide proofs that:

Theorem 1. *There always exists an optimal solution in which no node is visited more than once in the same group.*

Theorem 2. *For each pair of pickup (resp. delivery) nodes, there always exists an optimal solution in which they are visited at most once in the same group.*

Theorem 3. *Given the sets of nodes visited in each group of each vehicle, the problem of assigning the quantity loaded (resp. unloaded) at each station can be solved in polynomial time.*

4 Algorithms

We model our SPSDVRP as the problem of finding a minimum cost set of groups concatenations. Our formulation has an exponential number of variables, one for each group of each vehicle, and we recur to column generation techniques to solve its continuous relaxation.

Our pricing problem is a variant of *Resource Constrained Elementary Shortest Path Problem (RCESPP)* on a particular graph in which pickup nodes must be visited before delivery ones, and the profit collected at each node may be fractional. However, concerning the pricing problem we prove that:

Theorem 4. *There always exists an optimal solution in which there is at most one fractional pickup node (resp. delivery node) in the same group.*

Theorem 5. *The pickup (resp. delivery) node selected as fractional in each group, has a profit that is not greater than the less profitable non-fractional node visited in the same group.*

We solve our pricing problem by means of a label correcting algorithm in which each label is extended in three different ways: one collecting full profit, one collecting fractional profit, and one for when it is profitable to visit the node only without collecting any profit.

We included our CG procedure into a branch-and-price framework to achieve integrality of solutions, with heuristic pricers to speed-up CG, and a rounding heuristic to obtain good upper bounds to the value of the optimal solutions. We implemented our algorithms in C++ using SCIP framework and tested our methodology against datasets from the literature [4]. The results shown in Table 1 are promising: our algorithm solves to proven optimality all instances with 10 nodes in an average computing time smaller than 1 minute.

Instance	$\underline{z}$	$\bar{z}$	gap(%)	nodes	time(s)	Instance	$\underline{z}$	$\bar{z}$	gap(%)	nodes	time(s)
n10q10a	3719.00	3719	0.00	130	8.92	n10q10f	3037.00	3037	0.00	57	2.57
n10q10A	3055.00	3055	0.00	53	2.67	n10q10F	4097.00	4097	0.00	103	3.15
n10q10b	3353.00	3353	0.00	984	37.52	n10q10g	4179.00	4179	0.00	237	7.50
n10q10B	3745.00	3745	0.00	99	5.28	n10q10G	4221.00	4221	0.00	67	5.06
n10q10c	4239.00	4239	0.00	5	1.17	n10q10h	4194.00	4194	0.00	396	10.98
n10q10C	3392.00	3392	0.00	307	15.12	n10q10H	4118.00	4118	0.00	640	29.38
n10q10d	4472.00	4472	0.00	153	7.85	n10q10i	2523.00	2523	0.00	161	6.23
n10q10D	3307.00	3307	0.00	2771	72.46	n10q10I	3287.00	3287	0.00	72	2.16
n10q10e	3816.00	3816	0.00	870	54.63	n10q10j	3343.00	3343	0.00	150	6.76
n10q10E	4876.00	4876	0.00	154	5.22	n10q10J	3161.00	3161	0.00	169	3.78

Table 1: Results for instances with 10 nodes.

References

[1] P. DeMaio, *Bike-sharing: History, impacts, models of provision, and future*, Journal of Public Transportation, 12(4):4156, 2009.

[2] M. Battarra, J.-F. Cordeau, and M. Iori. *Pickup and delivery problems for goods transportation*, in Vehicle Routing: Problems, Methods, and Applications, Second Edition, chapter 6, pages 161–191, SIAM, 2014.

[3] D. Chemla, F. Meunier, R. Wolfler Calvo, *Bike sharing systems: Solving the static rebalancing problem*, Discrete Optimization, Volume 10, Issue 2, May 2013, Pages 120–146.

[4] M. Casazza, A. Ceselli, D. Chemla, F. Meunier, R. Wolfler Calvo, *The Multiple Vehicle Balancing Problem*, technical report, 2016, `https://hal.archives-ouvertes.fr/hal-01351215`.

Dynamic Cloudlet Assignment Problem: a Column Generation Approach

Alberto Ceselli[1], Marco Fiore[2], Marco Premoli[1], and Stefano Secci[3]

[1]Department of Computer Science, Università Degli Studi di Milano, Crema, Italy
[2]CNR-IEIIT, Torino, Italy
[3]UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France, e-mail: stefano.secci@upmc.fr

Major interest in network optimization is currently given to the integration of clusters of virtualization servers, also referred to as 'cloudlets', into mobile access networks for improved performance and reliability. Mobile access points (APs) are assigned (i.e., route their packets) to one or more cloudlets, with a cost in terms of latency for the users they provide connections to. Assignment of APs to cloudlet can be changed over time, with a cloudlet synchronization cost. We tackle the problem of the optimal assignment of APs to cloudlets over time, proposing dedicated mathematical models and column generation algorithms.

1 Model

Given a set of mobile access point sites (APs in the remainder), a set of virtualization server facility sites (cloudlets in the remainder) and the network connecting them, we aim at finding the best schedule for the assignment of APs to cloudlets over a planning time horizon so that the full AP demand is satisfied, no cloudlet capacity is exceeded and the management cost is minimum. Let A be a set of APs locations, K be a set of cloudlets and T be a set of time-slots in which the planning time horizon is split. For each $i \in A$ and $t \in T$, let d_i^t be the mobile traffic demand of AP i in time-slot t. For each $k \in K$, let C_k be the amount of demand that cloudlet k can handle in each time-slot and let $U \in [0, 1]$ represent the maximum allowed cloudlet utilization ratio. The assignment of an AP to a cloudlet implies a cost for the users connected to the AP in terms of communication latency, which is computed as the product of the demand traffic and the physical distance $m_{i,k}$ between AP $i \in A$ and cloudlet $k \in K$ in the network. Assignments can change over time, and each change implies a *switching* cost for the network, which is computed as the product of the demand traffic to be re-routed in the time-slot and the distance $l_{k',k''}$ between the pair of cloudlets $k', k'' \in K$ in the network. There is a trade-off between users' and network costs: to minimize the former an AP has to be assigned to its nearest cloudlet, while to minimize the latter an assignment to a distant cloudlet might be preferable, as long as it is not changing over time; let α and β be two non-negative parameters used to weight the relative importance of users and network costs.

We introduce two sets of variables: (i) variables $x_{i,k}^t$ model AP-cloudlet assignment, taking value 1 if AP $i \in A$ is assigned to cloudlet $k \in K$ in time-slot $t \in T$; and (ii) variables $y_{i,j,k}^t$

model the change of assignment in consecutive time-slots, taking value 1 if AP $i \in A$ is assigned to cloudlet $j \in K$ at time $(t-1) \in T$ and to cloudlet $k \in K$ at time $t \in T$.

We formulate our Dynamic Cloudlet Assignment Problem (DCAP) as follows:

$$\min \alpha \sum_{t\in T}\sum_{i\in A}\sum_{k\in K} d_i^t m_{ik} x_{ik}^t + \beta \sum_{t\in T}\sum_{i\in A}\sum_{\substack{(j,k)\in \\ K\times K}} d_i^t l_{jk} y_{ijk}^t \tag{1}$$

$$\text{s.t.} \sum_{i\in A} d_i^t x_{ik}^t \leq UC_k \qquad \forall t \in T, \forall k \in K \tag{2}$$

$$\sum_{k\in K} x_{ik}^t = 1 \qquad \forall i \in A, \forall t \in T \tag{3}$$

$$x_{ik}^t = \sum_{j\in K} y_{ijk}^t \qquad \forall i \in A, \forall t \in T \setminus \{1\}, \forall k \in K \tag{4}$$

$$x_{ik}^t = \sum_{j\in K} y_{ikj}^{t+1} \qquad \forall i \in A, \forall t \in T \setminus \{|T|\}, \forall k \in K \tag{5}$$

$$\boldsymbol{y} \in \{0,1\}, \boldsymbol{x} \in \{0,1\} \tag{6}$$

Objective function (1) minimizes the trade-off between users' and network costs. Constraints (2) impose that for each cloudlet the total demand assigned in each time-slot does not exceed a fraction U of its capacity; constraints (3) impose that the demand of each AP is completely assigned; constraints (4) and (5) link $\boldsymbol{x}$ and $\boldsymbol{y}$ variables. In the literature, both single or multi source assignment models are popular: in the former an AP is assigned to a single cloudlet in each time slot, in the latter the demand of an AP can be split and served by several cloudlets in the same time slot. We consider both possibilities: the single source conditions are enforced by keeping constraints (6), while in the multi source variant constraints on $\boldsymbol{x}$ variables are relaxed to $\boldsymbol{x} \in [0,1]$. We also remark that in the single source model, due to constraints (4) and (5), $\boldsymbol{y}$ variables automatically take integer values when integrality conditions on $\boldsymbol{x}$ are enforced.

2 Algorithm

In order to tackle data instances with large set of time-slots and large scale networks, we devised a column generation approach with a rounding strategy to restore integrality. Our Dantzig-Wolfe decomposition on model (1) – (6) yields to the following master problem:

$$\min \sum_{i\in A}\sum_{p\in\Omega^i} \left(\alpha \sum_{t\in T}\sum_{\substack{(j,k)\in \\ K\times K}} d_i^t l_{jk} \tilde{y}_{ijk}^{t,p} + \beta \sum_{t\in T}\sum_{k\in K} d_i^t m_{ik} \tilde{x}_{i,k}^{t,p} \right) z^p \tag{7}$$

$$\text{s.t.} -\sum_{i\in A}\sum_{p\in\Omega^i} d_i^t \tilde{x}_{ik}^{t,p} z^p \geq -UC_k \qquad \forall t \in T, \forall k \in K \tag{8}$$

$$\sum_{p\in\Omega^i} z^p = 1 \qquad \forall i \in A \tag{9}$$

$$z^p \geq 0 \tag{10}$$

where for each AP $i \in A$, Ω^i is the set of feasible *assignment paths*, that model the sequence of cloudlets to which the AP is assigned in the complete sequence of time-slots. For each path

$p \in \Omega^i$, let variable z^p take value 1 if path p is chosen to assign the related AP to the sequence of cloudlets. A path $p \in \Omega^i$ is encoded via binary parameters $\tilde{x}_{ik}^{t,p}$ and $\tilde{y}_{ijk}^{t,p}$, which behave as the corresponding variables $\boldsymbol{x}$ and $\boldsymbol{y}$. Objective function (7) aims in minimizing the management cost related to the chosen paths; (8) impose the maximum cloudlet utilization in a time-slot; (9) impose that for each AP a combination of assignment paths are chosen and hence that its demand is fulfilled in every time-slot.

Since formulation (7) – (9) contains a combinatorial number of variables, we optimize it with column generation techniques. Let $\lambda_{t,k}$ be the non-negative dual variables of constraints (8) and η_i be the free dual variables of constraints (9). The pricing problem is a shortest path for each AP. The corresponding formulation, given a fixed AP $\hat{i} \in A$, is the following:

$$\min \; -\eta_{\hat{i}} + \alpha \sum_{t \in T} \sum_{\substack{(j,k) \in \\ K \times K}} d_{\hat{i}}^t l_{jk} y_{\hat{i}jk}^t + \sum_{t \in T} \sum_{k \in K} \left(\beta d_{\hat{i}}^t m_{ik} + d_{\hat{i}}^t \lambda_{t,k}\right) x_{\hat{i},k}^t \tag{11}$$

$$\text{s.t.} \sum_{k \in K} x_{\hat{i}k}^t = 1 \qquad \forall t \in T \tag{12}$$

$$x_{\hat{i}k}^t = \sum_{j \in K} y_{\hat{i}jk}^t \qquad \forall t \in T \setminus \{1\}, \forall k \in K \tag{13}$$

$$x_{\hat{i}k}^t = \sum_{j \in K} y_{\hat{i}kj}^{t+1} \qquad \forall t \in T \setminus \{T\}, \forall k \in K \tag{14}$$

$$\boldsymbol{x} \geq 0, \boldsymbol{y} \geq 0 \tag{15}$$

This shortest path problem involves a directed layered graph $G(N, A)$, with $|T|$ layers, one for each time-slot. Each layer has one node for each cloudlet and one arc for each pair of nodes in consecutive layers. Each node $(t, k) \in T \times K$ has an associated cost given by $d_{\hat{i}}^t(\beta m_{ik} + \lambda_{t,k})$, while each arc connecting nodes (t, j) and $(t+1, k)$ has an associated weight given by $\alpha d_{\hat{i}}^{t+1} l_{j,k}$. This shortest path problem can be exactly solved with computational complexity $\mathcal{O}(TK^2)$ by means of dynamic programming.

Restoring Integrality In order to restore integrality, a rounding algorithm is executed at every CG iteration. Given the fractional solution $\tilde{S}$ of the CG master problem and the fractional variables values $\tilde{\boldsymbol{z}}$, we can compute the values of the corresponding $\tilde{\boldsymbol{x}}$ variables. For each time-slot $t \in T$ and for each AP sorted by descending highest fractional value of $\tilde{x}_{ik}^t$, the assignment is made with the cloudlet with enough residual capacity corresponding with the highest $\tilde{x}$. After each assignment the residual capacity of the chosen cloudlet is updated. Unfortunately, the rounding problem is a generalized assignment, which is APX-Hard: no a-priori guarantee on reaching feasibility is given by our algorithm, even if experimentally it proved to be successful in almost all CG iterations.

Greedy Initial Solution A simple greedy heuristic is used to initialize the master problem. It works as follows: for each time slot, APs are sorted by descending demand in the time slot and each AP is associated to a cloudlet chosen according to these rules: (i) take the cloudlet to which the AP was associated in the previous time-slot if the demand of the AP does not exceed its residual capacity and if it is not the first time-slot; (ii) find the nearest cloudlet for which the AP demand does not exceed the residual capacity, otherwise. A solution is provided in $\mathcal{O}(TA \log(A)K)$. Still no guarantees neither on quality nor on feasibility of the solution are given.

3 Computational Results

We implemented our algorithms in C++, using CPLEX 12.6 to solve the master LP subproblems, running tests on an Intel i7 4GHz workstation equipped with 32 GB of RAM. We created two datasets. The first one aims at reproducing realistic scenarii. We used a synthetic set of 1400 APs, whose coordinates are randomly drawn from two normal distributions, in order to model a metropolitan circular area with an higher density of APs in the city center. Ten clusters of APs were created with a standard k-means algorithm: their centers represent cloudlet locations and distances m_{ik} and l_{jk} were computed as euclidean distances accordingly. Planning time horizon was set to a single day. We experimented on four time discretizations: two hours, one hour, thirty minutes and fifteen minutes, corresponding to 12, 24, 48 and 96 time-slots, respectively. We drew mobile traffic demands for the fifteen-minute time-slots so that: (i) within a time-slot the distribution of APs demand follows a truncated heavy-tail power law distribution, and hence the majority of APs have low demand but a significant number of APs have high demand; (ii) the sum of all demands in a time-slot follows the standard daily activity profile, which shows a steep rise of the demand during morning rush-hour, followed by a stable rise until the peak of the evening rush-hour, that is in turn followed by a fall; and (iii) for the single AP, the change of demand during the day follows the same trend of the sum of demands. Demand for larger time slots has been obtained by aggregation. The second dataset aims at stressing our algorithms from a computational point of view. Demands were drawn uniformly at random for the fifteen-minute time slots, with no relationship between demands in consecutive time-slots. We set the demand of an AP in each time slot as the average of the demands in the fifteen-minute time-slots that are covered by it. Moreover for each demand matrix we computed five different instances by perturbing all demands with noise drawn uniformly at random in the interval $[-5\%, +5\%]$. Cloudlet capacity was set equal to $\left(\max_{t\in T} \sum_{i\in A} d_i^t/|K|\right) \cdot 1.05$. Finally, parameters α and β were both set to 0.5, while parameter U was set to 1.

Table 1 reports for each time-slot granularity (columns), and for the realistic and random datasets (rows), the mean, minimum and maximum values computed over the five perturbed instances of: (i) the execution time of our algorithm in seconds and (ii) the percentage gap between the final fractional solution and the best integer one found with rounding. We can notice that the realistic demands show better results both in terms of final gap and execution times. In particular, while in the realistic dataset the optimality gap is always lower than 1% and the finer discretization requires less than one minute of execution time, in the random dataset the gap is always higher than 1.5% and the execution takes up to several minutes. We also notice that on the realistic dataset, the optimality gap increases together with the number of time-slots, while an opposite behavior is observed on the random dataset. We impute this behavior to the smooth trend of realistic demands during the day, while random dataset involves sudden changes in consecutive time-slots.

	no. time-slots	12		24		48		96	
data type	stats.	gap	time	gap	time	gap	time	gap	time
Realistic	mean	0.96%	1.8s	0.70%	4.6s	0.61%	12.2s	0.42%	54.4s
	min	0.69%	1s	0.60%	4s	0.44%	11s	0.32%	51s
	max	1.31%	2s	0.85%	5s	0.73%	14s	0.48%	56s
Random	mean	1.77%	5s	1.89%	18s	2.25%	110.2s	2.53%	1165.6s
	min	1.69%	5s	1.63%	17s	2.13%	106s	2.42%	1098s
	max	1.88%	5s	2.05%	19s	2.41%	112s	2.67%	1244s

Table 1: Computational Results Statistics

Improved Space-efficient Linear Time Algorithms for Some Classical Graph Problems

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

[1]The Institute of Mathematical Sciences, HBNI, Chennai, India. sankardeep@imsc.res.in
[2]University of Siegen, Siegen, Germany. seungbum.jo@uni-siegen.de
[3]Seoul National University, Seoul, South Korea. ssrao@cse.snu.ac.kr

We provide space-efficient linear time algorithms for computing bridges, topological sorting, and strongly connected components improving on several recent results of Elmasry et al. [STACS'15], Banerjee et al. [COCOON'16] and Chakraborty et al. [ISAAC'16]. En route, we also provide another DFS implementation with weaker input graph representation assumption without compromising on the time and space bounds of the earlier results.

1 Introduction

Since the early days of designing graph algorithms, researchers have developed several approaches for testing whether a given undirected (or directed) graph $G = (V, E)$ with n vertices and m edges is (strongly connected) biconnected and/or 2-edge connected, and finding cut vertices and/or bridges of G. All of these methods use depth-first search (DFS) as the backbone to design the main algorithm. The classical linear time algorithms due to Tarjan [8, 9] computes the so-called "low-point" values (which are defined in terms of a DFS-tree of G) for every vertex v, and checks some conditions using that to determine whether G has the desired property. There are other linear time algorithms as well for these problems (see [7] and all the references therein). All of these classical algorithms take $O(m + n)$ time and $O(n)$ words (our model of computation is the standard word RAM model with word size $w = \Omega(\lg n)$ bits) of space. Our aim is to improve the space bounds of these algorithms without increasing the running time.

1.1 Motivation and Related Work

Motivated mainly by the "big data" phenomenon among others, recently there has been a surge of interest in improving the space complexity of the fundamental linear time graph algorithms by paying little or no penalty in the running time i.e., reducing the working space of the classical graph algorithms (which generally take $O(n \lg n)$ bits) to $o(n \lg n)$ bits without compromising on time. Towards this, Elmasry et al. [4] gave, among others, an implementation for DFS taking $O(m + n)$ time and $O(n \lg \lg n)$ bits of space. For sparse graphs (when $m = O(n)$), the space bound was improved further to $O(n)$ bits keeping the same linear time in [1]. Banerjee

Time	Space (in bits)	DFS	Testing biconnectivity & reporting cut vertices	Testing 2-edge connectivity & reporting bridges	Topological sort	Testing strong connectivity
$O(n+m)$	$O(n \lg n)$	[3]	[8]	[9]	[3]	[8]
$O(n+m)$	$O(n+m)$	[1, 6]	[1]	[1]	This paper	This paper
$O(n+m)$	$O(n \lg(m/n))$	[2]	[2]	[2]	This paper	This paper
$O(n+m)$	$O(n \lg \lg n)$	[4]	[6]	This paper	[4]	[4]

Table 1: Summary of our results.

et al. [1] gave, among others, a space efficient implementation for performing BFS using just $2n + o(n)$ bits of space and linear time, improving upon the result of [4]. Such algorithms for a few other graph problems also have been considered recently [2, 6].

1.2 Our Results

We assume that the input graph G, which is represented using *adjacency array* [1, 4, 2, 6], i.e., G is represented by an array of length $|V|$ where the i-th entry stores a pointer to an array that stores all the neighbors of the i-th vertex, is given in a read-only memory with a limited read-write working memory, and write-only output. We count space in terms of the number of bits in workspace used by the algorithms. Our main goal here is to improve the space bounds of some of the classical and fundamental graph algorithms. We summarize all our main results in Table 1. In this paper, basically we complete the full spectrum of results regarding the space bounds for these problems keeping the running time linear by providing/improving the missing/existing algorithms in the recent space efficient graph algorithm literature. Due to lack of space, we provide only sketches of our proofs.

2 Testing 2-Edge Connectivity and Finding Bridges

In an undirected graph G, a bridge is an edge that when removed (without removing the vertices) from a graph creates more components than previously in the graph. A (connected) graph with at least two vertices is 2-edge-connected if and only if it has no bridge. Let T denote the DFS tree of G. Following Kammer et al. [6], we call a tree edge (u, v) of T with u being the parent of v *full marked* if there is a back edge from a descendant of v to a strict ancestor of u, *half marked* if it is not full marked and there exists a back edge from a descendant of v to u, and *unmarked*, otherwise. They use this definition to prove the following: (i) every vertex u (except the root r) is a cut vertex exactly if at least one of the edges from u to one of its children is either an unmarked edge or a half marked edge, and (ii) root r is a cut vertex exactly if it has at least two children in T. Based on the above characterization, they gave $O(m+n)$ time and $O(n \lg \lg n)$ bits algorithm to test/report if G has any cut vertex. Our main observation is that we can give a similar characterization for bridges in G, and essentially using a similar implementation, we can also obtain $O(m + n)$ time and $O(n \lg \lg n)$ bits algorithms for testing 2-edge connectivity and reporting bridges of G. We start with the following lemma.

Lemma 1. *A tree edge $e = (u, v)$ in T is a bridge of G if and only if it is unmarked.*

Proof sketch: If e is unmarked, then no descendants of v reaches u or any strict ancestor of u, so deleting e would result in disconnected graph, thus e has to be a bridge. On the other direction, it is easy to see that if e is a bridge, it has to be an unmarked edge. □

Now we state our theorem below.

Theorem 2. *Given an undirected graph G, in $O(m+n)$ time and $O(n \lg \lg n)$ bits of space we can determine whether G is 2-edge connected. If G is not 2-edge connected, then in the same amount of time and space, we can compute and output all the bridges of G.*

Proof sketch: Using Lemma 1 and the similar implementation of using stack compression and other tools of the algorithm provided in Section 3.2 of Kammer et al. [6] with few modifications, we can prove the theorem. □

Note that the space bound of Theorem 2 improves the results of [1] and [2] for sufficiently dense graphs (when $m = \omega(n \lg \lg n)$ and $m = \omega(n \lg^{O(1)} n)$ respectively) while keeping the same linear runtime (see Table 1).

3 DFS without Cross Pointers

Banerjee et al. [1] and subsequently Kammer et al. [6] gave $O(m+n)$ bits and $O(m+n)$ time implementations of DFS improving on the bounds of [4] for sparse graphs. But both of these DFS implementations assume that the input graph is represented using the *adjacency array* along with *cross pointers* i.e., for undirected graphs, every neighbour v in the adjacency array of a vertex u stores a pointer to the position of vertex u in the adjacency array of v. See [4] for detailed definitions for directed graphs. We emphasize that this input assumption can double the space usage, compared to the raw adjacency array in worst case. In what follows, we provide the proof sketch of a DFS implementation taking the same time and space bounds as that of [1, 6] but without using the cross pointers. Our main theorem is as follows.

Theorem 3. *Given a directed or undirected graph G, represented as adjacency array, we can perform DFS traversal of G using $O(m+n)$ bits and $O(m+n)$ time.*

Proof sketch: We essentially modify the proof of [1] which uses a bitvector A of length $O(m+n)$ having one to one mapping with the unary encoding of the degree sequence to mark the tree edges, and subsequently uses cross pointers to find the parent of any vertex during backtracking as well as starting with next unvisited vertex after backtracking. We note that we can represent the parents of all the vertices in another bitvector P of length $O(m+n)$ (parallel to A). Now to perform backtracking efficiently, we could use the constant time *append only* structure (also with constant time rank/select) of Grossi et al. [5] along with the P array. With these modifications, we could get rid of cross pointers without compromising on the running time and space bound of the earlier algorithms. □

4 Testing Strong Connectivity and Topological Sorting

Towards giving improved space efficient algorithms for strong connectivity (SC) and topological sorting (TS), we first improve Lemma 4.1 of [4] which says the following: if DFS of a directed graph G takes $T(n,m)$ time and $S(n,m)$ space, then we can output the vertices of G in reverse postorder of the DFS tree T of G taking $O(T(n,m))$ time and $O(S(n,m) + n \lg \lg n)$ space. Combining this lemma with the classical algorithms for SC and TS [3] they obtained $O(n \lg \lg n)$ bits and $O(m+n)$ time algorithms for both these problems. We improve these by showing the following,

Theorem 4. *If DFS of a directed graph G takes $T(n,m)$ time and $S(n,m)$ space, then the vertices of G can be output in reverse postorder with respect to a DFS forest of G taking*

$O(T(n,m))$ time and $O(S(n,m)+m+n)$ space. As a result, we can also solve SC and TS in $O(m+n)$ time using $O(n+m)$ bits of space.

Proof sketch: We use the DFS algorithm of Theorem 3 to first mark all the tree edges in the array A. Now we start with the rightmost leaf vertex of the DFS tree and use rank/select operations [5] on A and P (as defined in the proof of Theorem 3) carefully to traverse the tree in reverse direction (along with standard DFS backtracking etc) to generate reverse postorder sequence. Now using this as the back bone of the classical algorithms, we obtain $O(m+n)$ bit and $O(n+m)$ time algorithms for SC and TS. □

Theorem 4 improves the result of [4] for sparse (when $m = O(n)$) graphs. Now if we use the DFS algorithm of Chakraborty et al. [2] and modify it suitably to perform the traversal of the DFS tree in reverse, we obtain the following result.

Theorem 5. *If DFS of a directed graph G takes $T(n,m)$ time and $S(n,m)$ space, then the vertices of G can be output in reverse postorder with respect to a DFS forest of G taking $O(T(n,m))$ time and $O(S(n,m)+n\lg(m/n))$ space. As a result, we can also solve SC and TS using $O(m+n)$ time and $O(n\lg(m/n))$ bits.*

References

[1] N. Banerjee, S. Chakraborty, and V. Raman. Improved space efficient algorithms for BFS, DFS and applications. In *22nd COCOON*, volume 9797, pages 119–130. Springer, LNCS, 2016.

[2] S. Chakraborty, V. Raman, and S. R. Satti. Biconnectivity, chain decomposition and st-numbering using O(n) bits. In *27th ISAAC*, volume 64 of *LIPIcs*, pages 22:1–22:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.*

[4] A. Elmasry, T. Hagerup, and F. Kammer. Space-efficient basic graph algorithms. In *32nd STACS*, pages 288–301, 2015.

[5] R. Grossi and G. Ottaviano. The wavelet trie: maintaining an indexed sequence of strings in compressed space. In *31st PODS*, pages 203–214, 2012.

[6] F. Kammer, D. Kratsch, and M. Laudahn. Space-efficient biconnected components and recognition of outerplanar graphs. In *41st MFCS*, 2016.

[7] J. M. Schmidt. A simple test on 2-vertex- and 2-edge-connectivity. *Inf. Process. Lett.*, 113(7):241–244, 2013.

[8] R. E. Tarjan. Depth-first search and linear graph algorithms. *SICOMP*, 1(2):146–160, 1972.

[9] R. E. Tarjan. A note on finding the bridges of a graph. *Inf. Pro. Lett.*, 2(6):160–161, 1974.

Exact Algorithms for Maximum Transitive Subgraph Problem

Sourav Chakraborty[1] and Nitesh Jha[2]

[1]Centrum Wiskunde & Informatica, Amsterdam, Netherlands
[1,2]Chennai Mathematical Institute, Chennai, India, e-mail: {sourav, nj}@cmi.ac.in

We study the problem of computing a *Maximum Transitive Subgraph (MTS)* of a given directed graph. This problem is known to be NP-hard. We give an algorithm that runs in time $O(4^{k^2}n^2)$ to output an MTS for a graph with treewidth k.

1. Introduction

Given a directed graph $G = (V, E)$, a subgraph S of G is said to be transitive if for every pair of edges $u \to v$ and $v \to w$ in S, the edge $u \to w$ is also present in S. S is called a *Maximum Transitive Subgraph (MTS)* if it is of the largest size (number of edges) possible. The same problem can also be posed in a weighted setting where edges have weights. Our goal in this article is to compute an MTS of a given graph.

The transitivity structure in a binary relation (directed graph) is a fundamental object that has a rich history in multiple areas of mathematics and computer science. Since transitivity is a desired structure, it is approached in multiple ways. Two most common are transitive closures and transitive subgraphs. The problems can then be posed in the form of an optimal or approximate solution. Problems have also been studied under the notion of *distance* from a transitive structure.

The problem of computing an MTS for a given graph is a well known NP-hard problem [10]. The recent work [2] gives a simple 0.25-approximation algorithm of obtaining an MTS in a general graph. For the case where the underlying undirected graph is triangle free, it gives a 0.874-approximation for the MTS problem. The idea there is to look at the related problem of directed maximum cuts in the same graph. We continue the study of algorithms for computing the MTS under different input restrictions.

Our main goal in this article is to understand the *parameterized complexity* of the MTS problem. A parameterization of a problem assigns an integer k to each input instance I and we say that a the problem is *fixed-parameter tractable* if there is an algorithm that solves the problem in time $f(k) \cdot |I|^{O(1)}$. Here, f is any computable function. First systematic study of parameterized complexity was done by Downey and Fellows [5]. More recent account of the field can be found in the texts [6, 7, 4].

Arnborg et al. [1] showed that the problem of MTS is fixed parameter tractable. They give an alternate proof of Courcelle's theorem [3] and express the MTS problem in *Extended Monadic Second Order*, thus giving a meta-algorithm for the problem. This algorithm is not explicit and f is known to be only a computable function.

Our main contribution in this paper is to prove that the problem of MTS is fixed parameter tractable by giving an explicit algorithm. For a given directed graph with treewidth at most k, we give an algorithm, for finding MTS, which runs in time $O(4^{k^2}n^2)$.

In Appendix A, we also give a poly-time algorithm for computing an MTS in a directed tree, and in Appendix B, we present an algorithm for finding MTS in a directed graph with treewidth at most k, in time $O(n^{k^2})$. Our main result (presented in Section 2) can be thought of as a generalization to these algorithms. Our main result is stated below.

Theorem 1. *There exists an algorithm that runs in time $O(4^{k^2}n^2)$ to output an MTS for a graph with treewidth k.*

There are explicit FPT algorithms known for related problems. For example, [8] shows that the problem of deciding whether a directed graph has a transitive induced subgraph of size k is fixed-parameter tractable. But there was no known explicit algorithm for the MTS problem parametrized by treewidth.

The related problem of Transitivity Editing is the problem of computing the minimum number of edge insertions or deletions in order to make the input digraph transitive. Weller et al [9] prove its NP-hardness and give a fixed-parameter algorithm that runs in time $O(2.57^k + n^3)$ for an n-vertex digraph if k edge modifications are sufficient to make the digraph transitive.

Notation

For any set S and $x \in S$, define $S - x = S \setminus \{x\}$. Let $G = (V, E)$ be a given directed graph. For $v \in V, e \in E$, define $G - v$ to be the graph obtained by removing the vertex v from G and $G \setminus e$ represents the graph obtained by deleting the edge e from G. The notation $F \subseteq G$ defines a subgraph F of G. For any subgraph F of G, $V(F)$ defines the vertex set of F and $E(F)$ defines the edge set of F. For $U \subseteq V$, $G(U)$ defines the induced subgraph on U.

For $A, B \subseteq V$, define $E(A, B) = \{u \to v : u \in A, v \in B\}$ and $\mathcal{E}(A, B) = E(A, B) \cup E(B, A)$. In the context of transitivity, we say that the two-path $u \to v \to w$ is *complete* if $u \to w \in E$. If $u \to w \notin E$, the two-path is called *incomplete*.

2. MTS is FPT Parameterised by Treewidth

We first introduce the basics of tree decomposition of an undirected graph $G = (V, E)$. We borrow the notations from [4]. Let $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ be a tree decomposition, where T is a tree whose every node t is assigned a vertex subset $X_t \subseteq V(G)$, called a bag, such that the following three conditions hold:

1. $\cup_{t \in V(T)} X_t = V(G)$.
2. $\forall$ edge $(u, v) \in E(G), \exists t \in T$ with $u, v \in X_t$.
3. $\forall u \in V(G)$, the set $T_u = \{t \in V(T) : u \in X_t\}$ induces a connected subtree of T.

Further, we use what is called a *nice* tree decomposition $\mathcal{T} = (T, \{X_t\}_{t \in V(T)})$ of G. Such a decomposition has the following properties.

1. The leaf and the root nodes are empty.
2. For every non-leaf node t is one of the following types:
   a) Introduce: t has exactly one child t' with $X_t = X_{t'} \cup \{u\}$ for some $u \notin X_{t'}$
   b) Forget: t has exactly one child t' with $X_t = X_{t'} \setminus \{u\}$ for some $u \in X_{t'}$
   c) Join: t has two children t' and t'' with $X_t = X_{t'} = X_{t''}$.

We perform a bottom up dynamic programming on $\mathcal{T}$ starting from the leaves and ending at the root. We describe the calculations performed at a node of any type (categorised above) using the computation performed at the children nodes. For any node $t \in \mathcal{T}$, denote by V_t the union of the bags associated with all the nodes in the subtree rooted at t, including X_t. Let G_t define the induced graph on V_t.

We define a table entry $m[t, F, I, O, U, Y]$ for each node $t \in \mathcal{T}$, for each transitive subgraph $F \subseteq E(X_t)$ and for each partition (I, O, U, Y) of X_t. The entry $M = m[t, F, I, O, U, Y]$ contains the MTS on $G(V_t)$ with the restriction that $G(X_t) \cap M = F$ and in M,

- no edges from the set $E(I, V_t \setminus X_t)$ are allowed,
- no edges from the set $E(V_t \setminus X_t, O)$ are allowed,
- any edge from the set $\mathcal{E}(V_t \setminus X_t, U)$ is allowed, and
- no edges from the set $\mathcal{E}(V_t \setminus X_t, Y)$ are allowed.

Vertices in U are said to be *unrestricted*. This partitioning has been defined to support the *Join* operation at a join node in the tree decomposition. The idea is same as before - avoid two-paths across separated partitions and avoid *incomplete* two-paths in the separator.

Leaf node: The only valid cell entry here is $m[t, \phi, \phi, \phi, \phi, \phi] = \phi$.

Introduce node: Suppose node t has child node t' such that $X_t = X_{t'} \cup \{v\}$. For any given partition (I, O, U, Y) of X_t and $F \subseteq G(X_t)$, we need to compute $m[t, F, I, O, U, Y]$.

First notice that the introduced vertex v can have edges to only the vertices in the set $X_{t'}$. Also, by definition, $MTS(G_t) \cap G(X_t) = F$. Applying both these conditions together, if v is not in $V(F)$, no edge incident on v can be included in the MTS of G_t. Hence, for $v \notin V(F)$,

$$m[t, F, I, O, U, Y] = m[t', F, I - v, O - v, U - v, Y - v]$$

Now we consider the case where $v \in V(F)$. For a vertex $r \in V(G)$ and $X \subseteq E(G)$, define the in-neighbours of r in X as $N^i_X(r) = \{s : s \to r \in X\}$ and the out-neighbours of r in X as $N^o_X(r) = \{s : r \to s \in X\}$. Define $N_X(r) = N^i_X(r) \cup N^o_X(r)$.

Consider the set $N_{X_t}(v) \setminus N_F(v)$. These neighbours of v in X_t, wherever they may lie in the partition (I, O, U, Y), can be kept as is in their designated partitions for recursion. The argument is as follows. Consider $u \in N_{X_t}(v) \setminus N_F(v)$. We want to check if any edge through u breaks transitivity. If $u \notin V(F)$, then u does not interact with any other vertex in X_t by definition and hence transitivity is maintained as before. If $u \in V(F)$, any edge in F incident on vertex u is already a part of a transitive set since F is transitive by definition.

We now deal with the set $N_F(v)$. Consider a vertex $u \in N_F(v)$. Following useful cases arise.

1. $u \in I \cap N^i_F(v)$: Here, an edge passing through u may break the transitivity. Such a vertex u must be removed from I and placed in O.
2. $u \in O \cap N^o_F(v)$: This is similar to the last case. We should move u from O to I.
3. $u \in U \cap N_F(v)$: Since the edges $\mathcal{E}(u, V_t \setminus X_t)$ are unrestricted to participate in an MTS, transitivity may break in two ways. Incomplete two-path of the form $r \to u \to v$ or $v \to u \to r$ where $r \in V_t \setminus X_t$ may result. We should disallow these cases.
4. $u \in Y \cap N_F(v)$: This case is fine as $\mathcal{E}(u, V_t \setminus X_t) = \phi$.

We incorporate all these restrictions in the following computation.

$$\begin{aligned}
F' &= F - v \\
I' &= (I \setminus N_F^i(v)) \cup (O \cap N_F^o(v)) \cup (U \cap N_F^o(v)) \\
O' &= (O \setminus N_F^o(v)) \cup (I \cap N_F^i(v)) \cup (U \cap N_F^i(v)) \\
U' &= U \setminus N_F(v) \\
Y' &= Y
\end{aligned}$$

The update method is then $m[t, F, I, O, U, Y] = m[t', F', I', O', U', Y'] \cup F$.

Forget Node: Suppose node t has child t' such that $X_t = X_{t'} \setminus \{v\}$. We update the current entry as follows:

$$m[t, F, I, O, U, Y] = \max m[t', F', I', O', U', Y']$$

where the maximum is over the following conditions:

$$F'|_{X_t} = F,\ I' = I,\ O' = O,\ Y' = Y,\ U' = U \cup \{v\}$$

Here, the transitive set F' is allowed to include the vertex v resulting in the condition $F'|_{X_t} = F$. We also allow v to have unrestricted edges since this effectively covers all the cases - only incoming edges on v, or only outgoing edges from v, or the case where v has both incoming and outgoing edges.

Join Node: Suppose node t has children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$. Define arbitrary partitions (to be fixed below) $X_{t_1} = I' \uplus O' \uplus U' \uplus Y'$ and $X_{t_2} = I'' \uplus O'' \uplus U'' \uplus Y''$. We have the following rule for updating the current entry:

$$m[t, F, I, O, U, Y] = \max(m[t_1, F, I', O', U', Y'] \cup m[t_1, F, I'', O'', U'', Y''])$$

under the restriction that: $I' \supseteq I, I'' \supseteq I$ and $O' \supseteq O, O'' \supseteq O$

For each vertex $v \in U$, one of the following is true: $v \in I' \cap I''$, or $v \in O' \cap O''$, or $v \in U' \cap Y''$, or $v \in Y' \cap U''$.

Here, we keep the F same in both t_1 and t_2 as required. In order to join at any vertex v in I, we demand such a vertex must be present in both I' and I'' but we also allow these sets to be larger. This is required as this leaves the possibility of a larger combination while transitivity is still maintained. A similar restriction is employed on O' and O''.

The vertices v in U are unrestricted but we need to be careful while using unrestricted vertices in the join operation. Such a vertex should only be allowed to be unrestricted on one side but completely isolated on the other side. This gives us the possibilities $v \in U' \cap Y''$ or $v \in Y' \cap U''$. But this restriction forbids the possibility of edges being used on both sides of v. Such a case could occur if v uses only incoming (or outgoing) edges on both the sides. To accommodate this, we have the options of $v \in I' \cap I''$ or $v \in O' \cap O''$. Finally, we take the maximum over all the legitimate join operations.

We now estimate the running time of our algorithm. Assuming the input graph has treewidth k, each node X_t is of size at most $k+1$. The number of partitions of type (I, O, U, Y) of X_t is at most 2^{k+4}. The number of transitive subgraphs F of X_t is at most 2^{k^2}. A single update of $m[\cdot]$ at any node can be done in at most n^2 steps. So a simple upper bound to the time complexity is $4^{k^2}n^2$.

3. Conclusion

In this article, we have continued the systematic study of computing a Maximum Transitive Subgraph of a given directed graph addressed recently in [2]. We show that this problem is fixed-parameter tractable when parameterized by treewidth. In particular, we give an algorithm that runs in time $O(4^{k^2}n^2)$ to output an MTS for a graph with treewidth k. An immediate question that arises is – whether we can reduce the exponent k^2 to $O(k)$.

Another interesting question that we have not addressed here is a lower bound for this problem. It would be interesting to arrive at any lower bound under the standard assumption of Exponential Time Hypothesis (ETH).

References

[1] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Problems easy for tree-decomposable graphs (extended abstract). In *Automata, Languages and Programming, 15th International Colloquium, ICALP88, Tampere, Finland, July 11-15, 1988, Proceedings*, pages 38–51, 1988.

[2] Sourav Chakraborty, Shamik Ghosh, Nitesh Jha, and Sasanka Roy. Maximal and maximum transitive relation contained in a given binary relation. In *Computing and Combinatorics - 21st International Conference, COCOON 2015, Beijing, China, August 4-6, 2015, Proceedings*, pages 587–600, 2015.

[3] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[4] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[5] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[6] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[7] Rolf Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford University Press, 2006.

[8] Venkatesh Raman and Somnath Sikdar. Parameterized complexity of the induced subgraph problem in directed graphs. *Inf. Process. Lett.*, 104(3):79–85, 2007.

[9] Mathias Weller, Christian Komusiewicz, Rolf Niedermeier, and Johannes Uhlmann. On making directed graphs transitive. In *Algorithms and Data Structures, 11th International Symposium, WADS 2009, Banff, Canada, August 21-23, 2009. Proceedings*, pages 542–553, 2009.

[10] Mihalis Yannakakis. Node- and edge-deletion np-complete problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 253–264, 1978.

Upper and lower bounds for the Swath Segment Selection Problem

Roberto Cordone[1], Giovanni Righini[1], and Andrea Taverna[2]

[1]Università degli Studi di Milano - Dipartimento di Informatica
[2]Università degli Studi di Milano - Dipartimento di Matematica

1 Problem definition

Polar orbiting satellites have a cameras pointed downwards which allows to monitor a strip of the Earth, named swath. The Earth's rotation implies that swaths monitored in north-to-south and south-to-north semi-orbits cross each other forming a sort of checkerboard. The *Swath Segment Selection Problem* (*SSSP*) models the problem of deciding which parcels of land should be monitored by the satellite and in which semi-orbit. For the sake of simplicity, we assume a time horizon in which a parcel can be monitored only in two opposite semi-orbits. The parcels can be modelled as cells (i, j) of a rectangular matrix, in which the set of m rows I correspond to the north-to-south semi-orbits and the set of n columns J corresponds to the south-to-north semi-orbits. A prize function $p : I \times J \to \mathbb{N}$ represents the value gained monitoring each parcel, and two weight functions $h : I \times J \to \mathbb{N}$ and $k : I \times J \to \mathbb{N}$ provide the amounts of memory required to monitor parcel (i, j) during the semi-orbits associated to row i and to column j; they can be different due to the shape of the parcel. The on-board memory and the downlink time available at the end of each semi-orbit limits the parcels that can be monitored in a semi-orbit; this is modelled by two capacity functions, $H : I \to \mathbb{N}$ for the rows and $K : J \to \mathbb{N}$ for the columns.

The *SSSP* can be formulated with two families of binary variables: if cell (i, j) is selected row-wise, $x_{ij} = 1$ and $y_{ij} = 0$; if it is selected column-wise, $y_{ij} = 1$ and $x_{ij} = 0$; if it is not selected, $x_{ij} = y_{ij} = 0$:

$$\max z = \sum_{i \in I} \sum_{j \in J} p_{ij} (x_{ij} + y_{ij}) \tag{1a}$$

$$\text{s.t.} \sum_{j \in J} a_{ij} x_{ij} \leq A_i \qquad i \in I \tag{1b}$$

$$\sum_{i \in I} b_{ij} y_{ij} \leq B_i \qquad j \in J \tag{1c}$$

$$x_{ij} + y_{ij} \leq 1 \qquad i \in I, j \in J \tag{1d}$$

$$x_{ij}, y_{ij} \in \{0, 1\} \qquad i \in I, j \in J \tag{1e}$$

The *SSSP* has been solved with a branch-and-bound based on the continuous relaxation of (1) in [6], and with a branch-and-bound based on the Lagrangean relaxation of constraints (1d) in [3]. A similar model has been proposed and treated with Constraint Programming in [1]. In this paper, we discuss for the first time the complexity and approximability

of the *SSSP*, we classify a number of Lagrangean relaxations and decompositions, establishing some dominance relations between them, and we make numerical experiments on two of them: the former is a Lagrangean relaxation woth the integrality property, which allows a very efficient multiplier update mechanism; the latter is theoretically stronger. Finally, we discuss a meta-heuristic approach based on the *Variable Neighborhood Descent* (*VND*) and *Variable Neighborhood Search* (*VNS*) frameworks.

2 Complexity and approximability

Theorem 1. *The* SSSP *is $\mathcal{NP}$-complete in the strong sense.*

Remark 2. *The* SSSP *admits an obvious exhaustive algorithm of $O(mn\,3^{mn})$ complexity.*

Theorem 3. *The* SSSP *can be solved in $O\big(mn \min\big(\prod_{j\in J} K_j \max_{i\in I} H_i, \prod_{i\in I} H_i \max_{j\in J} K_j\big)\big)$ time.*

Both algorithms are impractical, unless for very small matrices or very tight capacity constraints.

Theorem 4. *The* SSSP *can be 4-approximated in polynomial time and 2-approximated in pseudo-polynomial time.*

Owing to space limitations, proofs are omitted.

3 Lagrangean relaxations and decompositions

Formulation (1) admits several possible Lagrangean relaxations and decompositions. Relaxing one or more of its three families of constraints yields 7 different Lagrangean relaxations, which can be denoted as LR_s, where string s includes one or more of the following symbols, according to the relaxed families of constraints: R for the row-knapsack constraints (1b), C for the column-knapsack constraints (1c) and D for the disjunctive contraints (1d).

Relaxation LR_D was applied in [3] to obtain the best known current algorithm for the *SSSP*. We here consider the other relaxations. First of all, of course, LR_s dominates $LR_{s'}$ whenever it relaxes a smaller family of constraints (in short, $s \subseteq s' \Rightarrow LR_s \preceq LR_{s'}$). Then, the integrality property [5] allows to prove that LR_{RC}, LR_{RCD} and the continuous relaxation CR are all reciprocally equivalent and weaker than the other ones. The Lagrangean subproblem LR_{RC} can be solved by simple inspection to obtain an extremely fast upper bound:

$$z(\lambda,\mu) = \max_{x,y}\left[\sum_{i\in I}\sum_{j\in J}(p_{ij}-\lambda_i h_{ij})\,x_{ij} + \sum_{i\in I}\sum_{j\in J}(p_{ij}-\mu_j k_{ij})\,y_{ij}\right] + \sum_{i\in I}\lambda_i H_i + \sum_{j\in J}\mu_j K_j \quad (2a)$$

$$\text{s.t. } x_{ij} + y_{ij} \le 1 \qquad i\in I, j\in J \quad (2b)$$

$$x_{ij}, y_{ij} \in \{0,1\} \qquad i\in I, j\in J \quad (2c)$$

The upper bound $z(\lambda,\mu)$ is a piecewise linear function of the λ_i and μ_j multipliers. The dual problem of tuning them so as to minimize the upper bound enjoys a nice structural property: for fixed values of all λ_i, $z\left(\bar{\lambda},\mu\right)$ is a sum of independent convex piecewise linear functions in the μ_j multipliers. Conversely, for fixed values of all μ_j, $z(\lambda,\bar{\mu})$ is a sum of independent convex piecewise linear functions in the λ_i multipliers. This allows the following very efficient strategy:

- fix a starting value for the row multipliers λ_i $(i \in I)$
- for each column $j \in J$, enumerate the breakpoint values in which the derivative of $z(\lambda, \mu)$ changes, corresponding to a change in the optimal solution of the Lagrangean subproblem (2)
- select independently for each column multipliers μ_j $(j \in J)$ the breakpoint which minimizes $z(\lambda, \mu)$;
- fix the column multipliers μ_j to the optimal values found and apply the process to the row multipliers λ_i $(i \in I)$; then, go back to the starting point.

This procedure does not actually guarantee to reach the optimal bound, but our experiments show that it is order of magnitude faster than the subgradient procedure used in [3] to optimize the LR_D relaxation and numerically much stabler. In a relevant minority of instances, it can even provide a better bound in practice.

Lagrangean decomposition can be applied to the *SSSP* by duplicating the binary variables and relaxing the coupling constraints $x_{ij} = x'_{ij}$ and $y_{ij} = y'_{ij}$. Referring to the three families of constraints, it is possible to obtain 3 Lagrangean decompositions, in which two families of constraints use the original variables and the third one uses the new ones. We denote them by $LD_{RC|D}$, $LD_{RD|C}$ and $LD_{CD|R}$.

Thanks to the integrality property, the $LD_{RC|D}$ decomposition is equivalent to the LR_D relaxation investigated in [3]. The other two decompositions can both be strengthened by reintroducing the disjunctive constraints $x'_{ij} + y'_{ij}$ also in the second subproblem. The result is the same in the two cases, and we denote it as $LD_{RD|CD}$:

$$z(\lambda, \mu) = \max_{x,y} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} [(p_{i,j} - \lambda_{ij}) x_{ij} + (p_{ij} - \mu_{i,j}) y_{ij}] \right\} + \max_{x',y'} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} (x'_{ij} \lambda_{ij} + \mu_{ij} y'_{ij}) \right\} \tag{3a}$$

$$\text{s.t. } x_{i,j} + y_{i,j} \leq 1 \qquad i \in I, j \in J \tag{3b}$$

$$\sum_{j=1}^{m} h_{i,j} x_{i,j} \leq H_i \qquad i \in I \tag{3c}$$

$$x'_{i,j} + y'_{i,j} \leq 1 \qquad i \in I, j \in J \tag{3d}$$

$$\sum_{j=1}^{m} k_{i,j} y'_{i,j} \leq K_j \qquad j \in J \tag{3e}$$

$$x'_{ij}, y'_{ij} \in \{0, 1\} \qquad i \in I, j \in J \tag{3f}$$

This decomposition is theoretically stronger than the Lagrangean relaxations LD_{RD} and LD_{CD} [5], and therefore stronger than all Lagrangean relaxations except LR_D. To the best of our understanding, $LD_{RD|CD}$ is neither stronger nor weaker than LR_D.

4 A *VNS* meta-heuristic

We developed a local search meta-heuristic to determine lower bounds for the *SSSP*. The algorithm starts from the solution provided by the 2-approximation pseudo-polynomial algorithm

described in Theorem 4, where the knapsack subproblems associated to rows and columns are independently solved to optimality with the code by Ceselli et al. [2]. Then, feasibility violated by doubly selected cells (i, j) is retrieved keeping the row-wise selection when $h_{ij} < k_{ij}$, the column-wise selection otherwise; the resulting residual capacity is exploited by a greedy heuristic, which iteratively identifies the unselected cell with the highest value of $\max(p_{ij}/h_{ij}, p_{ij}/k_{ij})$ and selects it row-wise or column-wise according to which of the two ratios dominates.

The local search procedure considers several different neighborhoods. A basic neighborhood $\mathcal{N}_1$ includes all solutions obtained by selecting row-wise or column-wise a currently unselected cell. There are at most $O(mn)$ such solutions and all of them are improving. By $\mathcal{N}_2^{(d)}$, we denote the set of all solutions obtained by changing the selection of cells (i, j) and $(i, j + d)$ or (i, j) and $(i + d, j)$, where the cell indices are computed modulo n and m. This yields $m + n$ different neighborhoods consisting of $O(mn)$ solutions each.

Following the *VND* framework, we explore each $\mathcal{N}_2^{(d)}$ neighborhood as long as it provides improving solutions. Each neighborhood is explored with first-improve strategy to maximize the search speed. The $\mathcal{N}_1$ neighborhood is also explored at each iteration. When a neighborhood no longer provides better solutions, d is increased and the search is restarted.

When none of the neighborhoods proves useful, we resort to the *VNS* algorithm, applying a shaking procedure that select k random cells and randomly modifies their selection status, always respecting the capacity constraints. Then, the search starts again from the resulting solution. At the end of each *VND* run, the current local optimum is compared to the best known and the next shaking procedure restarts from the possibly updated incumbent lower bound.

Computational tests on 336 instances of the dataset used in [3] show that both the VND and the VNS algorithms provide near-optimal solutions (within 0.21% from optimality for the largest instances, with 500 rows and columns) in few minutes.

References

[1] I. Abi-Zeid, O. Nilo, and L. Lamontagne. A constraint optimization approach for the allocation of multiple search units in search and rescue operations. *INFOR: Information Systems and Operational Research*, 49(1):15–30, 2011.

[2] A. Ceselli and G. Righini. A branch-and-price algorithm for the capacitated p-median problem. *Networks*, 45(3):125–142, 2005.

[3] R. Cordone, F. Gandellini, and G. Righini. Solving the swath segment selection problem through lagrangean relaxation. *Computers and Operations Research*, 35(3):854–862, March 2008.

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

[5] M. Guignard. Lagrangean relaxation. *Top*, 11(2):151–228, December 2003.

[6] R. Knight and B. Smith. Optimal nadir observation scheduling. In *Proceedings of the Fourth International Workshop on Planning and Scheduling for Space (IWPSS 2004)*, Darmstadt, Germany, June 23–25th 2004. ESA–ESOC.

Robust single machine scheduling with a flexible maintenance activity

Paolo Detti[1], Gaia Nicosia[2], Andrea Pacifici[3], and Garazi Zabalo Manrique de Lara[1]

[1]Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Università di Siena, Italia
[2]Dipartimento di Ingegneria, Università "Roma Tre", Italia
[3]Dipartimento di Ingegneria Civile e Ingegneria Informatica, Università di Roma "Tor Vergata", Italia

1 Introduction

In this paper we address a problem arising in a manufacturing environment where a set of jobs must be scheduled together with the execution of a preventive maintenance activity. In particular, we consider the so-called *flexible maintenance*, in which the starting time of the maintenance task is determined in the scheduling decision process. This problem has been introduced in [4] but in a different applicative context. Maintenance must be performed within a given time window and its duration is uncertain and can only be estimated. Hence, when planning the schedule of the jobs, one must determine schedules which are *robust* to any possible deviations from the nominal value of the duration of the maintenance activity.

Several different versions of scheduling problems considering maintenance activities, which are usually modelled as machine unavailability, have been addressed in the literature. For a survey see [5]. In [7] the authors deal with the problem of scheduling a maintenance activity, within a given time interval, minimizing total jobs' completion time. They show that the problem—which was proven to be NP-hard in [1]—can be optimally solved by a dynamic program running in pseudopolynomial time and also that the SPT heuristic is a 9/7-approximation algorithm. Recently, the problem addressed in [7] has been generalized by considering a workload-dependent maintenance period. For the latter problem, different solution algorithms are proposed in [6]. Moreover, in [2], the authors study a similar version of the same problem in which it is not possible to start or complete a job during the maintenance period (while it is allowed to continue its processing).

In the last decades many authors, in several fields, have addressed robust optimization problems in order to take care of incomplete or erroneous data through a *proactive* approach. In particular, in robust discrete optimization, one usually assumes that, due to the variability of some parameters, a set of possible *scenarios* is defined. The robust approach consists in minimizing the worst case performance of a solution over all scenarios. To this purpose, a commonly used approach is that of minimizing the *maximum regret*, i.e., the maximum possible deviation of a given solution from the optimal one. In the context of single machine scheduling problems, robustness is an important issue addressed in several papers. One of the first papers on this topic is [3], where the authors consider two alternative measures of schedule robustness. These measures focus on a given schedule worst-case absolute or relative deviation from the

optimum over all scenarios. The authors study the problem of minimizing total completion time on a single machine, when processing times may vary in given intervals, and establish several properties of robust schedules.

Differently from most of the robust scheduling problems addressed in the literature, here, we consider jobs with deterministic processing times, whereas the uncertainty is only relative to the duration of the maintenance activity. On the other hand, the objective function depends only on the completion times of the jobs.

2 Problem statement

We assume the n jobs have processing times p_j, $j = 1, \ldots, n$. The maintenance activity (hereafter MA) must be performed within a time interval $[r, d]$ with given release time r and deadline d. The MA duration may vary in a discrete set of values and its realization value is unknown to the decision maker until the scheduling has been decided. In this context, the problem consists in finding a schedule of the n jobs and the MA, in order to minimize the total completion time of the jobs while processing the MA within the time interval $[r, d]$.

We assume that the duration of the MA depends on the particular scenario s that will be observed after the scheduling has been decided. We also assume that s belongs to a set $\mathcal{S}$ of scenarios, each corresponding to a possible realization $p = P(s)$ of the duration of the maintenance activity.

A solution of our problem is a sequence π of jobs, only. Its value depends on π and on the realization $P(s)$ and it is equal to the total completion time of the jobs in the so called *realization schedule* $\sigma(\pi, s)$ which is built as follows. Given a scenario s and $P(s)$, the MA is processed at the latest available time so that: (1) it does not violate the given deadline d and (2) it does not introduce any unnecessary idle time in the schedule after its release time. The jobs of D are processed according to the sequence π. Figure 1 illustrates these concepts through an example. More rigorously, a realization schedule can be defined as follows.

Definition 1 *Given a scenario $s \in \mathcal{S}$, the realization $P(s)$, and a sequence $\pi = \langle \pi_1, \pi_2, \ldots \pi_n \rangle$, let $k(s)$ be the index of the* critical *job in π such that*

$$d - p_{\pi_{k+1}} < P(s) + \sum_{i=1}^{k} p_{\pi_i} \leq d - \max\left\{0, r - \sum_{i=1}^{k} p_{\pi_i}\right\}$$

then the realization schedule $\sigma(\pi, s)$ *is obtained by scheduling the MA at the earliest possible time between jobs $\pi_{k(s)}$ and $\pi_{k(s)+1}$. Hence, the resulting sequence of jobs in $\sigma(\pi, s)$ is $\langle \pi_1, \ldots \pi_{k(s)}, MA, \pi_{k(s)+1}, \ldots \pi_n \rangle$.*

The value $\bar{C}(\pi, s)$ of a realization schedule $\sigma(\pi, s)$ is the total completion time of the jobs and it is given by

$$\bar{C}(\pi, s) = \sum_{\ell=1}^{n} C_\ell = \sum_{\ell=1}^{n} \left(\sum_{i=1}^{\ell} p_{\pi_i} \right) + (n - k(s)) \left(P(s) + \max\left\{0, r - \sum_{i=1}^{k(s)} p_{\pi_i}\right\} \right).$$

Observe that, since the realization of P is not known, the critical job and therefore the realization schedule $\sigma(\pi, s)$ are not known as well. Hence, a solution would merely consist in a jobs sequence π such that—whatever MA's duration is—the resulting schedule is satisfactory

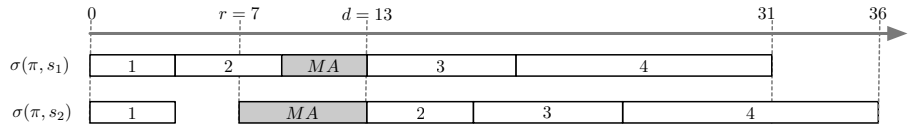


Figure 1: An example of realization schedules $\pi =< 1, 2, 3, 4 >$, where the processing times are 4, 5, 7, and 11, respectively and $r = 7$ $d = 13$. There are two scenarios and the two corresponding values for the MA duration are $P(s_1) = 4$ and $P(s_2) = 6$.

in terms of jobs' total completion time. In other words, π must be robust with respect to the variations of the maintenance activity duration.

Usually, in an optimization problem the *regret* of a solution x is the difference between the value obtained by x and that of an optimal solution. If some parameters may take several different values (and therefore there are different optimal solutions depending on the parameters' values) it makes sense to search for a solution minimizing the worst-case regret (i.e., the maximum value of the regret over all possible scenarios). Such a solution is called the minimax *worst-case absolute* regret solution. The scenario under which the maximum regret is realized is denoted as *worst-case absolute deviation* scenario. Here, we address the worst-case absolute deviation problem which can be formulated as follows.

> *Robust Single Machine Scheduling with Flexible Maintenance Activity* (RSMP):
> *Given*: jobs processing times p_j, $j = 1, \ldots, n$, a set $\mathcal{S}$ of discrete scenarios;
> *Find*: a jobs sequence π *such that*: the resulting schedule minimizes the worst-case absolute regret $\rho(\pi) = \max_{s \in \mathcal{S}} \bar{C}(\pi, s) - \bar{C}^*(s)$, where $\bar{C}^*(s)$ is the minimum total completion time value in scenario s.

In the literature many authors, besides the minimax worst-case absolute regret, consider also other measures of robustness (see, for instance, [3] in which the authors introduce a sort of "normalized" figure, denoted as worst-case *relative* regret).

3 Main results

Hereafter, we concisely list the main contributions of this work.

• In [1], it is shown that the deterministic (i.e., single-scenario) version of RSMP is NP-hard if $r > 0$, that is, the following result holds:

Remark 2 *RSMP is NP-hard for any fixed number $|\mathcal{S}| \geq 1$ of scenarios.*

• However even our robust version of the problem becomes easy if there is no release date for the maintenance activity:

Theorem 3 *If the release date of the MA is $r = 0$, then RSMP is polynomially solvable for any number $|\mathcal{S}| \geq 1$ of scenarios and an optimal solution is given by sequencing the jobs in SPT order.*

• When a number of scenarios are available, it comes natural to use, as a robust solution, a sequence which is optimal in at least one scenario. This could be pursued by iteratively applying the pseudopolynomial dynamic program proposed by [7]. Unfortunately, this approach, in general, does not yield a solution minimizing the maximum regret:

Theorem 4 *The optimal (robust) solution of RSMP may not correspond to the optimal solution of any of the scenarios in $\mathcal{S}$.*

• In [3] the authors consider the problem in which all processing times may assume values in given intervals (and no maintenance activity exists): They show that for any sequence, when minimizing total completion time, a worst-case absolute scenario occurs when the jobs' processing times are all equal either to their minimum or to their maximum values. However, it is possible to show that for RSMP this property does not hold:

Theorem 5 *There exist an instance I of RSMP and sequence π of the D jobs such that the maximum regret scenario does not correspond to an extreme value of the MA duration P.*

• As shown in [7], if $r > 0$, SPT is a 9/7-approximation algorithm for the deterministic counterpart of RSMP. This approximation ratio holds even when considering the robust version of the same problem, for some robustness measures such as the minimax regret when considering the worst-case *relative* deviation as a measure of regret. Unfortunately, this positive result does not extend to the case of minimax absolute regret objective:

Theorem 6 *The approximation ratio of SPT for RSMP is unbounded.*

• In addition to the above negative results, we performed a number of tests to assess the quality of two heuristic algorithms, namely the SPT algorithm and an improved version of the latter one in which a local search is performed. We compared the performance of these heuristics against two mixed integer linear programs (solved by CPLEX) and show that, despite Theorem 6, SPT may return acceptable regret values in a substantial fraction of test instances.

References

[1] Adiri I., J. Bruno, E. Frostig, A. H. G. Rinnooy Kan (1989). Single machine flow-time scheduling with a single breakdown, Acta Informatica, 26(7), 679–696.

[2] Chen Y. , A. Zhang, Z. Tan (2013). Complexity and approximation of single machine scheduling with an operator non-availability period to minimize total completion time, Information Sciences 251, 150–163.

[3] Daniels R.L., P. Kouvelis (1995). Robust Scheduling to Hedge Against Processing Time Uncertainty in Single-stage Production, Management Science, 41(2), 363–376.

[4] Detti P., G. Nicosia, A. Pacifici, G. Zabalo Manrique de Lara (2016). Robust single machine scheduling with external-party jobs, IFAC-PapersOnLine, 49 (12), pp. 1731–1736 , proceedings of IFAC MIM 2016, Troyes, France.

[5] Ma Y., C. Chu, C. Zuo (2010). A survey of scheduling with deterministic machine availability constraints, Computers & Industrial Engineering, 58(2), 199–211.

[6] Xu D., L. Wan, A. Liu, D. Yang (2015). Single machine total completion time scheduling problem with workload-dependent maintenance duration, Omega 52, 101–106.

[7] Yang S., Y. Maa, D.-l. Xu, J.-b. Yang (2011). Minimizing total completion time on a single machine with a flexible maintenance activity. Computers & Operations Research, 38, 755–770.

2-proper connection number of graphs

Trung Duy Doan[1,2,*], Christoph Brause[1], and Ingo Schiermeyer[1]

[1]Institute of Discrete Mathematics and Algebra, TU Bergakademie Freiberg, Freiberg, Germany, *brause@math.tu-freiberg.de, ingo.schiermeyer@tu-freiberg.de*
[2]School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, Hanoi, Vietnam, *trungdoanduy@gmail.com*

A path in an edge-coloured graph is called *properly coloured* if every two consecutive edges receive distinct colours. An edge-coloured graph G is called *k-properly connected* if every two vertices are connected by at least k internally pairwise vertex-disjoint properly coloured paths. The *k-proper connection number* of a connected graph G, denoted by $\mathrm{pc}_k(G)$, is the smallest number of colours that are needed in order to make G k-properly connected. In this paper, we prove a new upper bound for $\mathrm{pc}_k(G)$ in k-connected graphs and determine $\mathrm{pc}_2(G) = 2$ of several classes of 2-connected graphs.

1 Introduction

We use [4] for terminology and notation not defined here and consider simple, finite and undirected graphs only.

The concept of proper connection of graphs is an extension of proper colouring and is motivated by rainbow connection of graphs. Andrews et at. [1] and, independently, Borozan et at. [2] introduced the concept of proper connection of graphs. A path in an edge-coloured graph is called *properly coloured* if every two consecutive edges receive distinct colours. An edge-coloured graph G is called *k-properly connected* if every two vertices are connected by at least k internally pairwise vertex-disjoint properly coloured paths (for simplicity, we say k *disjoint properly coloured paths*). The *k-proper connection number* of a connected graph G, denoted by $\mathrm{pc}_k(G)$, is the smallest number of colours that are needed in order to make G k-properly connected. Clearly, if G is k-properly connected, then it is also k-connected. Conversely, if G is k-connected and properly edge-colouring, then G is k-properly connected, too. Hence, $\mathrm{pc}_k(G)$ can be easily bounded from above by the chromatic index $\chi'(G)$, which is at most $\Delta(G)$ or $\Delta(G) + 1$ by Vizing's Theorem. Thus, $\mathrm{pc}_k(G) \leq \Delta(G) + 1$ if G is k-connected. For $k = 1$, we write $\mathrm{pc}(G)$ instead of $\mathrm{pc}_1(G)$ and call $\mathrm{pc}(G)$ *proper connection number* of G. We note that, as proved by Borozan et al., $\mathrm{pc}(G) = 1$ if and only if $G \cong K_n$ [2], implying that $\mathrm{pc}_k(G) \geq 2$ for any $k \geq 2$ and any non-complete k-connected graph G. Furthermore, it is not hard to see that if $k \geq 2$ and G is a non-complete k-connected graph, then $\mathrm{pc}_k(G) \geq 2$. Therefore, the following result follows immediately.

*Financial support by the Free State of Saxony (Landesgraduiertenstipendium) in Germany and the National Foundation for Science and Technology Development (NAFOSTED) of Vietnam with project code 101.99-2016.20 is thankfully acknowledged.

Corollary 1. *If $k \geq 2$ and G is a k-connected graph, then $2 \leq \mathrm{pc}_k(G) \leq \Delta(G) + 1$.*

We proceed with four further fundameltal results for the k-proper connection number of a graph. Let us start with results for $k = 1$. If G is two connected, then we can bound the proper connection number $\mathrm{pc}(G)$ from above by 3.

Theorem 2 (Borozan et at. [2])**.** *If G is a 2-connected graph, then $\mathrm{pc}(G) \leq 3$.*

Furthermore, a useful results concerning spanning subgraphs is given by Andrews et al.

Lemma 3 (Andrews et al. [1])**.** *If G is a non-trivial connected graph and H is a connected spanning subgraph of G, then $\mathrm{pc}(G) \leq \mathrm{pc}(H)$. In particular, $\mathrm{pc}(G) \leq \mathrm{pc}(T)$ for every spanning tree T of G.*

We generalise Lemma 3 as follows:

Lemma 4. *Let $k \geq 2$ and G be a k-connected graph. If H is a k-connected spanning subgraph of G, then $\mathrm{pc}_k(G) \leq \mathrm{pc}_k(H)$.*

The last restult, in our list of fundamental ones, can be easily seen by Corollary 1, and by the observations that any cycle C_n has edge-chromatic number $2 + (n \mod 2)$ and an odd cycle cannot be coloured properly connected by two colours.

Fact 5. *If $C_n : v_1v_2...v_n$ is a cycle of order n, then $\mathrm{pc}_2(G) = 2 + (n \mod 2)$.*

The main aim of this this paper is to study the 2-proper connection number $\mathrm{pc}_2(G)$ in 2-connected graphs G. By Corollary 1, $\Delta(G) + 1$ is an upper boud for $\mathrm{pc}_2(G)$. We can characterise 2-connected graphs having 2-proper connection number $\Delta(G) + 1$. Furthermore, our next result shows further that, different from the proper connection number of a graph, there does not exist a constant C such that $\mathrm{pc}_2(G) \leq C$ for all 2-connected graphs G.

Theorem 6. *If G is 2-connected graph and different from an odd cycle, then $\mathrm{pc}_2(G) \leq \Delta(G)$. We achieve the 2-proper connection number $\Delta(G) + 1$ if and only if G is an odd cycle.*

Sketch of the Proof. For our proof of Theorem 6, we use the following two results.

Theorem 7 (Dirac [3])**.** *If G is a minimally spanning 2-connected graph, then G is chordless.*

Theorem 8 (Machado et al. [5])**.** *If G is a chordless graph of maximum degree at least 3, then G is $\Delta(G)$-edge-colourable.*

Trivially, if G is a cycle, then, by Fact 5, $\mathrm{pc}_2(G) = 2 + (n \mod 2) = \Delta(G) + (n \mod 2)$ and we obtain the result.

If G is dictinct from a cycle and it is 2-connected graph, then $\Delta(G) \geq 3$. Let H be a spanning subgraph of G which is minimally 2-connected. Now, $\Delta(H) \leq \Delta(G)$. By Theorem 7 and Theorem 8, $\mathrm{pc}_2(H) \leq \Delta(H)$. But now, by Lemma 4, $\mathrm{pc}_2(G) \leq \mathrm{pc}_2(H)$. Thus,

$$\mathrm{pc}_2(G) \leq \chi'(H) \leq \Delta(H) \leq \Delta(G),$$

and we obtain the result. □

Proposition 9. *For any $k \geq 2$, there exist infinitly many graphs G with $\Delta(G) = k$ and $\mathrm{pc}(G) = k$.*

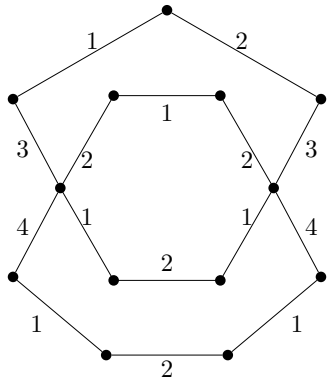


Figure 1: Graph with $\mathrm{pc}_2(G) = \Delta(G) = 4$

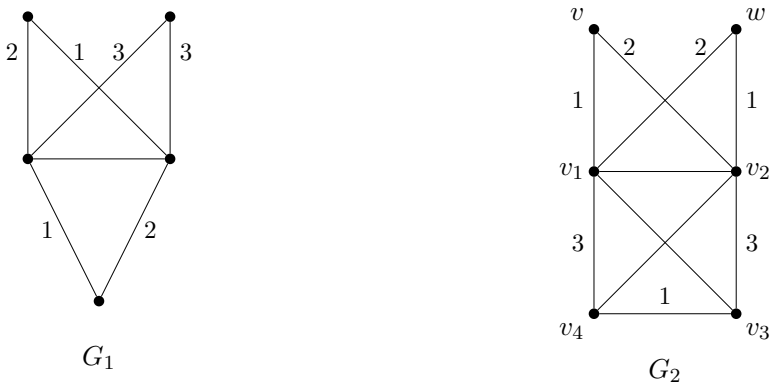


Figure 2: $\mathrm{pc}_2(G_i) = 3$ for $i \in \{1, 2\}$

The graph G depicted in Figure 1 has 2-proper connection number $\mathrm{pc}_2(G) = 4$ and is one of example graphs for Proposition 9. Furthermore, let G_1 and G_2 be the two graphs depicted in Figure 2. One can easily see that $\mathrm{pc}_2(G) = 3$ for $G \in \{G_1, G_2\}$.

By Proposition 9, there exist always 2-connected graphs having equal 2-proper connection number $\mathrm{pc}_2(G)$ and maximum degree $\Delta(G)$, but the difference $\Delta(G) - \mathrm{pc}_2(G)$ can be arbitrarily large, as our next result about dense graphs will show.

Theorem 10. *Let G be 2-connected graph of order n and $\omega(G)$ be the cardinality of a largest clique in G. If*

a) $\omega(G) = n$ and $n \geq 4$,

b) $\omega(G) = n - 1$, or

c) $\omega(G) = n - 2$ and $G \notin \{G_1, G_2\}$,

then $\mathrm{pc}_2(G) = 2$.

If we consider the relation between the number of edges $|E(G)|$ and the order n of G, then we obtain the following result.

Theorem 11. *Let G be a connected graph of order $n \geq 4$. If $|E(G)| \geq \binom{n-1}{2} + 2$, then* $\mathrm{pc}_2(G) = 2$.

The *independence number* of a graph G, denoted by $\alpha(G)$, is the cardinality of a largest independent set in G.

Theorem 12. *Let G be a graph of order $n \geq 4$ having at least $\alpha(G)$ vertices of degree $n-1$. Then* $\mathrm{pc}_2(G) = 2$

Next, we study the 2-proper connection number of the Cartesian product of two connected graphs. Let $G_1 \square G_2$ be the *Cartesian product* of G_1 and G_2. Futhermore, a *traceable* graph is a graph which admits a Hamiltonian path.

Theorem 13. *If G_1, G_2 are two traceable graphs, then* $\mathrm{pc}_2(G_1 \square G_2) = 2$.

If at least one of two graphs G_1, G_2 is not a traceable graph, then Theorem 13 is not necessarily true. We obtain the following result, implying that there are infinitly many pairs of connected graphs G_1, G_2 with $\mathrm{pc}_2(G_1 \square G_2) > 2$.

Proposition 14.

a) If $K_{1,m}$ is a star and P_n is a path with $m, n \in \mathbb{N}, m \geqslant 3, n \geqslant 2$, then $\mathrm{pc}_2(K_{1,m} \square P_n) = m$.

b) If $K_{1,m}, K_{1,n}$ are two stars with $m, n \in \mathbb{N}$ and $m, n \geq 3$, then $\mathrm{pc}_2(K_{1,m} \square K_{1,n}) = max\{m, n\}$.

Finally, we study the relation between 2-proper connection number and proper connection number by the following Theorem.

Theorem 15. *Let G_1, G_2 be connected graphs with $n(G_1) \geq 4$ and $n(G_2) \geq 2$. If $\delta(G_1) \geq 2$, then* $\mathrm{pc}_2(G_1 \square G_2) \leq \mathrm{pc}(G_1) + 1$.

References

[1] E. Andrews, C. Lumduanhom, E. Laforge, and P. Zhang, *On Proper-Path colourings in Graphs*, Journal of Combinatorial Mathematics and Combinatorial Computing, 97, 189-207.

[2] V. Borozan, S. Fujita, A. Gerek, C. Magnant, Y. Manoussakis, L. Montero, and Z. Tuza, *Proper connection of graphs*, Discrete Math. 312(17) (2012), 2550–2560.

[3] G.A. Dirac, *Minimally 2-connected graphs*, J. Reine Angew. Math. 228 (1967), 204- 216.

[4] F. Harary, *Graph Theory*, Addison-Wesley, 1969.

[5] R.C.S. Machado, C.M.H. de Firueiredo, and N. Trotignon, *Edge-colouring and total-colouring chordless graphs*, Discrete Math. 313 (2013), 1547-1552.

[6] V. G. Vizing, *On an estimate of the chromatic class of a p-graph*, Diskret. Anal. (3) (1964) 25-30.

An Efficient Ear Decomposition Algorithm

Debarshi Dutta[1], Kishore Kothapalli[1], G. Ramakrishna [2],
Sai Charan Regunta [2], and Sai Harsh Tondomker [2]

[1]International Institute of Information Technology, Hyderabad. India.
[2]Indian Institute of Information Technology Chittoor, Sri City, India.

An ear decomposition of a graph G is a partition of the edge set of G into a sequence of edge-disjoint paths, such that only the end vertices of each path appear in earlier paths. For a graph on n vertices and m edges, the state-of-art algorithm for obtaining an ear decomposition by Schmidt takes $O(m+n)$ time. We design and implement a new algorithm to obtain an ear decomposition for a biconnected graph, whose running time $O(m+n)$. In practice, however, our experiments reveal that, the proposed algorithm runs at least 2 times faster than Schmidt's algorithm. The speedup increases as the graph gets denser.

1 Introduction

Obtaining an ear decomposition of a graph is an important problem in the context of graph algorithms. An ear decomposition of a graph is used in several other graph algorithms such as testing connectivity, s-t-numbering and planarity-testing [6]. Also, an ear decomposition has been used as a paradigm to obtain parallel algorithms for various problems. Whitney has introduced the notion of ear decomposition to characterize biconnected graphs [7]. An *ear* of a graph G is a maximal path whose internal vertices have degree 2. An *ear decomposition* of a graph $G = (V, E)$ is a partition of E into a sequence $(P_0, P_1, \cdots, P_k)$, such that (i) P_0 is a cycle, (ii) for each $i \geq 1$, P_i is an ear of $P_0 \cup \ldots \cup P_i$. An ear decomposition $(P_0, P_1, \ldots, P_k)$ of G is an *open ear decomposition* if the end points of all the ears P_i, $i \geq 1$, are distinct.

Past Work. The concept of ear decomposition is very well studied both structurally and algorithmically. An ear decomposition is used to characterize biconnected graphs [7]. Further, the notion of nested ear decomposition and nice ear decompositions are developed to characterize series-parallel graphs and polygonal 2-trees, respectively [2, 5]. For a graph on n vertices and m edges, algorithms to obtain an ear decomposition in $O(m+n)$ time are known for a long time. Lovász has designed an algorithm for the first time to obtain an ear decomposition in parallel framework [4]. The state-of-art serial algorithm to construct an ear decomposition is proposed by Schmidt [6]. His algorithm is based on depth first search spanning tree and is simpler to visualize. The algorithm of Schmidt also runs in time $4m + O(n)$.

Motivation and Our Contribution. To the best of our knowledge, there are no studies on computing ear decomposition from a practical perspective in serial computing. The state of art algorithm by Schmidt is simple and elegant [6]. However, this algorithm has not been explored in practice. The main objective of this work is to obtain an ear decomposition algorithm that works well both in theory and practice. In this paper, we present an $O(m+n)$ algorithm that offers an improvement to the algorithm of Schmidt in the practical setting. In particular, we

show that, a large number of edges of a graph are *redundant* in the process of obtaining an ear decomposition. Removing such redundant edges in a prior pre-processing step can often result in practical improvement. Our characterization of redundant edges (trivial ear) is based on a similar notion that is identified in the context of biconnectivity [1]. In practice, our algorithm runs at least 2 times faster than Schmidt's algorithm on graphs having $\Omega(n \log n)$ edges.

2 Algorithm for Ear Decomposition

We use standard graph terminology from [7]. For a graph G, let $n = |V(G)|$ and $m = |E(G)|$ denote the number of vertices and edges in G, respectively. A graph is *biconnected* (*2-vertex connected*) if it does not contain a cut-vertex. An edge e in a biconnected graph is *non-essential* if the graph remains biconnected after the removal of e. For $i \geq 2$, an ear P_i in an ear decomposition is a *trivial ear*, if the number of edges in P_i is one. In the rest of the paper, G denotes an unweighted and undirected biconnected graph.

The main idea in Schmidt's algorithm for computing an ear decomposition is to obtain a depth first search tree T and process all the non-tree edges (edges in $G - T$) to construct $m - n + 1$ ears. The time required to perform these two steps is $4m + O(n)$, where at most $2m + O(n)$ is required in the individual steps.

Algorithm 1 shows an overview of our approach to compute an ear decomposition. At a higher-level, the main idea in our approach is to filter many non-essential edges and compute an ear decomposition on the rest of the graph.

Algorithm 1: An algorithm to find an ear decomposition of a biconnected graph G

1 Construct a breadth first search (BFS) spanning tree T of G ;
2 Construct a spanning forest F from G', where $G' = G - T$;
3 Find an ear-decomposition $\mathcal{P}$ of $T \cup F$ using Schmidt's algorithm ;
4 return the sequence of ears in $\mathcal{P}$ and the edges in G'' as trivial ears, where $G'' = G' - F$;

The proof of correctness of Algorithm 1 is shown in Theorem 3, using Lemma 1 and Lemma 2.

Lemma 1 ([7])**.** *A graph G is biconnected (2-vertex connected) if and only if G has an open ear decomposition.*

Lemma 2 ([1])**.** *Let T be a* BFS *spanning tree of G and F be a spanning forest in $G - T$. Then, the edges of each connected component of $G - T$ are in one biconnected component. The number of biconnected components in G and $T \cup F$ is same.*

Theorem 3. *For a biconnected graph G, let T be a* BFS*-spanning tree of G and F be a spanning forest in $G - T$. Then there is an ear decomposition $\mathcal{P}$ of G in which every edge in $G - (T \cup F)$ is a trivial ear.*

Proof. From Lemma 2, $T \cup F$ is biconnected. Then, by Lemma 1, there is an open ear decomposition $\mathcal{P}'$ of $T \cup F$. As a result, $\mathcal{P}'$ with each edge in $G - (T \cup F)$ being a trivial ear, becomes an open ear decomposition of G with the mentioned property. □

In Algorithm 1, Line 1 and Line 3 take $2m + O(n)$ time and $O(n)$ time, respectively. Constructing a spanning forest F from $G - T$ efficiently in Line 2 is a more involved task. If we use breadth first or depth first traversals, then Line 2 consumes $2m + O(n)$ time, the total running time is $4m + O(n)$. This matches with Schmidt's algorithm, and hence this is not a promising idea. If we use disjoint-set-forest datastructure with union-by-rank and path-compression, the Line 2 consumes $m\alpha(n) + O(n)$ time, the total run time is $2m + \alpha(n)m + O(n)$. A linked-list

based disjoint-set data-structure requires $m + O(n \log n)$ time to compute the task in Line 2. However, as the data-locality is very poor in linked-lists, our experiments reveal that this idea does not beat Schmidt's algorithm in practice.

To make our idea work, we introduce a randomized algorithm Algorithm 2, shown below for the implementation of Line 2 in Algorithm 1.

Algorithm 2: Algorithm to implement Line 2 in Algorithm 1

1 $X = \{v \in V(G) \mid degree(v) \text{ in } T = degree(v) \text{ in } G\}$, $F = \emptyset$;
2 **for** *each vertex* $u \in V(G) - X$ **do**
3 append each edge incident at u to F with probability $\frac{\log n}{n}$;

The key idea in the randomized algorithm is to choose a sparse spanning connected subgraph of $G - X$, instead of a spanning forest of G. For each vertex $u \in V(G)$, each edge incident at u is sampled with probability $\frac{\log n}{n}$ and all the sampled edges in the random process form a spanning connected subgraph of $G - X$. This insight is inferred from our experiments and support the correctness of Algorithm 2. Let the number of vertices and edges in the input graph H be n vertices and $\Omega(n \log n)$ edges, respectively. If H' is a spanning subgraph of H that is constructed by sampling each edge of H with probability $\frac{\log n}{n}$, then we claim that H' is connected with high probability. The proof of this claim is open and Lemma 4 would be helpful to derive the proof. The expected number of edges in F is in $O(n \log n)$. If we assume that each edge can be sampled in constant time, then Algorithm 2 runs in $m + O(n \log n)$ time, and hence the run time of Algorithm 1 is $3m + O(n \log n)$. This run time is further reduced to $2m + O(n \log n)$ time in next section. Since m is $\Omega(n \log n)$, our algorithm runs in $O(m + n)$ time. However, our algorithm performs well in practice as m increases beyond $\Omega(n \log n)$.

Lemma 4. *[3] A random graph $G(n, p)$ on n vertices and edge probability p is connected with very high probability, if $p = \frac{\ln n}{n}$.*

3 Implementation Details and Experiments

In this section, we describe the lower level details of our algorithm. Without loss of generality, let us assume that the vertices in the input graph are numbered from 1 to n. We use the *compressed sparse row* (CSR) representation to store the input graph and as well as the intermediate graphs. A CSR of a graph G consists of two arrays, namely a vertex array $V[\,]$ and an edge array $E[\,]$. For each i, $1 \leq i \leq n$, $V[i]$ denotes an index in $E[\,]$ such that all the neighbors of i are stored in $E[\,]$ at locations $V[i], \ldots, V[i+1] - 1$.

We recall the construction of spanning forest F in Algorithm 2. Each edge incident at a vertex u is sampled with probability $\frac{\log n}{n}$. Then, the expected number of edges in F that are incident on u for a dense graph is at most $\log n$. In practice, this step consumes more time as every edge needs to be processed. One way to perform this step is as follows. For sampling k edges that are incident on a vertex i, we first choose k random indices whose range is $V[i], \ldots, V[i+1]-1$, and move these edges from G to F. If the number of edges incident at a vertex are less than k, then we move all of them from G to F. The run-time of Algorithm 2 as per this idea is $O(n \log n)$, and thus the run-time of our main algorithm is $2m + O(n \log n)$. The edges that are not sampled from CSR of G will be remained as trivial ears.

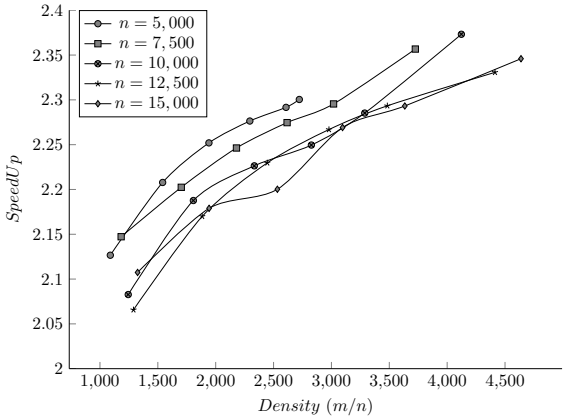


Figure 1: Density Vs SpeedUp

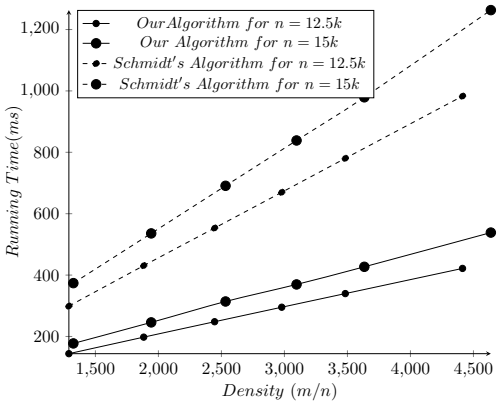


Figure 2: Schmidt's Vs Our Algorithm

Experimental Results

We implement our algorithms in C using `gcc` 4.8.5 compiler. We use Intel(R) Xeon(R) CPU E5-2650 v3 processor, which is based on x86_64 architecture. The frequency of the processor that we use in our experiments is 2.30GHz. The details on the number of vertices (n) and density (m/n) of the graphs considered in our experiments, which are constructed using GT-generator, are given in the left-side figure shown below. Our experiments reveal that the speedup is proportionate to the density of a graph. In other words, as the density of a graph increases, the speedup of our algorithm increases further and this is shown in Figure 1. The run-time of our algorithm against Schmidt's algorithm for $n = 15K$ and $n = 12.5K$, and for various densities are shown in Figure 2. In our experiments, the runtime plot against the density for $n = 5K$, $n = 7.5K$ and $n = 10K$ follow the similar trend shown in Figure 2. We conclude that, our algorithm runs at least 2 times faster than Schmidt's algorithm in practice.

Remark. Our algorithm can be used to obtain an approximated minimal biconnected graph. Finding the trade off between the quality of the solution and runtime by our algorithm against the state-of-art approximation algorithms to find a minimal biconnected graph is an interesting study.

References

[1] G. Cong and D. A. Bader. An experimental study of parallel biconnected components algorithms on symmetric multiprocessors (smps). *Inter. Par. and Dist. Proc. Symp.*, 2005.

[2] D. Eppstein. Parallel recognition of series-parallel graphs. *Inf. Comput.*, 98(1):41–55, 1992.

[3] P. Erdős and A Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, pages 17–61, 1960.

[4] L. Lovász. Computing ears and branchings in parallel. *Found. of Comp. Sci, 464–467*, 1985.

[5] N.S. Narayanaswamy and G. Ramakrishna. On minimum average stretch spanning trees in polygonal 2-trees. *Theor. Comput. Sci.*, 575(C):56–70, 2015.

[6] J. M. Schmidt. A simple test on 2-vertex- and 2-edge-connectivity. *Info. Proc. Lett.*, 113(7):241–244, 2013.

[7] Douglas B. West. *Introduction to graph theory - second edition.* Prentice Hall, 2001.

Parity Polytopes and Binarization

Dominik Ermel[1] and Matthias Walter[2]

[1]Otto-von-Guericke-Universitt Magdeburg, dominik.ermel@st.ovgu.de
[2]RWTH Aachen University, walter@or.rwth-aachen.de

1 Introduction

The term *binarization* refers to techniques to reformulate mixed-integer linear programs by replacing general integer (bounded) variables by sets of binary variables. Roy compared several approaches with the goal to separate strong cutting planes in the reformulation and project it back [5]. Recently, Bonami and Margot continued this line of work using simple split disjunctions of rank 1 and 2 to generate cutting planes [2]. The strongest bounds were obtained by a binarization in which an integer variable $z \in \{0, 1, \dots, n\}$ is replaced by the sum of n binary variables $x_1, \dots, x_n$, and symmetry-breaking constraints

$$1 \geq x_1 \geq \ldots \geq x_n \geq 0 \tag{1}$$

are added. We denote by X^n_{ord} the set of ordered binary vectors $x \in \{0, 1\}^n$ satisfying (1) and by P^n_{ord} their convex hull. Note that Inequalities (1) already describe P^n_{ord}. We call the vectors in X^n_{ord} *ordered binary vectors*.

We consider the strong binarization in combination with parity constraints on the affected integer variables. For this we define the *ordered even parity polytope* for a vector $r \in \mathbb{N}^k$ as

$$P^r_{\text{even}} := \text{conv}\{(x^{(1)}, \dots, x^{(k)}) \in X^{r_1}_{\text{ord}} \times \ldots \times X^{r_k}_{\text{ord}} \mid \sum_{i=1}^{k} \sum_{j=1}^{r_i} x^{(i)}_j \text{ even}\}.$$

The integer points in P^r_{even} are precisely those binary vectors of length $n := r_1 + \ldots + r_k$ whose entry-wise sum is even and which are ordered within each of the groups defined by r. In the special case of $r = \mathbb{1}_n$ (the all-ones vector), P^r_{even} is the even parity polytope, which has the following outer description [3] (with $[n] := \{1, 2, \dots, n\}$):

$$P^{\mathbb{1}_n}_{\text{even}} = \{x \in [0, 1]^n \mid \sum_{i \in [n] \setminus F} x_i + \sum_{i \in F} (1 - x_i) \geq 1 \text{ for all } F \subseteq [n] \text{ with } |F| \text{ odd}\} \tag{2}$$

Outline. We start by providing a complete outer description for P^r_{even} in the original space and establishing separation results. Section 3 is dedicated to an application in which binarization is applied to an integer programming model for the graphic traveling salesman problem. During computations for this model we observed that the corresponding parity constraints did not improve the dual relaxation value. In Section 4 we provide a theoretical explanation of this effect.

2 Outer Description and Separation Algorithms

We begin with the simple observation that for ordered binary vectors, parity can be measured using a linear function. For $n \in \mathbb{N}$ we define $f : P^n_{\text{ord}} \to \mathbb{R}$ via $f(x) := \sum_{i=1}^{n} (-1)^{i-1} x_i$. By bounding $x_i - x_{i+1} \geq 0$ for all even (resp. odd) indices, we obtain that f maps P^n_{ord} into $[0, 1]$. Thus, for $x \in X^n_{\text{ord}}$, the value $f(x)$ is binary, and it is easy to check that $f(x) = 1$ holds if and only if x has an odd number of 1's. Note that we will use f for different values of n, and assume that it is clear from the context. We can now state our main results.

Theorem 1. *Let $r \in \mathbb{N}^k$. The ordered even parity polytope P^r_{even} is equal to the set of points $(x^{(1)}, \ldots, x^{(k)}) \in P^{r_1}_{ord} \times \ldots \times P^{r_k}_{ord}$ that satisfy the inequalities*

$$\sum_{i \in [k] \setminus F} f(x^{(i)}) + \sum_{i \in F} (1 - f(x^{(i)})) \geq 1 \tag{3}$$

for all $F \subseteq [k]$ with $|F|$ odd.

Theorem 2. *Let $r \in \mathbb{N}^k$ and let $\hat{x} = (\hat{x}^{(1)}, \ldots, \hat{x}^{(k)}) \in P^{r_1}_{ord} \times \ldots \times P^{r_k}_{ord}$. We can decide in linear time if $\hat{x} \in P^r_{even}$ holds, and if this is not the case, obtain an odd set $F \subseteq [k]$ whose associated Inequality (3) is violated by $\hat{x}$.*

Similar results hold for ordered *odd* polytopes, defined for $r \in \mathbb{N}^k$:

$$P^r_{\text{odd}} := \text{conv}\{(x^{(1)}, \ldots, x^{(k)}) \in X^{r_1}_{\text{ord}} \times \ldots \times X^{r_k}_{\text{ord}} \mid \sum_{i=1}^{k} \sum_{j=1}^{r_i} x^{(i)}_j \text{ odd}\}.$$

Corollary 3. *The ordered odd parity polytope for $r \in \mathbb{N}^k$ is the set of $(x^{(1)}, \ldots, x^{(k)}) \in P^{r_1}_{ord} \times \ldots \times P^{r_k}_{ord}$ that satisfy Inequalities (3) for all $F \subseteq [k]$ with $|F|$ even.*

Corollary 4. *Let $r \in \mathbb{N}^k$ and let $\hat{x} = (\hat{x}^{(1)}, \ldots, \hat{x}^{(k)}) \in P^{r_1}_{ord} \times \ldots \times P^{r_k}_{ord}$. We can decide in linear time if $\hat{x} \in P^r_{odd}$ holds, and if this is not the case, obtain an even set $F \subseteq [k]$ whose associated Inequality (3) is violated by $\hat{x}$.*

3 Strengthened Blossom Inequalities for the Graphic TSP

As an application, we consider the *graphic traveling salesman problem (GTSP)*, defined as follows. The input consists of an undirected graph $G = (V, E)$, and the goal is to find a minimum-length closed walk in G that visits every node at least once. Recently, Sebő and Vygen developed a 7/5-approximation algorithm for this problem [6]. When modeling it as an integer program (IP) one has different options. The first is based on the observation that the problem is equivalent to the traveling salesman problem on the complete graph with $|V|$ nodes and edge weights $c \in \mathbb{R}^E_+$ where $c_{u,v}$ is equal to the (combinatorial) distance from u to v in G. This model has $\binom{|V|}{2}$ binary variables which is much greater than $|E|$ if G is sparse.

In order to model the problem with fewer variables we can start with the IP

$$\min \quad \sum_{e \in E} z_e \tag{4}$$

$$\text{s.t.} \quad z(\delta(S)) \geq 2 \qquad \text{for all } \emptyset \neq S \subsetneq V \tag{5}$$

$$z_e \geq 0 \qquad \text{for all } e \in E \tag{6}$$

$$z_e \in \mathbb{Z} \qquad \text{for all } e \in E, \tag{7}$$

where by $\delta(\cdot)$ we denote the cut-sets. The solutions to this problem correspond to the 2-edge-connected spanning subgraphs of G. Since not every such subgraph is a closed walk, this IP does not model the GTSP. The necessary additional requirement is the parity condition $z(\delta(v)) \in 2\mathbb{Z}$ for every $v \in V$. Such a constraint cannot be modeled directly, i.e., by adding linear inequalities. To obtain a correct model, one may add integer variables $y_v = \frac{1}{2}z(\delta(v))$ for every $v \in V$. Unfortunately, this does not contribute to the strength of the LP relaxation since relaxing y_v's integrality constraints essentially makes them redundant.

It is well-known that an optimum solution will never use an edge more than twice (otherwise we can decrease the value by 2 and obtain a better feasible solution). Hence we can restrict z_e to the set $\{0, 1, 2\}$ which allows us to perform binarization, i.e., to replace, for every $e \in E$, the variable z_e by $x_e^{(1)} + x_e^{(2)}$ with $x_e^{(1)}, x_e^{(2)} \in \{0, 1\}$. This again leads to the same strength of the LP relaxation, but now allows us to enforce parity constraints: since every closed walk traverses each cut $\delta(S)$ an even number of times, the following inequalities are valid for every $S \subseteq V$ and every $F \subseteq \delta(S) \times \{0, 1\}$ with $|F|$ odd.

$$\sum_{(e,i)\in(\delta(S)\times\{0,1\})\setminus F} x_e^{(i)} + \sum_{(e,i)\in F} (1 - x_e^{(i)}) \geq 1 \tag{8}$$

If we identify the variables $x_e^{(1)}$ and $x_e^{(2)}$ with the doubled edges of G, then these inequalities are the well-known *Blossom Inequalities*. For the TSP problem they model that every Hamiltonian cycle is also a perfect 2-matching.

Strengthened Constraints. To break the symmetry of $x_e^{(1)}$ and $x_e^{(2)}$ we add $x_e^{(1)} \geq x_e^{(2)}$ for each edge $e \in E$, and can further strengthen the inequalities using Theorem 1. Inequality

$$\sum_{e\in\delta(S)\setminus F} (x_e^{(1)} - x_e^{(2)}) + \sum_{e\in F} (1 - x_e^{(1)} + x_e^{(2)}) \geq 1 \tag{9}$$

is valid for every $S \subseteq V$ and every $F \subseteq \delta(S)$ and dominates (8). We can separate both constraints, (8) and (9) using the algorithm of Letchford et al. [4], making use of Theorem 2.

4 Strength of the Relaxation

We carried out computational experiments on the GTSP, and it turned out that the lower bound obtained by the linear relaxation (4)–(6) was never improved by binarization of the variables and addition of (strengthened) parity constraints. In fact, in the root node many (strengthened) Blossom Inequalities were added, but with no effect on the dual objective.

We investigated this effect, and observed the following weakness of the approach of binarization and addition of parity constraints. The decisive property of our model is that *only* parity constraints actually use the binarization variables $x_e^{(i)}$: all other variables consider the original variables z_e (or, equivalently, the sum of the binarization variables $x_e^{(i)}$).

In a more abstract setting we consider arbitrary (integer) variables $z_1, \ldots, z_k$ with domains $z_i \in [0, r_k] \cap \mathbb{Z}$ for all $i \in [k]$, and apply binarization, i.e., introduce variables $x_j^{(i)} \in P_{\text{ord}}^{r_i}$ and linking constraints $z_i = \sum_{j=1}^{r_i} x_j^{(i)}$ for each $i \in [k]$. We can assume that further (arbitrary) constraints that link the z-variables are also present.

Now consider a parity constraint on (a subset of) the z-variables, which is of course stated in terms of the corresponding x-variables. Suppose we have a certain fractional relaxation

solution $(\hat{z}, \hat{x})$ that may violate this parity constraint, but satisfies all other constraints. Our result in this section states that under mild conditions we can modify the x-variables such that the linking constraints are still satisfied (i.e., that the sums $\sum_{j=1}^{r_i} \hat{x}_j^{(i)}$ remain constant) and such that parity constraints are satisfied.

For this we define the function $\gamma(z) := \min(z, r - z, \frac{1}{2})$ for a variable $z \in [0, r]$.

Theorem 5. *Let $r \in \mathbb{N}^k$ and let $z \in [0, r_1] \times \ldots \times [0, r_k]$. Let $\mathcal{I}$ be a family of subsets $I \subseteq [k]$. If every $I \in \mathcal{I}$ satisfies $\sum_{i \in I} \gamma(z_i) \geq 1$, then there exist $x^{(i)} \in P_{ord}^{r_i}$ with $z_i = \sum_{j=1}^{r_i} x_j^{(i)}$ for all $i \in [k]$ such that for every $I \in \mathcal{I}$ the vector $(x^{(i)})_{i \in I}$ is contained in the even and odd parity polytopes corresponding to $(r_i)_{i \in I}$.*

The theorem essentially states sufficient conditions for the case that after binarization and enforcing of several parity constraints, the values of the original variables remain feasible. Note that in order to satisfy $\sum_{i \in I} \gamma(z_i) \geq 1$ for a constraint on variable set I, it suffices that two of the participating variables have a distance to their respective bounds of at least $\frac{1}{2}$, which is not very restrictive for nonbinary variables.

Implications for the Graphic TSP. We consider an optimum solutiuon $\hat{z} \in [0, 2]^E$ of the LP relaxation (4)–(6) of the model introduced in Section 3. The requirements for Theorem 5 are satisfied if and only if for every nontrivial cut $\delta(S)$ ($S \subsetneqq V$, $S \neq \emptyset$), the inequality $\sum_{e \in \delta(S)} \gamma(\hat{z}_e) \geq 1$ is satisfied. This is in particular the case if every cut contains two edges e, f with $\frac{1}{2} \leq \hat{z}_e, \hat{z}_f \leq \frac{3}{2}$. Note that integrality of the z-variables does not play a role here, i.e., even if $\hat{z}$ is integral, and $\hat{z}(\delta(S))$ has the wrong parity for some set S, then there may exist a (fractional) assignment for x-variables that is feasible for Constraints (9). For the IP model this means that the Blossom Inequalities only become useful if relevant z-variables are near their bounds or if branching or cutting restricted the x-variables further.

Acknowledgements. We thank the author of the `lrs` software [1] that we used to compute complete descriptions for ordered parity polytopes of small dimensions. Furthermore, we thank Stefan Weltge and Volker Kaibel for valuable discussions.

References

[1] David Avis. *A Revised Implementation of the Reverse Search Vertex Enumeration Algorithm*, pages 177–198. Birkhäuser Basel, 2000. ISBN 978-3-0348-8438-9.

[2] Pierre Bonami and François Margot. *Cut Generation through Binarization*, pages 174–185. Springer International Publishing, 2014. ISBN 978-3-319-07557-0.

[3] Robert Jeroslow. On defining sets of vertices of the hypercube by linear inequalities. *Discrete Mathematics*, 11(2):119–124, 1975.

[4] Adam N. Letchford, Gerhard Reinelt, and Dirk Oliver Theis. Odd minimum cut sets and b-matchings revisited. *SIAM Journal on Discrete Mathematics*, 22(4):1480–1487, 2008.

[5] Jean-Sébastien Roy. "Binarize and Project" to generate cuts for general mixed-integer programs. *Algorithmic Operations Research*, 2(1):37–51, 2007.

[6] András Sebő and Jens Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, pages 1–34, 2014. ISSN 1439-6912.

The Transportation Problem with Conflicts

Annette M.C. Ficker[1], Frits C.R. Spieksma[1], and Gerhard J. Woeginger[2]

[1]KU Leuven
[2]RWTH Aachen

1 Introduction

In the classical Transportation Problem we are given suppliers, each having a supply, and locations, each having a demand. For all possible combinations of supplier and location we are given a unit transportation cost. The goal is to fulfill the demand with minimum cost. This problem is well-known and efficiently solvable.

Many situations in practice have, as a base, this transportation problem. However, additional properties are often present. To illustrate this, consider a setting described in [5], where patients (suppliers) have to be allocated to hospital rooms (locations), with the additional constraint that each room should only contain patients of the same gender. We call a pair of patients with different gender a *forbidden pair*, and further, we call the set of forbidden pairs the *conflict set*. This example gives rise to the so-called Red-Blue Transportation problem.

Another example, see [1], comes from storage management where containers (suppliers) need to be placed in rows of a storage yard (locations), such that costs of operations (search, load) is minimized. Some containers are not allowed to be placed in the same row, due to their content or size. Again, two containers that cannot be placed in the same row are called a forbidden pair, and the set of forbidden pairs for a particular row form its conflict set. The resulting situation gives rise to the Transportation Problem with Exclusionary Side Constraints.

Our last example comes from [2], where companies (suppliers) want to promote their products to potential customers (locations). On the one hand, a customer wants to limit the number of promotions received from similar companies, inducing forbidden pairs of companies for each customer. On the other hand, companies want to geographically spread their promotion and therefore limit the number of promotions to customers living close to each other, inducing forbidden pairs of customers. This problem is called the Conflict-aware weighted Bipartite b-matching problem.

The Transportation Problem with Conflicts (TPC) generalizes all the problems mentioned above, by extending the input of the classical transportation problem with a conflict set for each supplier and each location.

Problem statement

In the Transportation Problem with Conflicts (TPC) we are given a bipartite graph $(S \cup D, E)$, where (see also Figure 1):

S: is the set of supply nodes (suppliers), with for each $i \in S$,

- a supply $s_i \in \mathbb{N}$, and
- a conflict set C_i, i.e., a collection of pairs of demand nodes.

D: is the set of demand nodes (locations), with for each $j \in D$,

- a demand $d_j \in \mathbb{N}$, and
- a conflict set F_j, i.e., a collection of pairs of supply nodes.

E: is the edge set (not necessarily complete), with for each edge $(i, j) \in E$ ($i \in S$ and $j \in D$),

- a capacity $u_{ij} \in \mathbb{N}$, and
- a weight $w_{ij} \in \mathbb{N}$.

$t \in \mathbb{N}$: is the threshold for the number of allowed conflicts to occur in an assignment.

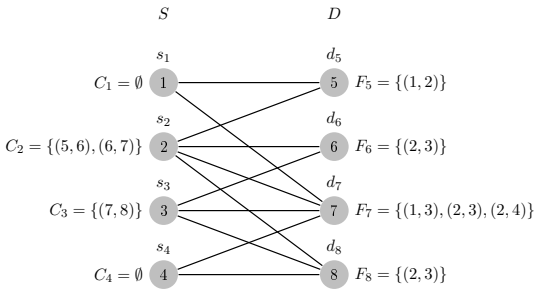


Figure 1: Transportation Problem with Conflicts

Similar to the classical Transportation Problem we assume that $\sum_{i \in S} s_i = \sum_{j \in D} d_j$. A solution is an integral assignment $x_{ij} \in \mathbb{N}$ indicating for each edge $(i, j) \in E$, how much supply is sent from supply node $i \in S$ to demand node $j \in D$. The value of an assignment equals $\sum_{i \in S} \sum_{j \in D} w_{ij} \cdot x_{ij}$.

We say that demand (supply) constraints are **fulfilled** if for all $j \in D$, $\sum_{i \in S} x_{ij} = d_j$ (for all $i \in S$, $\sum_{j \in S} x_{ij} = s_i$). We say that demand (supply) constraints are **respected** if for all $j \in D$, $\sum_{i \in S} x_{ij} \leq d_j$ (for all $i \in S$, $\sum_{j \in S} x_{ij} \leq s_i$).

Given a solution x, we say that a conflict occurs if $x_{ij_1} > 0$ and $x_{ij_2} > 0$, while $(j_1, j_2) \in C_i$. Likewise, a conflict occurs if $x_{i_1 j} > 0$ and $x_{i_2 j} > 0$, while $(i_1, i_2) \in F_j$. An assignment x is not feasible when more than t conflicts occur or when $x_{ij} > u_{ij}$ for an edge $(i, j) \in E$.

We consider two problems,

i) Find a feasible assignment fulfilling all demand and supply constraints, while minimizing total cost (min-TPC), and

ii) Find a feasible assignment respecting all demand and supply constraints, while minimizing total cost (max-TPC).

Conflict graphs

We analyze the TPC, and special cases of the TPC, by investigating possible structures within the conflict set. We build a *conflict graph* for each location as follows: there is a node for each supplier adjacent to the location and two nodes are connected if and only if the corresponding suppliers constitute a forbidden pair. A similar procedure is used to build a conflict graph for each supplier. For example, see Figures 2 & 3 for the conflict graphs of C_1 and F_7 from Figure 1.

Figure 2: Conflict Graph G_{C_1}

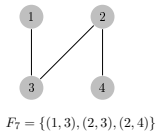


Figure 3: Conflict Graph G_{F_7}

Notation

We discuss several special cases of TPC and use a four-field notation to systematize the different special cases of TPC: TPC$(\alpha, \beta, \chi, \delta)$.

Definition 1. *TPC$(\alpha, \beta, \chi, \delta)$ is a special case of TPC where:*

$\alpha \in \{C_i, C, \emptyset\}$ *describes the nature of the conflict sets of the supply nodes; the three symbols stand for:*

- C_i*: arbitrary conflict sets, i.e. each supply node $i \in S$ has a conflict set,*
- C*: identical conflict sets, i.e. each supply node $i \in S$ has conflict set C,*
- $\emptyset$*: no conflict between demand nodes, i.e. each supply node $i \in S$ has an empty conflict set;*

$\beta \in \{F_j, F, F^B, \emptyset\}$ *describes the nature of the conflict sets of the demand nodes; the four symbols stand for:*

- F_j*: arbitrary conflict sets, i.e. each demand node $j \in D$ has a conflict set,*
- F*: identical conflict sets, i.e. each demand node $j \in D$ has conflict set C,*
- F^B*: identical conflict sets, and its induced conflict graph G_F is complete Bipartite,*
- $\emptyset$*: no conflict between supply nodes, i.e. each demand node $j \in D$ has an empty conflict set;*

$\chi \in \{u_{ij}, \infty, 1\}$ *describes the nature of the edge capacities; the three symbols stand for:*

- u_{ij}*: arbitrary capacities,*
- ∞*: uncapacitated, or equivalently $u_{ij} \geq \min\{s_i, d_j\}$,*
- 1*: $u_{ij} = 1$, for every edge $(i,j) \in E$,*

$\delta \in \{t, 0\}$ *describes the nature of the threshold; the two symbols stand for:*

- t*: arbitrary threshold, i.e. part of the input,*
- 0*: no conflicts are allowed.*

Our results

Note that when threshold $t = 0$, then, in a feasible assignment, the suppliers $i \in S$ assigned to a location $j \in D$ (i.e. those $i \in S$ with $x_{ij} > 0$) form an independent set in the conflict graph of that location $j \in D$. Indeed, since apparently none of the suppliers assigned are in conflict, there is no edge connecting them in the conflict graph.

Since finding a maximum independent set is difficult, so is solving an instance of TPC with threshold $t = 0$. However, there are graph classes for which independent set is easy, polynomial solvable. Therefore we have complexity results based on these graph classes and approximation results depending on the approximation factor for finding an independent set for each location.

We show that:

(i) TPC$(\emptyset, F, \infty, 0)$ remains NP-hard, even if F is a bipartite graph, or an interval graph or a planar graph,

(ii) TPC$(\emptyset, F, u_{ij}, 0)$ remains NP-hard, even if F is a simple path (tree) or a matching (forest),

(iii) TPC$(\emptyset, F^B, \infty, 0)$ remains NP-hard, even if supply $s_i = 1$ for each supply node $i \in S$ and all demand $d_j = 2$ for each demand node $j \in D$ (thereby settling a case left open in [5]),

(iv) TPC$(\emptyset, F_j, \infty, 0)$ remains NP-hard, even with bounded degree 6.

Regarding approximation, suppose we have a β_j-approximation for finding a MWIS, containing at most d_j nodes, in the conflict graph G_{F_j} induced by F_j for each $j \in D$.

Let $\beta = \max_{j \in D} \beta_j$, then by using results in [3] we obtain a $\beta(1 - 1/e)$-approximation algorithm for max-TPC$(\emptyset, F_j, \infty, 0)$ with supply $s_i = 1$ for each supply node $i \in S$.

References

[1] Cao, B. (1992). Transportation problem with nonlinear side constraints a branch and bound approach. *Zeitschrift für Operations Research*, 36(2):185–197.

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

[3] Fleischer, L., Goemans, M. X., Mirrokni, V. S., and Sviridenko, M. (2011). Tight approximation algorithms for maximum separable assignment problems. *Mathematics of Operations Research*, 36(3):416–431.

[4] Goossens, D. R. and Spieksma, F. C. R. (2009). The transportation problem with exclusionary side constraints. *4OR*, 7(1):51–60.

[5] Vancroonenburg, W., Della Croce, F., Goossens, D., and Spieksma, F. C. R. (2014). The red blue transportation problem. *European Journal of Operational Research*, 237(3):814 – 823.

A 3/2-Approximation Algorithm for Tree Augmentation via Chvàtal-Gomory Cuts

Samuel Fiorini[1]

[1]Université libre de Bruxelles

There are two main ingredients in this talk: (i) the weighted tree augmentation problem (WTAP), a fundamental network design problem, for which one can derive a factor-2 approximation algorithm in many different ways from known results, but at the same time no existing technique is currently permitting to go beyond factor-2; (ii) Chvátal-Gomory cuts, a general technique to strengthen LP relaxations, that seems to have been rarely if not never successfully used in approximation algorithms.

In the WTAP, we are given an undirected tree G, an additional set of edges called *links* and costs on the links. The goal is to choose a minimum cost subset of links such that adding them to G yields a 2-edge-connected graph.

The unweighted version of WTAP, known as the tree augmentation problem (TAP), is already APX-hard. That problem has attracted a lot of attention, and has seen an abundance of results, most of them technically involved. The best result so far is the combinatorial 3/2-approximation algorithm of Kortsarz and Nutov (TALG '16). In order to address some issues with the correctness of this last paper, Cheriyan and Gao (arXiv '15) recently derived an SDP-based $3/2 + \varepsilon$-approximation algorithm.

For WTAP, virtually no progress was made, until Adjiashvili (SODA '17) found a LP-based 1.9642-approximation algorithm in the case all weights are between 1 and some constant M. This work brought a slick new decomposition tool to the problem.

In my talk, I will explain how to combine Adjiashvili's decomposition, a stronger LP obtained after one round of $\{0, 1/2\}$-Chvátal-Gomory cuts, and seminal work by Edmonds and Johnson on matching in bidirected graphs, and obtain a $3/2 + \varepsilon$-approximation for WTAP under the bounded weights assumption. This is the current best result on WTAP, but at the same time the less technical.

Joint work with Martin Groß, Jochen Könemann, Laura Sanità (Waterloo).

Distances between bicliques and structural properties of bicliques in graphs

Marina Groshaus[1] and Leandro Montero[2]

[1]Departamento de Computación. Universidad de Buenos Aires. Buenos Aires, Argentina, groshaus@dc.uba.ar

[2]Interdisciplinary Centre for Security, Reliability and Trust. Université de Luxembourg. L-2721, Luxembourg, leandro.montero@uni.lu

A *biclique* is a maximal bipartite complete induced subgraph of G. The *biclique graph* of a graph G, denoted by $KB(G)$, is the intersection graph of the family of all bicliques of G. In this work we first define the distance between bicliques in a graph. We give a useful formula that relates the distance between bicliques in a graph G and the distance between their respectives vertices in $KB(G)$. Using it, we prove several results about the structure of bicliques in graphs and biclique graphs.

1 Introduction

The *biclique graph* of a graph G, denoted by $KB(G)$, is the intersection graph of the family of all bicliques of G. It was defined and characterized in [3]. However, no polynomial time algorithm is known for recognizing biclique graphs. Bicliques have applications in various fields, for example, biology: protein-protein interaction networks [2], social networks: web community discovery [4], genetics [1], medicine [5], etc.

In this work, we first define the distance between bicliques in a graph. We give a formula that relates the distance between bicliques in a graph G and the distance between their respectives vertices in $KB(G)$. This is an important tool for proving some results on bicliques in graphs. In particular, given two bicliques, we show the existence of more bicliques between them. Also, we give a different (and easier) proof of the necessity theorem for a graph to be a biclique graph stated in [3]. Finally, we prove that the condition of that theorem is not sufficient.

Along the paper we restrict to undirected simple graphs. Let $G = (V, E)$ be a graph with vertex set $V(G)$ and edge set $E(G)$. A *biclique* is a maximal bipartite complete induced subgraph of G. We assume that all graphs in this paper are connected. Given a family of sets $\mathcal{H}$, the *intersection graph* of $\mathcal{H}$ is a graph that has the members of $\mathcal{H}$ as vertices, and there is an edge between two sets $E, F \in \mathcal{H}$ when E and F have non-empty intersection. A graph G is an *intersection graph* if there exists a family of sets $\mathcal{H}$ such that G is the intersection graph of $\mathcal{H}$.

Theorem 1 ([3]). *Let G be a graph such that $G = KB(H)$, for some graph H, then every induced P_3 of G is contained in an induced diamond or an induced gem of G (Fig 1).*

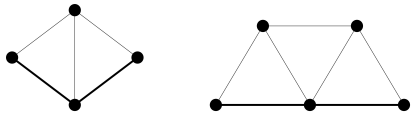


Figure 1: Induced P_3 in bold edges contained in a *diamond* and in a *gem* respectively

One natural question that arises from Theorem 1 is if this is also sufficient, that is, if it holds that a graph G is a biclique graph if and only if every induced P_3 is contained in an induced *diamond* or in an induced *gem*. We will show that this is not the case, by proving a result that gives us many graphs with that property that are not biclique graphs.

2 Distances in G and $KB(G)$

In this section we define the distance between bicliques in a graph. Also, we study the relation between the distance of bicliques in a graph G and the distance between their respective vertices in $KB(G)$.

Definition 2. *Let G be a graph and let B, B' be bicliques of G. We define the* distance *between B and B' as $d(B, B') = min\{d(b, b') \ / \ b \in B,\ b' \in B'\}$.*

The next lemma is important for giving an easier proof of Theorem 1 and also for proving that its condition it is not sufficient.

Lemma 3. *Let G be a graph and let B, B' be bicliques of G. Then* $d_{KB(G)}(B, B') = \lfloor \frac{d_G(B,B')+1}{2} \rfloor + 1$.

Now, based on the distance between two bicliques of a graph G we can assure the existence of other bicliques between them.

Lemma 4. *Let G be a graph and let B, B' be bicliques of G such that $d_G(B, B') = 1$, then there exist at least two more bicliques in G such that they are intersecting and they also intersect with B and B'.*

The following result is a generalization of Lemma 4.

Theorem 5. *Let G be a graph and let B, B' be bicliques of G such that $d_G(B, B') = k > 1$, then there exist at least $k + 1$ bicliques in G such that they are at distance at most $k - 1$ to each of B and B'.*

Proof. Let $v_0 \in B$ and $v_k \in B'$ be vertices such that $d_G(v_0, v_k) = d_G(B, B') = k$. Then, there exists a path $P = v_0 v_1 \dots v_k$ of length k between B and B' (in fact between v_0 and v_k). Clearly v_i is not adjacent to v_j for $0 \leq i < j \leq k$ and $j \neq i + 1$, otherwise a shortest path would exist between B and B'. Then, each triple $\{v_i, v_{i+1}, v_{i+2}\}$ is contained in a different biclique of G for $i = 0, \dots, k - 2$. Therefore we obtain $k - 1$ bicliques that are at distance at most $k - 1$ to each of B and B'. We obtain the two remaining bicliques as follows. As B is a biclique, there exists a vertex $x \in B$ such that $xv_0 \in E(G)$. If $xv_1 \notin E(G)$, then $\{x, v_0, v_1\}$ is contained in a biclique of G different from B. Now if $xv_1 \in E(G)$ then $\{x, v_1, v_2\}$ is contained in a biclique

of G different from B. It is worth to mention that $xv_i \notin E(G)$ for $i \geq 2$, otherwise a path of length less than k would exist between B and B'. Finally, we obtain the remaining biclique in the same way taking a vertex $y \in B'$ such that $yv_k \in E(G)$. □

Now we give the proof of Theorem 1 based on distances between bicliques.

Proof of Theorem 1. Let uvw be an induced P_3 in G and let U, V and W be the bicliques of H associated to the vertices u, v and w of G. As $d_G(u, w) = 2$, by Lemma 3 we have $d_H(U, W) = 2$ or $d_H(U, W) = 1$.

- Case $d_H(U, W) = 2$. Therefore, there is no edge between bicliques U and W. Now as V intersects with U and with W we have that V contains a $P_3 = \{a, b, c\}$ such that $a \in U$ and $c \in W$. Since $d_H(U, W) = 2$ and the path joining U and W uses the edges in V, we have by Theorem 5 that there are three bicliques between U and W in H. Furthermore, one of these bicliques is V. Let Z_1 and Z_2 be the other two bicliques in H and z_1 and z_2 their associated vertices in G. Following the proof of Theorem 5, we can see that Z_1 intersects with U, V and Z_2, and Z_2 intersects with V, W and Z_1. Now, depending on the intersection between U, Z_2 and W, Z_1 we conclude that either $\{u, v, w, z_2\}$, $\{u, v, w, z_1\}$ induce a *diamond* in G or $\{u, v, w, z_1, z_2\}$ induces a *gem* in G, that contains the P_3.
- Case $d_H(U, W) = 1$. Therefore, there exists an edge e between U and W. Now, if there is any of these edges such that $\{e\} \cap V \neq \emptyset$, then by Lemma 4, there are two intersecting bicliques (at least one of them must contain e) between U and W that also intersect U and W. If V is one of these two (if not, V intersects at least one of them since $\{e\} \cap V \neq \emptyset$), calling Z the biclique different from V, we have that $\{u, v, w, z\}$ is an induced *diamond* that contains the P_3 in G. Finally, for every edge e between U and W, $\{e\} \cap V = \emptyset$. Therefore, V contains a P_3 with one endpoint in U and the other in W. Now, as no edge between U and W intersects with V, using the edges in that P_3 in V, we obtain the same conclusion as in the first case.

□

Finally we show that there exist many graphs having every induced P_3 contained in an induced *diamond* of G that are not biclique graphs. That is, the condition of Theorem 1 is not sufficient. We recall that for a vertex v of G, $N(v)$ is the set of vertices adjacent to it.

Proposition 6. *Let $G = KB(H)$ for some graph H where $G \neq$ diamond. Then, there do not exist $v_1, v_2 \in V(G)$ such that $N(v_1) = N(v_2) = K_2$.*

Proof (Sketch) Suppose that there exist $v_1, v_2 \in V(G)$ such that $N(v_1) = N(v_2) = K_2$. Then, $d_G(v_1, v_2) = 2$ and therefore, if B is the biclique of H that corresponds to v_1 and B' is the biclique of H that corresponds to v_2, by Lemma 3, $d_H(B, B') = 2$ or $d_H(B, B') = 1$. We show only the first case since the second one is similar, applying first Lemma 4.

- Case $d_H(B, B') = 2$. Now, H must contain a subgraph as depicted in Figure 2. We will show that we arrive to a contradiction.

  Suppose first that one of both dotted edges does not exist, say vy. Then $\{v, x, y\}$ is contained in a biclique that does not intersect B'. This is a contradiction since $N(v_1) = N(v_2)$ in G. Suppose next that both dotted edges vy and vy' exist. Now in H there are at least four bicliques that intersect with B and B'. We obtain one for each choice of $\{x, y\}$ in B, $\{x', y'\}$ in B' and v. Then $K_2 \neq K_4 \subseteq N(v_1) = N(v_2)$ which is a contradiction.

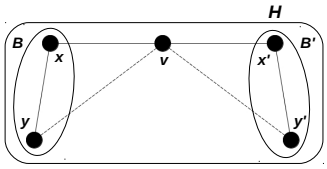


Figure 2: Graph H when $d_H(B, B') = 2$.

□

Figure 3 shows some examples of graphs having every P_3 in a *diamond* that are not biclique graphs.

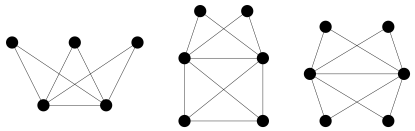


Figure 3: Examples of graphs that are not biclique graphs

Proposition 6 can be extended leading to the following conjecture.

Conjecture 7. *Let $G = KB(H)$ for some graph H where $G \neq$ diamond. Then, there does not exist $v_1, v_2, \ldots, v_i \in V(G)$ such that $N(v_1) = N(v_2) = \ldots = N(v_i) \subseteq K_i$ for $i \geq 2$.*

Using Conjecture 7, we could prove the following one.

Conjecture 8. *Let $G = KB(H)$ for some graph H. Then, G has a Hamiltonian cycle.*

References

[1] G. Atluri, J. Bellay, G. Pandey, C. Myers, and V. Kumar. Discovering coherent value bicliques in genetic interaction data. In *Proceedings of 9th International Workshop on Data Mining in Bioinformatics (BIOKDD'10)*, 2000.

[2] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, N. Zhang, G. Li, and R. Chen. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Research*, 31(9):2443–2450, 2003.

[3] M. Groshaus and J. L. Szwarcfiter. Biclique graphs and biclique matrices. *J. Graph Theory*, 63(1):1–16, 2010.

[4] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber-communities. In *Proceeding of the 8th international conference on World Wide Web, pages 14811493, 1999.*, 2000.

[5] N. Nagarajan and C. Kingsford. Uncovering genomic reassortments among influenza strains by enumerating maximal bicliques. *2012 IEEE International Conference on Bioinformatics and Biomedicine*, 0:223–230, 2008.

Determination of Large Sparse Derivative Matrices: Structural Orthogonality and Structural Degeneracy

Shahadat Hossain[1] and Ashraful Huq Suny[2]

[1,2]University of Lethbridge

We consider the problem of minimum cardinality column partitioning that arise in the determination of large and sparse/structured matrices of mathematical derivatives. We present an effective approach whereby a special submatrix or core is identified and partitioned. The partition is then extended to the entire matrix. Results from computational experiments are highly encouraging. We establish, for the first time, optimal partition for a set of test instances.

1 Introduction

Numerical methods for solving problems in nonlinear optimization and differential equations often require the evaluation of mathematical derivatives or sensitivities of some objective. In this paper we consider the determination of Jacobian matrix $F'(x)$ of a once continuously differentiable mapping $F : \mathbb{R}^n \mapsto \mathbb{R}^m$ at a given point $x \in \mathbb{R}^n$. The product of the Jacobian matrix with a vector s may be approximated as

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

with one evaluation of F at $(x + \epsilon s)$ assuming $F(x)$ has already been computed, where $\epsilon > 0$ is a small increment. If the function is available as a computer program then the techniques of algorithmic (or Automatic) Differentiation (AD) [5] can be applied to compute $b = F'(x)s$ accurate up to the machine round-off, at a cost which is a small multiple of the cost of one function evaluation. The evaluation of the Jacobian at a given point can therefore be achieved as matrix-vector products along some judiciously chosen directions s. If $F'(x)$ is dense then one can set s to the Cartesian basis vectors $e_i, i = 1, \ldots, n$ to obtain matrix A with n products As. When the Jacobian has many entries that vanish identically, it is possible to exploit such information to reduce the computational cost. The key observation is to approximate a group of columns in each product by utilizing sparsity. Columns j and l are structurally orthogonal, written $A(:, j) \perp A(:, l)$, if there does not exist index i such that $a_{ij} \neq 0$ and $a_{il} \neq 0$. Columns j and l are structurally dependent, written $A(:, j) \not\perp A(:, l)$, if they are not structurally orthogonal. For $A(:, j) \perp A(:, l)$, we have

$$A(:, j) + A(:, l) \approx \frac{1}{\epsilon}[F(x + \epsilon(e_j + e_l)) - F(x)], \tag{2}$$

such that the nonzero unknowns in columns j and l are determined from the product $b = As, s = e_j + e_l$ *directly*. With a priori known sparsity pattern of a sparse matrix our goal is to partition its columns into smallest number of structurally orthogonal groups such that the nonzero unknowns can be determined directly.

> Obtain vectors $s_j \in \mathbb{R}^n, j = 1, \ldots, p$ where p is minimized such that the products $b_j = As_j, j = 1, \ldots, p$ or $B = AS$ determine the matrix A directly.

Determining a sparse Jacobian matrix by exploiting sparsity as in (2) is due to Curtis, Powell, and Reid [1] (henceforth the CPR method). Coleman and Moré [2] further analyze the column partitioning problem and formally show that the problem of finding a minimum cardinality column partitioning *consistent with direct determination* is equivalent to a vertex coloring problem of an associated graph and that the problem is NP-Hard.

2 Matrix Partitioning and Degeneracy

The sparsity pattern of a matrix $A \in \mathbb{R}^{m \times n}$ is denoted

$$\mathcal{S}(A) = \{(i, j) | a_{ij} \neq 0\}$$

where a_{ij} is the entry in row i and column j. We also use colon notation of [4] to denote submatrices; column j and row i of matrix A are $A(:, j)$ and $A(i, :)$, respectively. More generally, for $\mathcal{I} \subset \{1, \ldots, m\}$ and $\mathcal{J} \subset \{1, \ldots, n\}$, $A(\mathcal{I}, \mathcal{J})$ denotes the submatrix of A consisting of rows and columns (whose indices are) in $\mathcal{I}$ and $\mathcal{J}$, respectively. Note that sparsity pattern of a matrix is invariant under column permutations in the sense that a column permutation alters only the labeling. The set of neighbors of column j in the submatrix induced by columns $\mathcal{J}$, is defined as

$$\mathcal{N}_{\mathcal{J}}(A(:, j)) = \{A(:, l), l \in \mathcal{J} \mid l \neq j, A(:, j) \not\perp A(:, l)\}.$$

In other words, neighbors of $A(:, j)$ are $A(:, l), l \in \mathcal{J}$ that are structurally dependent on $A(:, j)$. The degree (of dependency) of column j is the number of neighbors of $A(:, j)$ and is denoted by $d_{\mathcal{J}}(A(j, :)) = |\mathcal{N}_{\mathcal{J}}(A(:, j))|$. For brevity and when there is no ambiguity we use simplified notations: $\mathcal{N}_{\mathcal{J}}(j) \equiv \mathcal{N}_{\mathcal{J}}(A(:, j))$, $d_{\mathcal{J}}(j) \equiv d_{\mathcal{J}}(A(j, :))$. The minimum degree (of dependency) in the submatrix induced by columns $\mathcal{J}$ is defined as

$$\delta_{\mathcal{J}} = \min \{d_{\mathcal{J}}(j) | j \in \mathcal{J}\}.$$

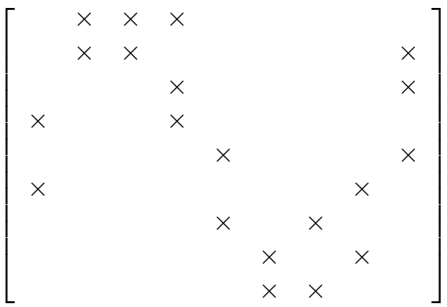
It has been shown that direct determination where the number of matrix-vector products minimized is NP-hard [2, 7]. If there is an effective procedure for finding mutually dependent columns of largest cardinality then these columns can be trivially partitioned and the partition can be extended to include the remaining columns. Unfortunately, no such effective procedure exists for finding a maximum dependent set of columns [7].

Let ρ_i denote the number of nonzero entries in row i of matrix A. Then $\rho = \max_i \rho_i, i = 1, \ldots, m$ is a lower bound on the number of groups in a structurally orthogonal column partition. However, such a bound can be arbitrarily poor [6]. In our work, we consider a relaxation of mutual structural dependency of the columns. Specifically, we are interested in finding the largest submatrix A' induced by columns $\mathcal{J}$ of matrix A such that

- $\delta_{\mathcal{J}}$ is maximum over all induced submatrices of A,

- $\mathcal{J}$ is computationally easy to find.

The submatrix A' is termed *structural δ-core* of columns of matrix A and $\delta \equiv \max_{\mathcal{J} \subset \{1,\ldots,n\}} \delta_{\mathcal{J}}$ is the *structural column degeneracy* (or *degeneracy* for short) of matrix A. It can be shown that the columns of A' can be partitioned into $\delta_{\mathcal{J}} + 1$ structurally orthogonal groups and that for each column j not in A' there is a column group of A' such that column j is structurally orthogonal to the columns in that group. Thus, matrix A also has a $\delta_{\mathcal{J}} + 1$ structurally orthogonal partition. Moreover, an algorithm for finding a "good" structurally orthogonal partition of columns of A can be specified by finding a minimum partition of the columns of A' and then extending the partition to matrix A. Borrowing ideas from [11] we give an efficient linear-time algorithm for identifying structural δ-core of columns of a sparse matrix A. The algorithm proceeds by identifying a column with minimum degree at each step. We illustrate this algorithm using an example. The left part of Figure 1 displays a sparse matrix; the table on the right demonstrates the algorithm for structural for finding δ-core.



Step	$\mathcal{J}$	j	$\delta_{\mathcal{J}}$
0	$\{1,\ldots,9\}$	6	2
1	$\{1,\ldots,5,7,\ldots,9\}$	8	1
2	$\{1,\ldots,5,7,9\}$	1	1
3	$\{2\ldots,5,7,9\}$	7	1
4	$\{2,\ldots,5,9\}$	5	1
5	$\{2,3,4,9\}$	3	3
6	$\{2,4,9\}$	9	2
7	$\{2,4\}$	2	1
8	$\{4\}$		0

Figure 1: Core decomposition of a sparse matrix

Columns $\{1,5,6,7,8\}$ each has degree 2 and the remaining columns are of degree 3. We arbitrarily pick column 6. Removing column 6 reduces the degree of columns 7 and 8 by one. The degree of other columns is not affected. Let the next minimum degree column be column number 7. The table in Figure 1 depicts the minimum degree column (j) that is picked at each step. At step 5, we are left with submatrix consisting of columns $2,3,4$ and 9. It can be directly verified that the columns $2,3,4$ and 9 constitute the maximum cardinality structurally dependent set of columns for matrix A which is also the structural δ-core for $\delta = 3$. The concepts of degeneracy and core are well-known in graph theory [10]. While the algorithm that we have described above has an equivalent interpretation in terms of graphs [11], it is to be emphasized that in a computer implementation a sparse matrix is preferable as many of the combinatorial tasks can be expressed as fundamental sparse matrix kernel operation such as sparse matrix-vector multiplication or sparse matrix-matrix multiplication [6].

3 Numerical Experiments

In this section we provide preliminary numerical test results on test instances drawn from MatrixMarket collection. In Table 1 labels of the columns are as follows: n denotes the

Table 1: Number of matrix-vector products in direct determination

Matrix	n	ρ	δ	$\lvert\mathcal{J}\rvert$	DSJM	BB	CM
cage11	39082	31	141	832	62	56*	**60**
cage12	130228	33	162	1710	68	61*	**65**
fidap002	441	125	129	275	125	125	**125**
fidap003	1821	62	65	482	62	54	**62**
fidap005	27	15	14	21	15	15	**15**
fidap015	6867	18	25	4482	22	18	**18**
fidap018	5773	18	25	2710	22	18	**18**
fidap022	839	62	70	155	64	62	64
fidap029	2870	9	12	2643	12	9	**9**
fidap031	3909	75	107	2322	93	90*	**90**
fidap033	1733	18	23	1054	21	18	**18**
fidap035	19716	18	25	7252	22	18	**18**
fidapm05	42	21	20	30	21	21	**21**

number of columns in the matrix, ρ denotes the maximum number of nonzero entries in any row, δ denotes the degeneracy, and $\mathcal{J}$ denotes structural δ-core. Each matrix is partitioned using a heuristic in the software package DSJM [9]; the number of group in the structurally orthogonal partition is reported in column labeled DSJM. We apply Brélaz's exact coloring algorithm, a branch-and-bound method, on the structural δ-core; the resulting partition is reported in column BB. In our combined method, reported under column CM, we first find a structurally orthogonal partition of the structural δ-core using the exact method BB and then extend the partition using a greedy approach. Each column that is not in the core is included in the least numbered structurally orthogonal column group. The numbers in **bold** represent improvement or new information. On *cage11*, *cage12*, and *fidap031* the exact algorithms does not terminate. However, we are still able to use the partitioning information and extend it to the entire matrix. The resulting partition is better than the one computed by DSJM. Also, for a vast majority of test instances the exact algorithm does not terminate when applied to the entire matrix. We can verify the optimality of partitions from the values ρ, δ, BB, and CM. On problems *fidap029*, *fidap033*, *fidap035*, we have equality of lower bound (ρ) and upper bound (CM). Therefore, the CM partition is minimum.

4 Concluding Remarks

We have presented a new approach that uses a partial structurally orthogonal column partition obtained by applying an exact procedure on a much smaller submatrix and then extending the partial partition to the entire matrix. Except for small test instances determining minimum structurally orthogonal column partition is impractical. Using our approach we are able to determine and verify optimal partition of a number of test instances for the first time. The full version of the paper will include relevant theory and elaborate test results.

Acknowledgement. *This research is supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant (Individual).*

References

[1] Curtis, A. R., Powell, M. J. D., Reid, J. K.: On the Estimation of Sparse Jacobian Matrices. IMA J. Appl. Math. **13(1)**, 117–119 (1974)

[2] Coleman, T. F., Moré J. J.: Estimation of Sparse Jacobian Matrices and Graph Coloring Problems. SIAM J. Numer. Anal. **20(1)**, 187–209 (1983)

[3] Golub, G. H. and Van Loan, C. F.: Matrix Computations (3rd Ed.). Johns Hopkins University Press, Baltimore, MD, USA (1996)

[4] Griewank, A. and Walther, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation (Second Ed.). Soc. for Industrial and Applied Math., Philadelphia, PA, USA. (2008)

[5] Hossain, S., Steihaug, T.,: Graph models and their efficient implementation for sparse Jacobian matrix determination. Discrete Appl. Math. **161(2)**, 1747–1754 (2013)

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

[7] Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[8] Mahmudul Hasan, Shahadat Hossain, Ahamad Imtiaz Khan, Nasrin Hakim Mithila, and Ashraful Huq Suny. DSJM: a software toolkit for direct determination of sparse Jacobian matrices. In *Proceedings of the 5th International Conference on Mathematical Software, ICMS*, volume 9725 of *LNCS*, pages 275–283, Berlin,Germany, 2016. Springer.

[9] Don R Lick and Arthur T White. k-Degenerate graphs. *Canadian J. of Mathematics*, 22:1082–1096, 1970.

[10] David W Matula and Leland L Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the ACM (JACM)*, 30(3):417–427, 1983.

Computing kernels in graphs with a clique-cutset

Ayumi Igarashi[1], Frédéric Meunier[2], and Adèle Pass-Lanneau[2]

[1]University of Oxford, Department of Computer Science
[2]Université Paris Est, CERMICS (ENPC)

In a directed graph, a kernel is a subset of the vertices that is both independent and absorbing. Not all directed graphs have a kernel, and finding classes of graphs having always a kernel or for which deciding the existence of a kernel is polynomial has been the topic of many works in graph theory. We formalize some techniques to build a kernel in a graph with a clique-cutset, knowing kernels in the pieces with respect to the clique-cutset. As a consequence, we obtain for instance that computing a kernel in a clique-acyclic orientation of a chordal graph can be done in polynomial time. We enlighten some consequences in the theory of hedonic games.

1 Introduction

A subset S of the vertices of a directed graph is *independent* if no two vertices in S are adjacent, and *absorbing* if for any vertex u not in S, there is a vertex $v \in S$ such that the arc (u, v) exists in the graph. A *kernel* is a subset of vertices that is both independent and absorbing. Kernels have been introduced in 1944 by Von Neumann and Morgenstern [15] as a tool for studying positional or Nim-type games. Since then, other applications in game theory have been found [3, Sections 7 and 8]. They also play a role in graph theory: they are for instance at the heart of Galvin's proof of Dinitz's conjecture on list coloring [8].

A directed graph such that every induced subgraph has a kernel is *kernel-perfect*. Identifying classes of kernel-perfect graphs has been the motivation of many works, see the bibliography in [3]. A *clique-cutset* of a directed graph $D = (V, A)$ is a subset $C \subseteq V$ such that C induces a clique in D and $D[V \setminus C]$ is disconnected. For each connected component B of $D[V \setminus C]$, the directed graph induced by $B \cup C$ is a *piece* of D with respect to C. Jacob [11] proved that if every piece with respect to some clique-cutset C is kernel-perfect, so is D. Jacob's theorem has been one of the main tools used by Maffray for proving his result about kernels in i-triangulated graphs [12]. We prove the following lemma, which strengthens Jacob's theorem and adds to it an algorithmic flavor.

Lemma 1. *Let C be a clique-cutset of a digraph $D = (V, A)$ and B, B' a bipartition of $V \setminus C$ such that $B \cup C$ is a piece of D with respect to C. Suppose that there exist subsets of vertices $K^{(i)}$ such that $K^{(i)}$ is a kernel of $D\left[B \cup \left(C \setminus \bigcup_{j=1}^{i-1} K^{(j)}\right)\right]$ for every $i \in \{1, \dots, |C| + 1\}$ and such that $D\left[B' \cup \left(C \cap \bigcup_{j=1}^{|C|} K^{(j)}\right)\right]$ has a kernel K. Then there exists $i \in \{1, \dots, |C| + 1\}$ such that $K \cup K^{(i)}$ is a kernel of D.*

This lemma implies in particular that, if $B \cup C$ and $B' \cup C$ induce kernel-perfect graphs, then D is kernel-perfect as well. We can then see that Lemma 1 implies Jacob's theorem by applying it recursively on the directed graph induced by $B' \cup C$. It is worth noting that the proof of our lemma is much shorter than the proof Jacob gave for his theorem. We provide the full proof at the end of this extended abstract.

In a graph class closed under taking induced subgraphs, the *atoms* are the graphs that have no clique-cutset. According to Jacob's theorem, if the atoms of such a class are kernel-perfect, so are all graphs in the class. The following theorem, proved almost directly from Lemma 1, ensures that if kernels are polynomially computable in the atoms of this class and in their induced subgraphs, then they are polynomially computable on the whole class. Deciding whether a directed graph has a kernel or is kernel-perfect are NP-complete problems [1, 4]. Moreover, not so many classes of kernel-perfect graphs with polynomial algorithms for computing kernels are known.

Theorem 2. *Consider a class of directed graphs closed under taking induced subgraphs. Suppose that there is a polynomial algorithm that, for any induced subgraph of an atom, computes a kernel when it exists. Then there is a polynomial algorithm that, given any digraph D of the class, returns either a kernel of D, or an induced subgraph of D with no kernel.*

If every directed graph in the class is kernel-perfect, then the algorithm computes in polynomial time a kernel. However, if there are directed graphs in the class that are not kernel-perfect, the algorithm may fail to find a kernel, even if one exists, but it outputs then a certificate of non kernel-perfectness.

Orientations of chordal graphs form a graph class satisfying the condition of the theorem (see Section 2). It does not seem to have been known that a kernel can be found in polynomial time in kernel-perfect orientations of chordal graphs. We emphasize that, while the proof given by Jacob for his theorem is constructive and combines kernels of the pieces, we have not been able to adapt it to get a polynomial algorithm for chordal graphs or any graph class satisfying Theorem 2. Yet, there are families of interval graphs on which the straightforward application of the implicit algorithm in the latter proof is exponential.

2 Clique-acyclic orientations of perfect graphs

An arc (u, v) in a directed graph is *reversible* if the arc (v, u) exists, and *irreversible* otherwise. An *orientation* of an undirected graph G is a directed graph obtained by orienting each edge either in one direction or both ways. A *suborientation* is an orientation in which every edge in oriented in only one direction (there are no reversible arcs in a suborientation). An orientation is *clique-acyclic* if in every clique the subgraph of irreversible arcs is acyclic, or, equivalently, if every clique has a vertex absorbing all other vertices in the clique.

One of the main results on kernel-perfect graphs is a theorem by Boros and Gurvich [2], originally conjectured by Berge and Duchet in 1980, stating that every clique-acyclic orientation of a perfect graph has a kernel. An intriguing feature of their proof and of the subsequent proofs is that none of them provides an efficient method to compute a kernel. The complexity of this problem is an open question [9]. There are only few subclasses of perfect graphs for which the problem is known to be polynomial, e.g., bipartite graphs or line-graphs of bipartite graphs (via the Gale-Shapley algorithm for stable marriages [7, 13]). Theorem 2 allows to add to this list chordal and DE graphs. *DE graphs*, or "directed edge path graphs", introduced

by Monma and Wei [14], are defined as the intersection graphs of families of directed paths in directed trees (where two paths intersect if they share an arc). The atoms of chordal graphs (resp. DE graphs) are cliques (resp. line-graphs of bipartite graphs) and they satisfy thus the condition of Theorem 2.

Corollary 3. *The problem of computing a kernel in a clique-acyclic orientation of a chordal graph is polynomial.*

Corollary 4. *The problem of computing a kernel in a clique-acyclic orientation of a DE graph is polynomial.*

The complexity of kernel computation seems to be simpler when limited to suborientations. For instance, the polynomiality of finding a kernel is an easy exercise in the case of clique-acyclic suborientations of chordal graphs and was already known for clique-acyclic suborientations of DE graphs [5]. The following result also goes in that direction. A *circular-arc graph* is the intersection graph of intervals on a circle.

Proposition 5. *There is a polynomial algorithm that decides if a clique-acyclic suborientation of a circular-arc graph has a kernel and computes such a kernel when it exists.*

3 An application to hedonic games

A *hedonic game with graph structure* [10] is a triple $(N, (\succeq_i)_{i\in N}, L)$ where N is a finite set of *players*, each $\succeq_i$ is a complete and transitive *preference relation* over the subsets including i, and L is a set of pairs of players. A subset of players is a *feasible coalition* if they induce a connected subgraph of (N, L), seen as a graph. A partition π of N into feasible coalitions is *core stable* if, for every feasible coalition S, there exists a player $i \in S$ who weakly prefers his current coalition $\pi(i)$ to S, i.e., $\pi(i) \succeq_i S$.

One can associate core stable partitions with kernels as follows. Let $\mathcal{F}_L$ be the set of all feasible coalitions. Consider the directed graph $(\mathcal{F}_L, A)$ where $(S, T) \in A$ if there exists a player $i \in S \cap T$ with $T \succeq_i S$. The core stable partitions of a hedonic game are precisely the kernels of $(\mathcal{F}_L, A)$. Demange [6, Theorem 2, p.767] proved that finding a core stable partition when (N, L) is a tree can be done in polynomial time in the number of feasible coalitions. Since the intersection graphs of subtrees of a tree (where intersecting here means sharing a common vertex) are exactly the chordal graphs, Corollary 3 strengthens Demange's result to the more general hedonic game where $\mathcal{F}_L$ is restricted to some predetermined subfamily $\mathcal{F}$ of subsets of players.

4 Proof of Lemma 1

To ease the notation, let us define $X^{(i)} := C \cap \bigcup_{j=1}^{i-1} K^{(j)}$. The sequence $(X^{(i)})_{i=1,2,\dots}$ of subsets of C is nondecreasing (for inclusion). There is thus an index $k \in \{1, \dots, |C| + 1\}$ such that $X^{(k)} = X^{(k+1)}$. Since $K^{(k)}$ is a kernel of $D[B \cup (C \setminus X^{(k)})]$, we have $K^{(k)} \cap C \subseteq C \setminus X^{(k)}$. The equality $X^{(k)} = X^{(k+1)}$ implies that $K^{(k)} \cap C \subseteq X^{(k)}$. Hence, $K^{(k)} \cap C = \varnothing$.

Assume that $D[B' \cup X^{(|C|+1)}]$ has a kernel K. Suppose first that K and C have an empty intersection. Since C is a clique-cutset, the set $K \cup K^{(k)}$ is independent. The vertices in $B \cup (C \setminus X^{(k)})$ being absorbed by $K^{(k)}$ and those of $B' \cup X^{(k)}$ being absorbed by K, we get that $K \cup K^{(k)}$ is a kernel of D.

Suppose then that K and C have a nonempty intersection. Denote by v a vertex in $K \cap X^{(|C|+1)}$. By definition of $X^{(|C|+1)}$, there is an index $\ell \in \{1, \ldots, |C|+1\}$ such that $v \in K^{(\ell)}$. Since C is a clique-cutset, the set $K \cup K^{(\ell)}$ is independent in D. The vertices in $B \cup (C \setminus X^{(\ell)})$ being absorbed by $K^{(\ell)}$ and those of $B' \cup X^{(\ell)}$ by K, we get that $K \cup K^{(\ell)}$ is a kernel of D. □

References

[1] S.D. Andres and W. Hochstättler. Perfect digraphs. *Journal of Graph Theory*, 79:21–29, 2014.

[2] E. Boros and V. Gurvich. Perfect graphs are kernel solvable. *Discrete Mathematics*, 159:35–55, 1996.

[3] E. Boros and V. Gurvich. Perfects graphs, kernels, and cores of cooperative games. *Discrete Mathematics*, 306:2336–2354, 2006.

[4] V. Chvátal. On the computational complexity of finding a kernel. Technical Report CRM300, Centre de Recherches Mathématiques, Université de Montréal, 1973.

[5] O. Durand de Gevigney, F. Meunier, C. Popa, J. Reygner, and A. Romero. Solving coloring, minimum clique cover and kernel problems on arc intersection graphs of directed paths on a tree. *4OR*, 9:175–88, 2011.

[6] G. Demange. On group stability in hierarchies and networks. *Journal of Political Economy*, 112(4):754–778, 2004.

[7] D. Gale and L.S. Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, 69:9–15, 1962.

[8] F. Galvin. The list chromatic index of a bipartite multigraph. *Journal of Combinatorial Theory, Series B*, 63:153–158, 1995.

[9] Egerváry Research Group. Egres open, finding kernels in special digraphs. `http://lemon.cs.elte.hu/egres/open/Findingkernelsinspecialdigraphs`.

[10] A. Igarashi and E. Elkind. Hedonic games with graph-restricted communication. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS 2016, pages 242–250, 2016.

[11] H. Jacob. Kernels in graphs with a clique-cutset. *Discrete Mathematics*, 156:265–267, 1996.

[12] F. Maffray. On kernels in i-triangulated graphs. *Discrete Mathematics*, 61:247–251, 1986.

[13] F. Maffray. Kernels in perfect line-graphs. *Journal of Combinatorial Theory, Series B*, 55:1–8, 1992.

[14] C.L. Monma and V.K. Wei. Intersection graphs of paths in a tree. *Journal of Combinatorial Theory, Series B*, 41:141–181, 1986.

[15] O. Morgenstern and J. von Neumann. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

Submodular Secretary Problems: Cardinality, Matching, and Linear Constraints

Thomas Kesselheim*[1] and Andreas Tönnis †[2]

[1]Max-Planck-Institut für Informatik and Saarland University, Saarbrücken Informatics Campus, Germany., thomas.kesselheim@mpi-inf.mpg.de.
[2]Universität Bonn, Bonn, Germany., atoennis@uni-bonn.de.

1 Introduction

In the classic secretary problem, one is presented a sequence of items with different scores online in random order. Upon arrival of an item, one has to decide immediately and irrevocably whether to accept or to reject the current item. The objective is to accept the best of these items. Recently, combinatorial generalizations of this problem have attracted attention. In these settings, feasibility of solutions are stated in terms of matroid or linear constraints. In most cases, these combinatorial generalizations consider linear objective functions. This way, the profit gained by the decision in one step is independent of the other steps.

In this paper, we consider general monotone submodular functions[1]. For example, the *submodular secretary problem*, independently introduced by Bateni et al. [1] and Gupta et al. [3], is an online variant of monotone submodular maximization subject to cardinality constraints. In this problem, we are allowed to select up to k items from a set of n items. The value of a set is represented by a monotone, submodular function. Now, stated as an online problem, items arrive one after the other and every item can only be selected right at the moment when it arrives. The values of the submodular function are only known on subsets of the items that have already arrived. The objective function is designed by an adversary, but the order of the items is uniformly at random.

We call an algorithm c-competitive if for any objective function v chosen by the adversary, the set of selected items ALG satisfies $\mathbf{E}[v(\mathrm{ALG})] \geq (c-o(1)) \cdot v(\mathrm{OPT})$, where OPT is a size-$k$ subset of items that maximizes v.

Previous algorithms for submodular secretary problems were designed by modifying offline approximation algorithms for submodular objectives so that they could be used in the online environment. We take a different approach. Our algorithms are inspired by algorithms for

*Supported in part by the DFG through Cluster of Excellence MMCI.
†Work was done while this author was at RWTH Aachen University, supported by the DFG GRK/1298 "AlgoSyn".

[1]A function $f\colon 2^U \to \mathbb{R}$ for given ground set U is called *submodular* if for all $S \subseteq T \subseteq U$ and every $x \in U \backslash T$ holds $f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$.

linear objective functions. We repeatedly solve the respective offline optimization problem and use this outcome as a guide to make decisions in the current round. Generally, it is enough to only compute approximate solutions. Our results nicely separate the loss due to the online nature and due to limited computational power. Using polynomial-time computations and existing offline algorithms, we significantly outperform existing online algorithms. Certain submodular functions or kinds of constraints allow better approximations, which immediately transfer to even better competitive ratios. This is, for example, true for submodular maximization subject to a cardinality constraint if the number of allowed items is constant. Also, if computational complexity is no concern like in classical competitive analysis, our competitive ratios become even better.

2 Algorithmic Ideas and Results

Our new algorithm for the *submodular secretary problem* [4] follows the following natural paradigm. We reject the first $\frac{n}{e}$ items. Afterwards, for each arriving item, we run an α-approximation algorithm $\mathcal{A}$ for the offline optimization problem on the part of the instance that we have seen so far. If the current item is included in this solution and we have not yet accepted too many items, we accept it. Otherwise, we reject it. In our work, we treat the offline approximation as a black-box.

```
Drop the first ⌈pn⌉ − 1 items;
for item j arriving in round ℓ ≥ ⌈pn⌉ do        // online steps ℓ = ⌈pn⌉ to n
    Set U^{≤ℓ} := U^{≤ℓ−1} ∪ {j};
    Let S^(ℓ) = A(U^{≤ℓ});                      // black box α-approximation
    if j ∈ S^(ℓ) then                                // tentative allocation
        if |Accepted| < k then                           // feasibility test
            Add j to Accepted;                          // online allocation
        end
    end
end
```

Algorithm 1: Submodular secretary problem

Given an α-approximate algorithm for monotone submodular maximization subject to a cardinality constraint, we present an $\frac{\alpha}{e}\left(1-\frac{\sqrt{k-1}}{(k+1)\sqrt{2\pi}}\right)$-competitive algorithm for the submodular secretary problem. That is, we achieve a competitive ratio of at least 0.31α for any $k \geq 2$. Asymptotically for large k, we reach $\frac{\alpha}{e}$. Again, if computational complexity is no concern, then we can use the optimal offline solution in every iteration. In this case, we have $\alpha = 1$. In comparison, the previously best algorithm is $\frac{e-1}{e^2+e} \approx 0.170$-competitive [2].

For the analysis, we bound the expected value obtained by the algorithm recursively. It then remains to solve the recursion and to bound the resulting term. Generally, the recursive approach can be used for any secretary problems with cardinality constraints. It could be of independent interest, especially because it allows to obtain very good bounds also for rather small values of k.

One option for the black-box offline algorithm is the standard greedy algorithm by Nemhauser and Wolsey [6]. It always picks the item of maximum marginal increase until it has picked k items. Generally, this algorithm is $1-\frac{1}{e}$-approximate. However, it is known that if one

compares to the best solution with only $k \leq k$ items the approximation factor improves to $1 - \exp\left(-\frac{k}{k}\right)$. We exploit this fact to give a better analysis of our online algorithm when using the greedy algorithm in each step. We show that the algorithm is 0.238-competitive for any k and asymptotically for large k it is 0.275-competitive.

Additionally, we consider the *submodular secretary matching problem*. In this problem, one side of a bipartite graphs arrives online in random order. Upon arrival, vertices are either matched to a free vertex on the offline side or rejected. The objective is a submodular function on the set of matched pairs or edges. It is easy to see that the submodular secretary problem is a special case of this more general problem. Fortunately, similar algorithmic ideas work here as well. Again, we combine a sampling phase with a black box for the offline problem and get an 0.207α-competitive algorithm. Here the best previously known algorithm is $\frac{1}{95}$-competitive [5].

Finally, we show how our new analysis technique can be used to generalize previous results on linear packing programs towards submodular maximization with packing constraints. Here, we use a typical continuous extension towards the expectation on the submodular objective. We parameterize our results in d, the column sparsity of the constraint matrix, and B, the minimal capacity of the constraints. We achieve a competitive ratio of $\Omega(\alpha d^{-\frac{2}{B-1}})$ if both parameters are not known to the algorithm. If d and B are known beforehand we give different algorithm that is $\Omega(\alpha d^{-\frac{1}{B-1}})$-competitive. For both cases, the best previously known algorithm is $\Omega(\frac{1}{m})$-competitive [1].

References

[1] MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. *ACM Trans. Algorithms*, 9(4):32, 2013.

[2] Moran Feldman, Joseph Naor, and Roy Schwartz. Improved competitive ratios for submodular secretary problems (extended abstract). In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, pages 218–229, 2011.

[3] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Proc. 6th Intl. Conf. Web and Internet Economics (WINE)*, pages 246–257, 2010.

[4] Thomas Kesselheim and Andreas Tönnis. Submodular secretary problems: Cardinality, matching, and linear constraints. *CoRR*, abs/1607.08805, 2016.

[5] Tengyu Ma, Bo Tang, and Yajun Wang. The simulated greedy algorithm for several submodular matroid secretary problems. *Theoret. Comput. Sci.*, 58(4):681–706, 2016.

[6] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.*, 3(3):177–188, 1978.

Probabilistic Analysis of Facility Location on Random Shortest Path Metrics

Stefan Klootwijk[1] and Bodo Manthey[1]

[1]University of Twente, Enschede, The Netherlands

Abstract
The facility location problem is an $\mathcal{NP}$-hard optimization problem. Therefore, approximation algorithms are often used to solve large instances. Probabilistic analysis is a widely used tool to analyze such algorithms. Most research on probabilistic analysis of $\mathcal{NP}$-hard optimization problems involving metric spaces, such as the facility location problem, has been focused on Euclidean instances, and also instances with independent (random) edge lengths, which are non-metric, have been researched. However, we would like to extend this knowledge to other, more general, metrics.
We investigate the facility location problem using random shortest path metrics. We analyze some probabilistic properties for a simple heuristic which gives a solution to the facility location problem: opening a certain number of arbitrary facilities (with that certain number only depending on the facility opening cost). We show that, for almost any facility opening cost, this heuristic yields a $1+o(1)$ approximation in expectation. In the remaining few cases we show that this heuristic yields an $O(1)$ approximation in expectation.

Keywords: Facility location, Random shortest paths, Random metrics, Approximation algorithm

1 Introduction

The (uncapacitated) facility location problem can be described as follows: given a (complete) graph $G = (V, E)$, facility opening cost f_i for each vertex $v_i \in V$ and a distance $d(u, v)$ between each pair of vertices $u, v \in V$, find a subset $U \subseteq V$ of vertices at which you open facilities such that the total cost is minimized. Here, the total cost is given by the sum of the opening cost f_i for all vertices $v_i \in U$ and the sum of the 'connection' cost $\min_{u \in U} d(v, u)$ for all vertices $v \in V \backslash U$.

This problem is known to be $\mathcal{NP}$-hard [2]. Therefore research on the facility location problem (and other $\mathcal{NP}$-hard problems) has been focused on different heuristics, ranging from straightforward to rather sophisticated, and their worst-case performance (for instance [4, 7]).

So far, probabilistic analysis of heuristics for optimization problems like the facility location problem has been focused on instances either using Euclidean space or on (non-metric)

instances with independent (random) edge lengths, since such instances are technically relatively easy to handle. However, we would like to apply probabilistic analysis to more general metric instances. To do so, we use so-called 'random shortest path metrics', which have also been used by Bringmann et al. [1], who initiated this research.

Random shortest path metrics are defined as follows. Consider an undirected complete graph $G = (V, E)$ on n vertices. For any edge $e \in E$, let $w(e) \sim \text{Exp}(1)$ be the weight of edge e, independently drawn from the standard exponential distribution. Then, the distances $d(u, v)$ between each pair of vertices $u, v \in V$ are defined as the minimum total weight of any u, v-path $\mathcal{P}$ in G. The underlying model of random shortest path metrics is also known as first-passage percolation.

Many structural properties of random shortest path metrics are known, such as the expected shortest path length ($\ln(n)/n$ in expectation as $n \to \infty$) [1, 3, 6], and the number of edges on the shortest path between any two vertices [5].

We consider instances of the facility location problem for which the distances d are randomly generated using the principle of random shortest path metrics and for which every vertex has the same nonnegative facility opening cost f, i.e. $f_i = f \geq 0$ for all $i \in V$. This implies that the total cost of any solution $\varnothing \neq U \subseteq V$ is given by

$$\mathsf{cost}(U) = f \cdot |U| + \sum_{v \in V \setminus U} \min_{u \in U} d(v, u).$$

Although we do not mention it explicitly, the facility opening cost f does depend on the size of the instance, i.e., we have $f = f(n)$. It makes sense to do this, since the expected distance between two arbitrary vertices also depends on n when using random shortest path metrics.

We show that the most trivial procedure of opening a fixed number of arbitrary facilities (with that fixed number only depending on the facility opening cost f) yields a $1 + o(1)$ approximation in expectation unless $f \in \Theta(1/n)$. If $f \in \Theta(1/n)$, then this procedure is shown to yield a $O(1)$ approximation in expectation.

2 An intuitive approach

Intuitively, we observe that the optimal solution for our problem will satisfy $|U| \approx n$ when the facility opening cost f is (almost) 0. On the other hand, the optimal solution will satisfy $|U| = 1$ when the facility opening cost is relatively large. And when the facility opening cost f are neither 'relatively small' nor 'relatively large', $|U|$ will be neither close to n nor close to 1 for the optimal solution.

Let OPT denote the total cost of the optimal solution to the facility location problem. Furthermore, for $k \in [n] := \{1, \ldots, n\}$, let OPT_k denote the total cost of the optimal solution to the corresponding k-median problem. Then it follows that

$$\mathsf{OPT} = \min_{\varnothing \neq U \subseteq V} \mathsf{cost}(U) = \min_{\varnothing \neq U \subseteq V} \left(f \cdot |U| + \sum_{v \in V \setminus U} \min_{u \in U} d(v, u) \right) = \min_{k \in [n]} \left(f \cdot k + \mathsf{OPT}_k \right).$$

Moreover, based on the results of Bringmann et al. [1, Sect. 5] we know that $U_k := \{v_1, \ldots, v_k\}$ is a good approximation for the optimal solution to the k-median problem (whenever k is not too large), and that the expected cost of this solution is given by $\mathbb{E}[\mathsf{cost}_k(U_k)] = \ln(n/k) + \Theta(1)$.

So, intuitively we see that

$$\mathbb{E}[\mathsf{OPT}] = \mathbb{E}\left[\min_{k\in[n]}\left(f\cdot k + \mathsf{OPT}_k\right)\right] \approx \min_{k\in[n]}\left(f\cdot k + \mathbb{E}[\mathsf{OPT}_k]\right)$$
$$\approx \min_{k\in[n]}\left(f\cdot k + \mathbb{E}[\mathsf{cost}_k(U_k)]\right) = \min_{k\in[n]}\left(f\cdot k + \ln(n/k) + \Theta(1)\right).$$

Finally, we observe that the function $g(k) = f\cdot k + \ln(n/k)$ is minimal for $k = 1/f$, resulting in $\mathbb{E}[\mathsf{OPT}] \approx 1 + \ln(nf) + \Theta(1)$. Combining this observation with the foregoing intuitive arguments, it seems likely that any arbitrary solution U to the facility location problem with $|U| \approx 1/f$ yields a good approximation for OPT.

3 Main results

We are interested in the expected approximation ratio of an algorithm that opens approximately $1/f$ arbitrary facilities. In order to analyze this, we use the rather trivial algorithm which opens exactly $k := \min\{\lceil 1/f\rceil, n\}$ randomly chosen facilities. Let TRIV denote the total cost of the solution computed by this algorithm, and let OPT denote the total cost of an optimal solution to the facility location problem. Then, using a result from Bringmann et al. [1, Sect. 5], we can derive the probability distribution of TRIV.

Lemma 1. *If $k = n$, then TRIV has a degenerate probability distribution with $\mathbb{P}(\mathsf{TRIV} = nf) = 1$. Otherwise, the distribution of TRIV is given by*

$$\mathsf{TRIV} \sim k\cdot f + \sum_{i=k}^{n-1}\mathrm{Exp}(i),$$

where the $\mathrm{Exp}(i)$ are independent exponentially distributed random variables with parameter i.

Observe that we can use our intuitive approach to show that $\mathbb{E}[\mathsf{TRIV}] \approx \mathbb{E}[\mathsf{OPT}]$. However, using a more thorough analysis, in which we combine this probability distribution with some bounds for OPT, we can show that TRIV yields either a constant or an asymptotically optimal approximation ratio. This is summarized in the following theorem.

Theorem 2. *Let OPT denote the total cost of the optimal solution to the facility location problem, and let TRIV denote the total cost of the solution which opens exactly $\min\{\lceil 1/f\rceil, n\}$ randomly chosen facilities. Then, it follows that*

$$\mathbb{E}\left[\frac{\mathsf{TRIV}}{\mathsf{OPT}}\right] = O(1).$$

Moreover, if either $f \in o(1/n)$ or $f \in \omega(1/n)$, then it follows that

$$\mathbb{E}\left[\frac{\mathsf{TRIV}}{\mathsf{OPT}}\right] = 1 + o(1).$$

In order to prove this theorem, we divide the range of possible (asymptotic) facility opening costs f in three (slightly overlapping) intervals, corresponding to the three intuitive cases mentioned above: opening all facilities ($f \le (2-\varepsilon)/n$), opening exactly one arbitrary facility ($f \ge 1/n^{\varepsilon}$), and opening some arbitrary facilities ($(1+\varepsilon)/n \le f \le M/n^{\varepsilon}$). For each case we have found a threshold such that conditioning the expected approximation ratio on the events

OPT is larger relatively smaller than this threshold, allows us to prove the bounds mentioned in Theorem 2.

These thresholds are chosen in such a way that for the case in which OPT is larger than the threshold, it is relatively easy to bound the conditional expected approximation ratio. On the other hand, the thresholds are also chosen in such a way that the probability of OPT being smaller than the threshold becomes sufficiently small. By doing so, we are able to show that the (relatively) large conditional expected approximation ratio in this case becomes negligible when multiplied with that probability.

4 Final remarks

As far as we are aware, these results form only a second step into the research of the behavior of (combinatorial) optimization problems using random shortest path metrics (the first step being the results in by Bringmann et al. [1]). Even though random shortest path instances are more difficult to analyze than Euclidean instances or instances with independent random edge lengths, we were able to derive some good results when analyzing the facility location problem on it.

It would be interesting to see whether it is possible to prove similar results when using more sophisticated heuristics that aim to solve the facility location problem. Furthermore, there are many other $\mathcal{NP}$-hard (combinatorial) optimization problems involving metric spaces for which it would be interesting to know how they behave on random shortest path metrics.

References

[1] K. Bringmann, C. Engels, B. Manthey, and B.V.R. Rao. Random shortest paths: Non-euclidean instances for metric optimization problems. *Algorithmica*, 73(1):42–62, 2015. doi: 10.1007/s00453-014-9901-9.

[2] G. Cornuejols, G.L. Nemhauser, and L.A. Wolsey. The uncapacitated facility location problem. In Pitu B. Mirchandani and Richard L. Francis, editors, *Discrete Location Theory*, chapter 3, pages 119–171. Wiley-Interscience, New York, 1990. ISBN 978-0-471-89233-5.

[3] R. Davis and A. Prieditis. The expected length of a shortest path. *Information Processing Letters*, 46(3):135–141, 1993. doi: 10.1016/0020-0190(93)90059-I.

[4] A.D. Flaxman, A.M. Frieze, and J.C. Vera. On the average case performance of some greedy approximation algorithms for the uncapacitated facility location problem. *Combinatorics, Probability and Computing*, 16(5):713–732, 2007. doi: 10.1017/S096354830600798X.

[5] R. van der Hofstad, G. Hooghiemstra, and P. Van Mieghem. First-passage percolation on the random graph. *Probability in the Engineering and Informational Science*, 15(2): 225–237, 2001. doi: 10.1017/S026996480115206X.

[6] S. Janson. One, two and three times log n/n for paths in a complete graph with random weights. *Combinatorics, Probability and Computing*, 8(4):347–361, 1999. doi: 10.1017/S0963548399003892.

[7] S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013. doi: 10.1016/j.ic.2012.01.007.

[8] J. Vygen. Approximation algorithms for facility location problems (lecture notes). Technical Report No. 05950, Research Institute for Discrete Mathematics, University of Bonn, 2005. URL `http://www.or.uni-bonn.de/~vygen/files/fl.pdf`.

Determining the Optimal Pure Strategies for Average Markov Decision Problem

Dmitrii Lozovanu[1] and Stefan Pickl[2]

[1]Institute of Mathematics and Computer Science of Academy of Sciences of Moldova;
e-mail:lozovanu@math.md
[2]Institute for Theoretical Computer Science, Mathematics and Operations Research, Universität der Bundeswehr, Germany; e-mail:stefan.pickl@unibw.de

ABSTRACT: The infinite horizon average Markov decision problem with finite state and action spaces is considered and an approach for determining the optimal pure stationary strategies is proposed.

1 Introduction and problem formulation

We consider the average Markov decision problem determined by:
- a finite set of states X;
- a finite set of actions $A(x)$ for each state $x \in X$;
- 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 step reward $r_{x,a}$ for each state $x \in X$ and every action $a \in A(x)$;
- a starting state $x_0 \in X$.

The infinite horizon average Markov decision problem consists in determining a stationary policy of choosing the actions in the states for which the average reward per transition in the corresponding Markov process is maximal. It is well-known [4] that an optimal stationary policy for such a problem can be found by using the following linear programming model:
Maximize

$$\psi(\alpha, \beta) = \sum_{x \in X} \sum_{a \in A(x)} r_{x,a} \alpha_{x,a} \tag{1}$$

subject to

$$\begin{cases} \sum\limits_{a \in A(y)} \alpha_{y,a} - \sum\limits_{x \in X} \sum\limits_{a \in A(x)} p^a_{x,y} \alpha_{x,a} = 0, \ \ \forall y \in X; \\ \sum\limits_{a \in A(y)} \alpha_{y,a} + \sum\limits_{a \in A(y)} \beta_{y,a} - \sum\limits_{x \in X} \sum\limits_{a \in A(x)} p^a_{x,y} \beta_{x,a} = \theta_y, \ \forall y \in X; \\ \alpha_{x,a} \geq 0, \quad \beta_{y,a} \geq 0, \ \ \forall x \in X, \ a \in A(x), \end{cases} \tag{2}$$

where θ_y for $y \in X$ represent arbitrary positive values that satisfy the condition $\sum\limits_{y\in X} \theta_y = 1$, where θ_y for $y \in Y$ are treated as the probabilities of choosing the starting state $y \in X$. In the case $\theta_y = 1$ for $y = x_0$ and $\theta_y = 0$ for $y \in X \setminus \{x_0\}$ we obtain the linear programming model for an average Markov decision problem with fixed starting state x_0.

This linear programming model corresponds to the multichain case of an average Markov decision problem. If each stationary policy in the decision problem induces an ergodic Markov chain then the restrictions (2) can be replaced by the restrictions

$$\begin{cases} \sum\limits_{a\in A(y)} \alpha_{y,a} - \sum\limits_{x\in X} \sum\limits_{a\in A(x)} p^a_{x,y}\, \alpha_{x,a} = 0, \quad \forall y \in X; \\ \sum\limits_{y\in X} \sum\limits_{a\in A(y)} \alpha_{y,a} = 1; \\ \alpha_{y,a} \geq 0, \quad \forall y \in X, \quad a \in A(y). \end{cases} \tag{3}$$

In the linear programming model (1),(2) the restrictions

$$\sum_{a\in A(y)} \alpha_{y,a} + \sum_{a\in A(y)} \beta_{y,a} - \sum_{x\in X} \sum_{a\in A(x)} p^a_{x,y}\beta_{x,a} = \theta_y, \ \forall y \in X$$

with the condition $\sum\limits_{y\in X} \theta_y = 1$ generalize the constraint

$$\sum_{x\in X} \sum_{a\in A(y)} \alpha_{y,a} = 1$$

in the linear programming model (1),(3) for the ergodic case.

The relationship between feasible solutions of problem (1),(2) and stationary strategies in the average Markov decision problem is the following:

Let (α, β) be a feasible solution of the linear programming problem (1), (2) and denote by $X_\alpha = \{x \in X | \sum\limits_{a\in X} \alpha_{x,a} > 0\}$. Then (α, β) possesses the properties that $\sum\limits_{a\in A(x)} \beta_{x,a} > 0$ for $x \in X \setminus X_\alpha$ and a stationary strategy $s_{x,a}$ that correspond to (α, β) is determined as

$$s_{x,a} = \begin{cases} \dfrac{\alpha_{x,a}}{\sum\limits_{a\in A(x)} \alpha_{x,a}} & \text{if } \ x \in X_\alpha; \\ \dfrac{\beta_{x,a}}{\sum\limits_{a\in A(x)} \beta_{x,a}} & \text{if } \ x \in X \setminus X_\alpha, \end{cases} \tag{4}$$

where $s_{x,a}$ expresses the probability of choosing the actions $a \in A(x)$ in the states $x \in X$.

Thus, *a stationary strategy* in a Markov decision problem is a mapping s that for every state $x \in X$ provides a probability distribution over the set of actions $A(x)$; if these probabilities take only values 0 and 1, then s is called *pure stationary strategy*, otherwise s is called *mixed stationary strategy.*

It is well known [1, 4] that for an arbitrary average Markov decision problem it always has an optimal solution that corresponds to a pure stationary strategy. However, as it is noted in [4], the linear programming problem (1),(2) may have a basic solution (α, β) for which the

corresponding stationary strategy s determined through (4) is not a pure stationary strategy for the Markov decision problem. So, if we solve the linear programming problem (1),(2) and find an optimal basic solution (α^*, β^*) then the corresponding optimal stationary strategy s^* determined according to (4) may be not a pure strategy.

In this contribution we formulate a new optimization model in terms of stationary strategies for the average Markov decision problem that allows to determine all its optimal pure stationary strategies. The proposed model is related to quasi-linear programming in which it is necessary to maximize a quasi-linear objective function on a convex polyhedron set.

2 The main results

First we show that an average Markov decision problem in terms of stationary strategies can be formulated as follows:
Maximize

$$\psi(s,q,w) = \sum_{x\in X} \sum_{a\in A(x)} f(x,a) s_{x,a} q_x \tag{5}$$

subject to

$$\begin{cases} q_y - \sum\limits_{x\in X} \sum\limits_{a\in A(x)} p^a_{x,y}\, s_{x,a} q_x = 0, & \forall y \in X; \\ q_y + w_y - \sum\limits_{x\in X} \sum\limits_{a\in A(x)} p^a_{x,y} s_{x,a} w_x = \theta_y, & \forall y \in X; \\ \sum\limits_{a\in A(y)} s_{y,a} = 1, & \forall y \in X; \\ s_{x,a} \geq 0, \ \ \forall x \in X, \ \forall a \in A(x); \quad w_x \geq 0, \ \forall x \in X, \end{cases} \tag{6}$$

where θ_y are the same values as in problem (1), (2) and $s_{x,a}$, q_x, w_x for $x \in X$, $a \in A(x)$ represent the variables that must be found, where q_x for $x \in X$ express the limiting probabilities in the states for a given strategy s when the starting sate is chosen with probability θ_y.

We proved that the optimization problem (5), (6) determines all optimal stationary strategies for the multichain average Markov decision problem with finite f state and action spaces.

The main result of the paper that allows to ground algorithms for determining the optimal pure stationary strategies for the average Markov decision problem is represented by the following theorem.

Theorem 1. *Let an average Markov decision problem be given and consider the function*

$$\psi(s) = \sum_{x\in X} \sum_{a\in A(x)} f_{(x,a)} s_{x,a}\, q_x, \tag{7}$$

where q_x for $x \in X$ satisfy the condition

$$\begin{cases} q_y - \sum\limits_{x\in X} \sum\limits_{a\in A(x)} p^a_{x,y}\, s_{x,a} q_x = 0, & \forall y \in X; \\ q_y + w_y - \sum\limits_{x\in X} \sum\limits_{a\in A(x)} p^a_{x,y} s_{x,a} w_x = \theta_y, & \forall y \in X. \end{cases} \tag{8}$$

Then on the set S of solutions of the system

$$\begin{cases} \sum\limits_{a\in A(x)} s_{x,a} = 1, \quad \forall x \in X; \\ \qquad s_{x,a} \geq 0, \quad \forall x \in X,\ a \in A(x) \end{cases} \tag{9}$$

the function $\psi(s)$ depends only on $s_{x,a}$ for $x \in X,\ a \in A(x)$ and $\psi(s)$ is quasi-linear on S (i.e. $\psi(s)$ is quasi-convex and quasi-concave on S).

Remark 2. *The function (7) on S depends only on $s_{x,a}$ for $x \in X,\ a \in A(x)$ because system (8) uniquely determines $q_x, \forall x \in X$ for a given $s \in S$.*

Thus, based on Theorem 1, we can determine an optimal pure stationary strategy using classical gradient descent methods for the maximization of a quasi-linear function (7) on a convex polyhedron set S. It is easy to observe that the convergence of some of the algorithms for determining the optimal stationary strategies can be grounded using the proposed optimization models and Theorem 1. Additionally the proposed models can be useful for the formulation of the average stochastic positional games in terms of pure stationary strategies [3].

3 Conclusion

An average Markov decision problem with finite state and action spaces can be formulated and studied in terms of stationary strategies using optimization models (5),(6) and (7)-(9). Classical optimization methods and the corresponding algorithms for the maximization of a quasi-linear function (7),(8) on the convex polyhedron set determined by (9) can be applied for finding the optimal pure stationary strategies in the average Markov decision problem.

References

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

[2] Lozovanu D., Pickl S. *Determining the optimal strategies for discrete control problems on stochastic networks with discounted costs.* Discrete Applied Mathematics, 182, p. 169-180, 2015.

[3] Lozovanu D., Pickl S. *On Nash equilibria for stochastic games and determining the optimal strategies of the players.* Contribution to game theory and management, St. Petersburg University, V VIII, 187–198, 2015.

[4] Puterman M. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley, New Jersey 2005.

Probabilistic Properties of Highly Connected Random Geometric Graphs

Bodo Manthey[1] and Victor M.J.J. Reijnders[1]

[1]University of Twente, Department of Applied Mathematics, Enschede, The Netherlands

In this paper we study the probabilistic properties of reliable networks of minimal total edge lengths. We study reliability in terms of k-edge-connectivity in graphs in d-dimensional space. We show this problem fits into Yukich's framework for Euclidean functionals for arbitrary k, dimension d and distant-power gradient p, with $p < d$. With this framework several theorems on the convergence of optimal solutions follow. We apply Yukich's framework for functionals so that we can use partitioning algorithms that rapidly compute near-optimal solutions on typical examples. These results are then extended to optimal k-edge-connected power assignment graphs, where we assign power to vertices and charge per vertex. The network can be modelled as a wireless network.

1 Introduction

The design of fault tolerant networks is an important issue in today's research, due to their numerous applications [1]. The goal is to find cheap and reliable networks with some specific characteristics. Reliability is often expressed in terms of the connectivity of a network. For example, we might want to have multiple paths between each pair of nodes to account for possible failures in a link.

Wireless ad hoc networks have also received significant attention in recent studies [4, 6]. Instead of direct connections between nodes, communication takes place through single-hop transmissions or by relaying through intermediate nodes. Here we assign a transmission power to each node. As transmission range is directly related to power usage and therefore to battery lifetime, the goal is to find a fault tolerant network with minimal total power usage.

Finding a cheapest k-edge-connected network is NP-hard [5], and so is finding a minimal power wireless network [3]. As we still want to have reasonably good solutions in acceptable computation time, we need to find good heuristics. We fit the problems into Yukich's framework for Euclidean functional [8] to get limit theorems and concentration results, as well as using them for analysis of the partitioning algorithm.

Partitioning algorithms have shown a lot of potential with similar problems [2]. In practice, partitioning algorithms are very fast. Partitioning algorithms divide the whole problems into smaller cells and compute optimal solutions on these. Then these solutions are joined to obtain a solution for the whole problem.

2 Definitions and Results

All graphs in this paper are undirected and simple. Let $G = (V, E)$ be a graph. We assume $V \subset \mathbb{R}^d$, where d is a constant and V is finite. The cost of an edge is its length raised to the power of the distant-power gradient $p > 0$. So adding edge (u, v) to a graphs increases the cost by $|(u, v)|^p$, where $|(u, v)|$ denotes the Euclidean distance between u and v. Here, we assume p is a constant.

A graph is k-edge-connected if the graph is still connected when at most $k - 1$ edges are removed, or if it is complete. The latter is to make sure k-edge-connected graphs on less than $k + 1$ nodes still exist, which saves us from dealing with all kind of exceptions in proofs. Alternatively, a network is k-edge-connected if there exist at least k edge-disjoint paths between every pair of vertices.

Let $d \in \mathbb{N}$ be arbitrary and let $p > 0$. Then $\mathrm{MkEE}^p(V)$ is the minimal length of a k-edge-connected graph in terms of summed edge lengths on V with pth power-weighted edges. Thus

$$\mathrm{MkEE}^p(V) = \min_{X \in \mathcal{S}(V,k)} \sum_{e \in X} |e|^p, \tag{1}$$

where $\mathcal{S}(V, k)$ is the set of k-edge-connected simple graphs on V and $|e|$ denotes the Euclidean length of an edge e. Following Yukich [8], we call MkEE^p a functional.

One of the desired properties for functionals is subadditivity. Roughly speaking, this shows that the function value of a whole set is not larger than the sum of function values of the sets in a partition of this set (with some error term).

Theorem 1. *For $p \geq 1$, MkEE^p is geometrically subadditive, i.e. for all finite sets V, all rectangles R and all partitions of R into rectangles R_1 and R_2 we have*

$$\mathrm{MkEE}^p(V \cap R) \leq \mathrm{MkEE}^p(V \cap R_1) + \mathrm{MkEE}^p(V \cap R_2) + C_1(\operatorname{diam} R)^p, \tag{2}$$

where $C_1 = C_1(d, p)$ is a constant.

We would also want MkEE^p to be superadditive. Roughly speaking, this would show that the function value of a whole set is not lower than the sum of function values of the sets in a partition. Combining sub- and superadditivity makes the functional nearly additive in the sense that $\mathrm{MkEE}^p(F, R) \approx \mathrm{MkEE}^p(F, R_1) + \mathrm{MkEE}^p(F, R_2)$.

We could then approximate the optimal solution value of the whole set by the sum of optimal solutions on its partitions. It is easily checked however that MkEE^p does not possess superadditivity. This is why we introduce the canonical boundary functional, an idea first articulated in Redmond's thesis [7]. In boundary functionals, the entire boundary of the rectangle is considered as one additional vertex that can be used. We also refer to Yukich [8] for more on this topic.

MkEE^p_B is the boundary functional of MkEE^p, so that $\mathrm{MkEE}^p_B(V \cap R)$ is the minimal length of a k-edge-connected boundary graph in terms of summed edge lengths on $V \cup \partial R$ in d-dimensional rectangle R with pth power-weighted edges. Here ∂R denotes the boundary of R. A vertex v is connected to ∂R by adding edge (v, v_∂) where $v_\partial = \arg\min_{w \in \partial R} |(v, w)|$.

Theorem 2. *For $p \geq 1$, MkEE^p_B is a superadditive functional, i.e. for all finite sets V, all rectangles R and all partitions of R into rectangles R_1 and R_2 we have*

$$\mathrm{MkEE}^p_B(V \cap R) \geq \mathrm{MkEE}^p_B(V \cap R_1) + \mathrm{MkEE}^p_B(V \cap R_2). \tag{3}$$

As we cannot directly show near additivity, we want to show that MkEE^p and MkEE^p_B are pointwise close. Then we would get approximately get sub- and superadditivity for both functionals.

Theorem 3. *For $1 \leq p < d$, MkEE^p is pointwise close to MkEE^p_B, i.e. for all finite sets $V \subset [0,1]^d$ we have*

$$|\,\mathrm{MkEE}^p(V) - \mathrm{MkEE}^p_B(V)| = o(|V|^{(d-p)/d}). \tag{4}$$

We have shown geometric subadditivity, superadditivity and pointwise closeness, creating a powerful set of properties. These properties are more useful for obtaining other results when the functional also is smooth. This describes how strong the variations of a functional are if vertices are added or deleted. Smooth functionals behave a lot more predictable and therefore it plays an important role in many limit theories.

Theorem 4. *For $1 \leq p < d$, MkEE^p is smooth, i.e. for all finite sets U and V we have*

$$|\,\mathrm{MkEE}^p(U \cup V) - \mathrm{MkEE}^p(U)| = O(|V|^{(d-p)/d}). \tag{5}$$

One of the concentration results we have obtained is stated below. It shows that the functional values are not far from their expected value.

Theorem 5. *For $1 \leq p < d$ and $k \in \mathbb{N}$, there exists a constant $\alpha = \alpha(d,k) \geq 0$ such that*

$$\lim_{n\to\infty} \mathrm{MkEE}^p(V,R)/n^{(d-p)/p} = \alpha \quad \textit{c.c., and} \tag{6}$$

$$\lim_{n\to\infty} \mathrm{MkEE}^p_B(V,R)/n^{(d-p)/p} = \alpha \quad \textit{c.c.,} \tag{7}$$

where $n = |V|$. Here c.c. denotes complete convergence.

3 Partitioning algorithm

In a partitioning algorithm, the Euclidean plane is divided into a number of cells that all contain only a few points. On each cell an optimal solution is calculated. This is generally much faster than calculating a solution on all points at once, as these problems are often NP-hard. The solutions of all cells are then joined to obtain a solution for the whole set.

We implement a partitioning scheme for MkEE^p having a polynomial running time, for which we derive approximation guarantees.

Algorithm 6 (Partitioning Scheme)**.**
Input: *set $V \subseteq [0,1]^d$ of n points and number of points per cell s*

1. *Partition $[0,1]^d$ into $\ell = \sqrt[d]{n/s}$ stripes of dimension $d-1$ such that each stripe contains exactly $n/\ell = (n^{d-1}s)^{1/d}$ points.*
2. *Keep partitioning each $i+1$-dimensional stripe into ℓ stripes of dimension i such that each stripe contains exactly $n/\ell^i = (n^{d-i}s^i)^{1/d}$ points. Stop at $i = 1$ so that each 2-dimensional stripe is partitioned into ℓ cells with $n/\ell^d = s$ points. In this way we end up with $\ell^d = n/s$ cells. Here we assume $s > k$.*
3. *Compute a graph achieving the optimal solution of MkEE^p for each cell.*
4. *Join the graphs to obtain a k-edge-connected graph on V.*

It can be easily verified that the graph we get as an output from Algorithm 6 is k-edge-connected. With this algorithm and the properties obtained in Section 2 we can now give running time and approximation guarantees. Depending on the way we compute the optimal solution on each cell, we need to vary s to get a polynomial running-time.

Theorem 7. *If the algorithm for computing an optimal solution on each cell in Algorithm 6 has a running time of $O(C^{n^2})$ for some constant C, the Partitioning Scheme has a polynomial running time if we choose $s = O(\sqrt{\log n})$. The approximation guarantee then becomes $\mathrm{MkEE}^p(V) + O((n/s)^{(d-p)/d})$ for k-edge-connected graphs.*

4 Extention to wireless networks

Besides our model for wired networks, we consider a different model for wireless networks. These are defined by assigning power to each vertex. A power assignment PA assigns a real, positive value to all vertices $v \in V$. The corresponding power assignment graph then contains all edges (u, v) for which $\mathrm{PA}(u), \mathrm{PA}(v) \geq |(u, v)|^p$. The costs of k-edge-connected power assignment graphs is then simply the sum of all assigned powers. We obtain results similar to Theorems 1 – 5 for the functional in wireless networks.

References

[1] Fatiha Bendali, I Diarrassouba, Ali Ridha Mahjoub, M Didi Biha, and Jean Mailfert, *A branch-and-cut algorithm for the k-edge connected subgraph problem*, Networks **55** (2010), no. 1, 13–32.

[2] Markus Bläser, Bodo Manthey, and BV Raghavendra Rao, *Smoothed analysis of partitioning algorithms for euclidean functionals*, Algorithmica **66** (2013), no. 2, 397–418.

[3] Andrea EF Clementi, Paolo Penna, and Riccardo Silvestri, *On the power assignment problem in radio networks*, Mobile Networks and Applications **9** (2004), no. 2, 125–140.

[4] Maurits de Graaf and Bodo Manthey, *Probabilistic analysis of power assignments*, International Symposium on Mathematical Foundations of Computer Science, Springer, 2014, pp. 201–212.

[5] Michael R Gary and David S Johnson, *Computers and intractability: A guide to the theory of np-completeness*, 1979.

[6] Ram Ramanathan and Regina Rosales-Hain, *Topology control of multihop wireless networks using transmit power adjustment*, INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 2, IEEE, 2000, pp. 404–413.

[7] Charles Redmond, *Boundary rooted graphs and euclidean matching algorithms*, Ph.D. thesis, Lehigh University, Bethlehem, PA, USA, 1993.

[8] Joseph E Yukich, *Probability theory of classical euclidean optimization problems*, Lecture Notes in Mathematics, vol. 1675, Springer-Verlag, Berlin, 1998.

Radio connectivity of graphs

Ruxandra Marinescu-Ghemeci[1]

[1]Faculty of Mathematics and Computer Science, University of Bucharest, Romania,
The Research Institute of the University of Bucharest ICUB, Romania,
verman@fmi.unibuc.ro

Given a graph G and a vertex coloring c, G is called l-radio connected if between any two distinct vertices u and v there is a path such that coloring c restricted to that path is a l-radio coloring. The smallest number of colors needed to make G l-radio connected is called the l-radio connection number of G. In this paper we introduce these notions and initiate the study of radio connectivity, providing results on the 2-radio connection number, also called $L(2,1)$-connection number: upper and lower bounds, exact values for known classes of graphs and graph operations.

1 Introduction

Various types of graph colorings have been introduced in literature motivated by problems in communication networks. An important property in such networks is connectivity, that is to have paths between each pair of vertices. Many times it is not sufficient to have arbitrary paths, but paths that assure a safe communication. For example, if interference may occur, it is necessary to have paths along which interferences are avoided. This might mean, for example, that the labels associated to the edges or vertices of the path should be distinct, as in the case of rainbow connectivity. Related to this, different types of constrains were imposed to the colors of the edges of vertices in a path. For example, if only adjacent edges in a path are required to have distinct colors, such a path is called *proper path*. The notion of *proper connectivity*, similar to rainbow connectivity but through proper paths instead of rainbow paths, was introduced in [1]. Also, a more general case was considered in [4], where edges at distance at most l in a path are required to have distinct colors. Similar problems were studied for vertex-colorings.

Yet, there are situations when, in order to have no interference, it is necessary that the difference between labels of close edges or vertices - close meaning at distance less than a fixed level l, to be greater than a certain limit. To model this type of requests radio colorings were introduced. First, Hale [3] considered only two levels of interference and defined a $L(2,1)$-labeling of a graph. This type of labeling was later generalized to more levels of interferences - $L(d_1, d_2, ..., d_l)$-labelings, among which the most known are radio colorings, where $d_i = l+1-i$. For a fixed level l, a *l-radio coloring* is a function $c : V(G) \to \mathbb{N}^*$ satisfying the following property (called *radio condition*): $|c(u) - c(v)| \geq l + 1 - d(u,v), \forall u \neq v \in V(G)$. Assuming that the smallest color used is 1, the maximum color used by c is called the span of c and is denoted $span(c)$. The minimum span of a radio coloring is the *l-radio number* of G. For $l = 2$, c is actually a $L(2,1)$-labeling and 2-radio number is called $L(2,1)$-number and denoted $\lambda(G)$.

But, in order to solve interference or security problems sometimes is not necessary to color all vertices of the graph such that every pair of vertices satisfy the radio condition, but to assure that between every pair of vertices there is at least one path such that the coloring restricted to that path is a radio coloring, as in the case of proper connectivity. Motivated by this, the aim of this paper is to introduce the notion of l-radio connectivity for a vertex-colored graph and present results for the case when $l = 2$ regarding upper and lower bounds, exact values for some classes of graphs and graph operations, existence problems.

Let G be a connected graph and $c : V(G) \to \mathbb{N}$ a coloring of G having the smallest value 1 and the maximum value $span(c) = k$. Consider l a number representing the number of levels of interference. A path P in G is called *l-radio path* if coloring c restricted to $V(P)$ is a l-radio coloring for P. The coloring c is called *l-radio path coloring* if there exists a l-radio path between every pair of distinct vertices of G. A graph is *l-radio connected* if it admits a l-radio path coloring. The minimum span of a l-radio path coloring of G is called *l-radio connection number* of G and is denoted $rcc^l(G)$. A l-radio path coloring with span equal to $rcc^l(G)$ is called a *minimum l-radio path coloring*. For $l = 2$, since a 2-radio coloring is called $L(2,1)$-coloring, we will use the notions of *$L(2,1)$-path coloring*, *$L(2,1)$-paths*, *$L(2,1)$-connected graph*. Denote $\lambda c(G) = rcc^2(G)$ and refer to it as *$L(2,1)$-connection number* of G.

For basic notions and notations we refer to [5]. Denote by $[n] = \{1, 2, \ldots, n\}$.

Next we will state results on $L(2,1)$-connection number of a graph. We will study 2-(edge) connected graphs, since robust networks present interest for interference problems. $L(2,1)$-connection number of a graph is often much smaller than its $L(2,1)$-number, thus $L(2,1)$-path coloring might be more useful in practice than $L(2,1)$-labelings for networks with interferences.

2 $L(2,1)$-connection number of a graph

Lemma 1. *Let G be a connected graph with $n \geq 2$ vertices.*

1. *If G is a tree then $\lambda c(G) = \lambda(G)$*
2. *$\lambda(P_{diam(G)+1}) \leq \lambda c(G) \leq \lambda(G)$*
3. *If H is a spanning connected graph of G, then $\lambda c(G) \leq \lambda c(H)$.*

Proposition 2. *Let G be a connected graph with $n \geq 2$ vertices.*

1. *$\lambda c(G) = 3$ if and only if $G = P_2$*
2. *$\lambda c(G) = 4$ if and only if $3 \leq n \leq 4$ and $G \neq S_3$*
3. *$\lambda c(G) \geq 5$ if and only if $n \geq 5$ or $G = S_3$.*

Proposition 3. *If G is a graph with $n \geq 5$ vertices having a Hamiltonian path, then $\lambda c(G) = 5$.*

Corollary 4. *$\lambda c(C_n) = 4$ for $n = 3, 4$ and $\lambda c(C_n) = \lambda c(K_n) = 5$ for $n \geq 5$.*

Note that difference between $\lambda(G)$ and $\lambda c(G)$ can be large. In fact, next result holds.

Proposition 5. *For any pair of integers $a, b \geq 5$ with $a + 1 < b$ there exists a graph G with $\lambda c(G) = a$ and $\lambda(G) = b$.*

Consider now graphs obtained by some classical graph operations. We remind that for two graphs G and H the Cartesian product $G\square H$ has vertex set $V(G) \times V(H)$ and two vertices (u, v) and (u', v') are adjacent if ($u = u'$ and $vv' \in E(H)$) or ($uu' \in E(G)$ and $v = v'$). For a vertex $v \in V(H)$ denote G_v the graphs induced in $G\square H$ by vertices from $V(G) \times \{v\}$. Note that G_v is isomorphic to G. The join $G \vee H$ is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G \vee H) = E(G) \cup E(H) \cup \{uv | u \in V(G) \text{ and } v \in V(H)\}$.

Proposition 6. *Let $2 \leq m \leq n$ such that $m + n \geq 5$. Then $\lambda c(K_{m,n}) = 5$.*

Theorem 7. *Let G and H be two connected nontrivial graphs. Then $\lambda c(G \square H) = 4$ if $|V(G)| = |V(H)| = 2$, otherwise $\lambda c(G \square H) = 5$.*

Sketch of Proof. Let T and T' be spanning rooted trees of G, respectively H. Color the spanning tree of each copy G_v level by level from the root, alternating colors 1 and 5 if the level of v is even in T', and 4 and 2 otherwise. □

Proposition 8. *Let G and H be two connected nontrivial graphs. Then $\lambda c(G \vee H) = 4$ if $|V(G)| = |V(H)| = 2$, otherwise $\lambda c(G \vee H) = 5$.*

Corollary 9. *Let $2 \leq n_1 \leq n_2 \leq ... \leq n_q$. Then $\lambda c(K_{n1,...,n_q}) = 4$ if $q = 2$ and $n_1 = n_2 = 2$, otherwise $\lambda c(K_{n1,...,n_q}) = 5$.*

For a path $P = [v_1, \ldots, v_p]$, denote by $start_2(P) = v_2$ and $end_2(P) = v_{p-1}$.

The following Theorem provide an upper bound of λc for 2-connected graphs and also a particular type of $L(2,1)$-path coloring with 3 fixed colors, useful for finding an upper bound for an arbitrary graph, obtained by combining $L(2,1)$-path colorings of its blocks.

Theorem 10. *Let G be a 2-connected graph with $n \geq 4$ vertices and $L \geq 10$. Let S be a vertex in $V(G)$ and $cs, cs_1, cs_2 \in [L]$ such that $cs_1 \neq cs_2$ and $|cs - cs_i| \geq 2$ for $i = 1, 2$. Then there exists a $L(2,1)$-path coloring c of G with span at most L such that $c(S) = cs$ and each vertex v in G has an associated multiset containing two colors, namely: for $v \neq S$, $C(v) = \{cv_1, cv_2\}$ with $|c(v) - cv_i| \geq 2$, $i = 1, 2$ and $C(S) = \{cs_1, cs_2\}$ satisfying the properties:*

1. *For every pair of vertices $x \neq y \in V$ there exists a $L(2,1)$-path P_{xy} from x to y such that $c(start_2(P_{xy})) \in C(x)$ and $c(end_2(P_{xy})) \in C(y)$.*
2. *For every vertex $x \neq S$ there exist two $L(2,1)$-paths P_x and P'_x from x to S such that $c(start_2(P_x)), c(start_2(P'_x)) \in C(x)$, $c(end_2(P_x)) = cs_1$ and $c(end_2(P'_x)) = cs_2$.*

Sketch of Proof. It suffices to consider $L = 10$. Let $C_p = [v_1 = S, v_2, \ldots, v_p, v_{p+1} = v_1]$ be a cycle containing S with $p \geq 4$ (exists, since G is 2-connected). Then G has an ear decomposition such that the initial cycle is C_p [5]. The idea of the proof is by induction on the number of ears added to C_p. Consider first C_p. Color $c(S) = cs$, $c(v_2) = cs_1$ and $c(v_p) = cs_2$. Then color the other vertices of C_p as follows. If $p = 4$ choose $c(v_3) \in [L] - \{c(S), cs_1, cs_2, cs_1 \pm 1, cs_2 \pm 1\}$. Otherwise color in this order $v_3, \ldots, v_{p-3}$ such that $c(v_i) \in [L] - \{c(v_{i-1}), c(v_{i-2}), c(v_{i-1}) \pm 1\}$; then choose $c(v_{p-2}) \notin [L] - \{c(v_{p-4}), c(v_{p-3}), c(v_{p-3}) \pm 1, cs_2\}$; then color v_{p-1} such that $c(v_{p-1}) \notin [L] - \{c(v_{p-3}), c(v_{p-2}), c(v_{p-2}) \pm 1, cs_2, c(S), cs_2 \pm 1\}$. Let $C(v_i) = \{c(v_{i-1}), c(v_{i+1})\}$ for $2 \leq i \leq p$. Since c is actually a $L(2,1)$-labeling of C_p, properties 1 and 2 are satisfied.

Assume now that the statement is true before ear P is added. Denote by G' the graph before adding ear P and by G the obtained graph. By induction, there exists a $L(2,1)$-path coloring c' of G' with stated properties. Denote x and y the extremities of P such that $y \neq S$. We extend c' to a $L(2,1)$-path coloring c of G such that properties 1 and 2 are verified. This will be done by coloring vertices of P in a similar manner we colored C_p, but taking into account also colors from $C(x) = \{cx_1, cx_2\}$ and $C(y) = \{cy_1, cy_2\}$ through which we can build paths from x or y to any other vertex of G'. It is actually enough to consider P_{xy} a $L(2,1)$-path from x to y in G' with property 1 and assume $c(start_2(P_{xy})) = cx_1$. Then, if we connect vertices of P with vertices of G' only through vertex y or path P_{xy}, there is no need to avoid color cx_2 for vertices of P. □

Corollary 11. *If G is a 2-connected graph, then $\lambda c(G) \leq \min\{10, \Delta(G)+3\}$.*

Theorem 12. *Let G be a connected graph with $n \geq 5$ vertices and b the maximum number of bridges incident in the same vertex. Then $\max\{b+2, 5\} \leq \lambda(G) \leq \max\{10, b+5\}$.*

Sketch of Proof. Let $L = \max\{10, b+5\}$. We describe an algorithm for finding a $L(2,1)$-path coloring of G using at most on L colors. Consider T the block - cut vertex tree associated to G and fix as root a cut vertex r. Traverse T starting from r level by level, exploring only vertices corresponding to cut vertices. Denote V' the set of colored vertices. When a cut vertex x is explored, color as described bellow the vertices of the blocks that are direct descendants of x such that a property similar to property 1 from Theorem 10 is satisfied in $G[V']$.

For r consider $c(r) = 1$ and $C(r) = \{3\}$. Let x be cut vertex currently explored and $C(x) = \{cx_1, cx_2\}$. If $cx_2 = cx_1$ then choose $cx_3 \in [L] - \{cx_1, c(x), c(x) \pm 1\}$. Otherwise set $cx_3 = cx_2$. Modify set $C(x) = \{cx_1, cx_3\}$. Note that cx_3 is either cx_2 or a new value added to $C(x)$. Color the blocks that are direct descendants of x in T in the following order: blocks with 4, 3, respectively 2 vertices.

For blocks with 4 vertices apply Theorem 10 for $S = x$, $cs = c(x)$, $cs_1 = cx_1$, $cs_2 = cx_3$.

For a block with 3 vertices x, u_1, u_4, choose cx_4 such that $|cx_4 - cx_1| \geq 2$ and $|cx_4 - c(x)| \geq 2$ (cx_4 can be equal with cx_3). Set $c(u_i) = cx_i$, $C(u_i) = \{c(x), cx_{5-i}\}$ for $i = 1, 4$.

Let B_i, $i \in [k]$ be the blocks with 2 vertices that are direct descendants of x, corresponding to bridges xv_i. Note that $k \leq b$. Choose for $c(v_i)$ distinct colors from $[L] - \{c(x), cx_1, cx_3, c(x) \pm 1\}$ and add them to $C(x)$, then set $C(v_i) = \{c(x)\}$. □

Corollary 13. *If G is a 2-edge connected graph, then $\lambda c(G) \leq \min\{10, \Delta(G)+3\}$.*

3 Conclusions

Finding a minimum $L(2,1)$-path coloring present interest not only from theoretical point of view, but also for algorithmic approaches and applications. All 2-edge-connected graphs with at least 5 vertices presented in this paper have $L(2,1)$-connection number 5, which is the lower bound. Some natural questions raise at the beginning of the study of $L(2,1)$-connectivity: what type of graphs have $L(2,1)$-connection number 5; are there efficient algorithms to find the $L(2,1)$-connection number of a graph or at least to decide if it equals 5?

References

[1] V. Borozan, S. Fujita, A. Gerek, C. Magnant, Y. Manoussakis, L. Montero, Zs. Tuza, *Proper connection of graphs*, Discrete Mathematics, 312(17)(2012), 2550–2560.

[2] T. Calamoneri, *The $L(h,k)$-Labelling Problem: An Updated Survey and Annotated Bibliography*, Computer Journal 54(8)(2011), 1344-1371.

[3] W. K. Hale, *Frequency assignment: theory and applications*, Proc. IEEE 68(1980), 1497-1514.

[4] X. Li, C. Magnant, M. Wei, X. Zhu, *Distance proper connection of graphs*, arXiv:1606.06547.

[5] D. B. West, *Introduction to Graph Theory, Second edition*, Prentice Hall, 2001.

Domination structure with nonempty minimal edge set for cubic graphs

Misa Nakanishi[1]

[1]nakanishi@2004.jukuin.keio.ac.jp

From a research of several recent papers, we are concerned with domination number in cubic graphs. A cubic graph is partitioned by substructures which are prescribed for a minimal edge set of a minimum dominating set. It is applied to Reed's conjecture and gives a sufficient condition. Also, the minimum dominating set of 3-connected cubic graphs is solved.

1 Introduction

In this paper, a graph G is simple and undirected with a vertex set V and an edge set E. A *dominating set* $X \subset V$ is such that every vertex of $V - X$ is adjacent to a vertex of X. The minimum cardinality taken over all minimal dominating sets of G is the *domination number* $\gamma(G)$. The minimum cardinality taken over all maximal independent sets of G is the *independent domination number* $i(G)$. A minimum dominating set is called a *d-set*.

For the domination number of a graph, in decades the research on cubic graphs has intensively studied that show several important results. A random 3-regular graph asymptotically almost surely has no 3-star factors [2]. Reed indicated that almost all cubic graphs are hamiltonian, also the upper bound of the domination number of a connected cubic graph G is conjectured as $\lceil |G|/3 \rceil$ [6]. Then the counterexamples that exceed the bound have shown, for example, there is an extremal graph of the domination number 21 over 60 vertices following the series of cubic graphs beyond the boundary [5] [4].

The connected cubic graphs that have the domination number above the bound have a minimum dominating set as an independent set. Otherwise the conjecture is true.

Also, we refine the structure of path covers used in the proof of cubic graph domination by Reed. The minimum dominating set of 3-connected cubic graphs is determined by our proposed structure.

2 Preliminary results

A sufficient condition for $\gamma(G) = i(G)$ was represented for a general graph G by an induced subgraph isomorphic to $K_{1,3}$, also called 3-star, free [1].

Proposition 1 ([1]). *If G does not have an induced subgraph isomorphic to $K_{1,3}$, then $\gamma(G) = i(G)$.*

An induced subgraph, say I, is defined as $N[v_1] \cup N[v_2]$ on two adjacent vertices v_1 and v_2 with degree at least three. We observe I as a forbidden subgraph for $\gamma(G) = i(G)$ with the simplest proof.

Proposition 2 ([3]). *For a graph G, if $I \not\subseteq G$ then $\gamma(G) = i(G)$.*

Proof. Let X be a d-set of G and $E(X)$ be minimal. For $x, y \in X$ such that $xy \in E(G)$, let $d_G(x) = 2$ and $N_G(x) - y = \{x'\}$. For all $z \in N_G(x') - x$, if $z \notin X$ then $X - \{x\} \cup \{x'\} = X'$ that is a d-set. $||X|| - 1 \geq ||X'||$ contrary to the minimality of $E(X)$. If there is $z \in N_G(x') - x$ such that $z \in X$ then $X - \{x\} = X''$ that is a d-set and contrary to the minimality of $V(X)$. □

A 3-connected cubic graph was conjectured as the difference between the independent domination number and the domination number is one, but it was disproved as taken in infinity.

Proposition 3 ([7]). *For any $c \in \{0, 1, 2, 3\}$ and any integer $k \geq 0$ there exist infinitely many cubic graphs with connectivity c (say one as G) for which $i(G) - \gamma(G) = k$.*

We consider a completion of Reed's conjecture. The next statement is suggested. It has some counterexamples in cubic graphs with connectivity one and two. A graph H_4 in [5], for example, is observed as it has a minimum dominating set as an independent set.

Conjecture ([6]). Every connected cubic graph G contains a dominating set of $\lceil |G|/3 \rceil$ vertices.

3 Main theorem

Theorem 4. *For a connected cubic graph G, if $\gamma(G) > \lceil |V|/3 \rceil$ then $\gamma(G) = i(G)$.*

Let C_G be a cycle space with length 0 mod 3 for a graph G. Two cycles are connecting without *seam* if and only if one of them is constructed by adding one ear (not a cycle) to the other. Let C_0 be a maximal subset of C_G such that cycles are connecting without seam.

Theorem 5. *For a 3-connected cubic graph G, C_0 determines its d-set.*

4 Perspective

Further, we have a proposition.

Theorem 6. *For a 3-connected graph G, C_0 determines its d-set.*

References

[1] Robert B. Allan, Renu Laskar: On domination and independent domination numbers of a graph. Discrete Mathematics. 23, 73-76 (1978)

[2] Hilda Assiyatun, Nicholas Wormald: 3-star factors in random d-regular graphs. European Journal of Combinatorics. 27, 1249-1262 (2006)

[3] E. J. Cockayne, O. Favaron, C. M. Mynhardt, J. Puech: A characterization of (γ, i)-trees. Journal of Graph Theory. 34, 277-292 (2000)

[4] Alexander Kelmans: Counterexamples to the cubic graph domination conjecture. arXiv:0607512v1. 20 Jul 2006

[5] A. V. Kostochka, B. Y. Stodolsky: On domination in connected cubic graphs. Discrete Mathematics. 304, 45-50 (2005)

[6] Bruce Reed: Paths, stars and number three. Combinatorics, Probability and Computing. 5, 277-295 (1996)

[7] I. E. Zverovich, V. E. Zverovich: Disproof of a conjecture in the domination theory. Graphs and Combinatorics. 10, 389-396 (1994)

The Greedy Algorithm for Capacitated Covering Problems

Britta Peis[1], José Verschae[2], and Andreas Wierz[1]

[1]RWTH Aachen University
[2]Pontificia Universidad Católica de Chile

1 Introduction

Integer programs of the form $\min\{c^T x : Ax \geq r, x \in \mathbb{Z}_+^n\}$ (P) with $A \in \mathbb{Z}_+^{\mathcal{L}\times E}$, $r \in \mathbb{Z}^{\mathcal{L}}$ and $c \in \mathbb{Z}_+^E$ are called covering problems. Here, we assume that $\mathcal{L}$ is a family of subsets of the elements which, for each row of A, describes the columns which are zero. That is, $S \in \mathcal{L}, e \in S \Rightarrow A_{S,e} = 0$. The name stems from the following observation: solutions to such problems use (multiple) copies of the columns in order to cover the righthandside value for each constraint. Since all coefficients are non-negative, adding additional copies of a column can not render a solution vector infeasible.

There are interesting connections between covering and scheduling problems. In fact, several single machine scheduling problems such as $1|pmnt|\sum_j f_j(C_j)$, with f_j non-decreasing, or $1|r_j|\sum_j w_j C_j$ can be formulated as covering problems in a relatively simple way. The former was pointed out in [11] to be a generalization of unsplittable flow cover on a path, which can be 4-approximated [1].

Besides this, many interesting combinatorial optimization problems can also be formulated in this manner. Famous examples are subset cover, cut covering and contra-polymatroids. Contra-polymatroids are of the form above with $\mathcal{L}$ being the Boolean lattice and constraints of the form $\sum_{e\notin S} x_e \geq r(S)$ for all $S \in \mathcal{L}$. The function $r : \mathcal{L} \to \mathbb{Z}_+$ is supermodular, monotone decreasing and non-negative. For contra-polymatroids, a very simple dual greedy algorithm is able to obtain an optimum solution. The dual to the linear relaxation of the covering problem above can be stated as $\max\{y^T r : y^T A \geq c, y \geq 0\}$ (D). Set $S = \emptyset$, increase the corresponding dual variable y_S until some element e gets tight. Set $x_e = r(S + e) - r(S)$, add e to S and iterate until $r(S) = 0$. Polymatroids, and the depicted greedy algorithm, are known for almost half a century [4].

For the knapsack cover problem, and many other problems as well, the exact same algorithm was used in order to obtain constant factor approximations [1, 2, 12]. In most of these known approximation results, a problem specific analysis was used. Many times, these analyses look very similar and use essentially the same techniques.

If $A \in \{0,1\}^{m\times n}$, there are fairly general conditions on the system (A, r) known, which ensure that the greedy algorithm computes an optimum solution [4, 5, 6, 8, 9, 10]. Most results have in common that the constraints form some kind of lattice on which r is either submodular or supermodular. We discuss similar conditions in case of $A \in \mathbb{Z}_+^{m\times n}$ and approximate solutions.

2 Our results

We consider integer covering problems (P) with constraints for subsets of elements which form a ring-family. A family $\mathcal{F}$ of subsets of groundset E is called a ring-family if any two sets $S, T \in \mathcal{F}$ imply $S \cup T, S \cap T \in \mathcal{F}$. That is, a family is called a ring-family, if it is closed under union and intersection. We also assume that $\emptyset, E \in \mathcal{F}$. A ring-family forms a lattice $\mathcal{L} = (\mathcal{F}, \subseteq, \cup, \cap)$. The Boolean lattice is an example for a ring-family.

We analyze the following simple dual greedy algorithm. Initialize $y \equiv 0, x \equiv 0$. Select the minimal element in $\mathcal{L}$ and increase y_S until $\sum_S y_S A_{S,e} = c_e$ some element $e \notin S$. Let S' be the minimal element in $\mathcal{L}$ which also contains e. Set $x_e = \lceil \frac{r(S)^+ - r(S')^+}{A_{S,e}} \rceil$. Remove all elements from $\mathcal{L}$ which do not contain S' and iterate until $r(S) \leq 0$.

In general, we may not assume that the greedy algorithm will terminate with a feasible solution for (P). But we can show that it will, if the system (A, r) satisfies the following conditions.

(P1) r is monotone: $r(S) \geq r(T)$ for all $S \preceq T$ and

(P2) For each element $e \in E$, $A_{*,e}$ is monotone: $A_{S,e} \geq A_{T,e}$ for all $S \preceq T$.

(P3) For every two sets $S \preceq T \in \mathcal{L}$ with $r(T) > 0$ and $e \notin T$, let $S' = \min(\mathcal{L} \setminus S \setminus e)$ and $T' = \min(\mathcal{L} \setminus T \setminus e)$. Then $\frac{r(S)-r(S')}{A_{S,e}} \geq \frac{r(T)-r(T')}{A_{T,e}}$.

Unfortunately, these properties do not suffice in order to obtain good approximation guarantees. Even with two elements, we can construct instances which have an unbounded integrality gap. Moreover, we can show that subset cover can be formulated as a problem of type (P) satisfying (P1) - (P3). Hence, no $o(1 - \log n)$ approximation for (P) exists unless $NP = DTIME(n^{O(\log \log n)})$ [7]. The former issue can be handled by a careful truncation of coefficients of the matrix A.

Definition 1. *Given a system (A, r) on a ring-family satisfying (P1) - (P3), we define the system (A', r) as below to be the* truncated system. *For $S \in \mathcal{L}$ and $e \notin S$, let $S' = \min(\mathcal{L} \setminus S \setminus e)$. Then set $A'_{S,e} = \min\{A_{S,e}, r(S)^+ - r(S')^+\}$.*

We can show that the truncated system (A', r) contains the same integer feasible points as the system (A, r). Although the truncation no longer satisfies (P3), we can also show that the greedy algorithm applied to system (A', r) still obtains a feasible solution. The approximation factor depends on the following parameter δ. Given $S \in \mathcal{L}$ with $e \notin S$, let $S' = \min(\mathcal{L} \setminus S \setminus e)$ and define $\delta_{S,e} = \frac{A'_{\emptyset,e}}{A'_{S,e}}$, if $r(S') \geq 0$ and $A'_{S,e} > 0$, and $\delta_{S,e} = 1$, otherwise. Let $\delta = \max_{S,e} \delta_{S,e}$ be the maximum of these terms.

The value of δ is the maximal ratio between coefficients in A' which may occur on a chain in the dual constructed by the greedy algorithm before the rank becomes negative. Intuitively, the following happens: If the algorithm selects many elements with small coefficient $A'_{S,e}$, these may have a very large contribution $A'_{\emptyset,e} \leq \delta A'_{S,e}$ towards the constraint for $S = \emptyset$. Hence, a small value of δ guarantees that no constraint is oversubscribed by a large factor. This, in-turn, results in a good approximation guarantee using standard arguments. Our main result, stated in its simplest form, is the following.

Theorem 2. *The greedy algorithm applied to the truncation (A', r) of a system (A, r) on a ring-family satisfying (P1) - (P3) obtains a solution with cost no larger than $b(\delta + 1)OPT$,*

or $b\delta OPT$, if r is non-negative. Here, $b = 1$, if $\frac{r(S)^+ - r(S')^+}{A'_{S,e}} \in \mathbb{Z}_+$ for all $S \in \mathcal{L}, e \notin S, S' = \min(\mathcal{L} \setminus S \setminus e)$, and $b = 2$, otherwise.

Our approximation factors coincide with many well-known results, for example, for polymatroids, polymatroid intersection, vertex cover, knapsack cover with item multiplicity and separable convex cost- and concave utility functions, generalized steiner trees, minimum multicut on trees, knapsack cover with precedence constraints, or flow cover on a line. In terms of lower bounds, we can construct a family of instances whose truncation (A', r) has an integrality gap of $o(\log \delta)$.

References

[1] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *JACM*, 2001.

[2] T. Carnes and D. Shmoys. Primal-dual schema for capacitated covering problems. *IPCO*, 2008.

[3] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 1979.

[4] J. Edmonds. Submodular functions, matroids, and certain polyhedra. *Combinatorial structures and their applications*, 1970.

[5] U. Faigle and S. Fujishige. A general model for matroids and the greedy algorithm. *Mathematical programming*, 2009.

[6] U. Faigle and W. Kern. An order-theoretic framework for the greedy algorithm with applications to the core and weber set of cooperative games. *Order*, 2000.

[7] U. Feige. A threshold of ln n for approximating set cover. *JACM*, 1998.

[8] A. Frank. Increasing the rooted-connectivity of a digraph by one. *Mathematical Programming*, 1999.

[9] S. Fujishige. A note on frank's generalized polymatroids. *Discrete Applied Mathematics*, 1984.

[10] S. Fujishige. Dual greedy polyhedra, choice functions, and abstract convex geometries. *Discrete Optimization*, 2004.

[11] W. Höhn, J. Mestre, and A. Wiese. How unsplittable-flow-covering helps scheduling with job-dependent cost functions. *ICALP*, 2014.

[12] S. McCormick, B. Peis, J. Verschae, and A. Wierz. Primal–dual algorithms for precedence constrained covering problems. *Algorithmica*, 2016.

[13] A.Schulz and J. Verschae. Min-sum scheduling under precedence constraints. *LIPIcs-Leibniz International Proceedings in Informatics*, 2016.

Brouwer's conjecture on the sum of Laplacian eigenvalues of a graph

Shariefuddin Pirzada[1] and Hilal A. Ganie[2]

[1,2]Department of Mathematics, University of Kashmir, Srinagar, Kashmir, India

Let G be a simple graph with n-vertices, m edges and having Laplacian eigenvalues $\mu_1, \mu_2, \ldots, \mu_{n-1}, \mu_n = 0$. Let the sum of the k largest Laplacian eigenvalues of G be $S_k(G) = \sum_{i=1}^{k} \mu_i$. Brouwer conjectured that $S_k(G) \leq m + \binom{k+1}{2}$, for all $k = 1, 2, \ldots, n$. We obtain upper bounds for $S_k(G)$ in terms of the clique number ω, the vertex covering number τ and the diameter d of a graph G. We show that Brouwer's conjecture holds for certain classes of graphs.

1 Introduction

Let $G(V, E)$ be a simple graph with n vertices and m edges having the vertex set $V(G) = \{v_1, v_2, \ldots, v_n\}$ and the edge set $E(G) = \{e_1, e_2, \ldots, e_m\}$. The adjacency matrix $A = (a_{ij})$ of G is a $(0,1)$-square matrix of order n whose (i,j)-entry is equal to 1 if v_i is adjacent to v_j and equal to 0, otherwise. Let $D(G) = diag(d_1, d_2, \ldots, d_n)$ be the diagonal matrix associated to G, where $d_i = \deg(v_i)$, for all $i = 1, 2, \ldots, n$. The matrix $L(G) = D(G) - A(G)$ is called the Laplacian matrix and its spectrum is called the Laplacian spectrum (L-spectrum) of the graph G. This is a real symmetric and positive semi-definite matrix. Further, we take $0 = \mu_n \leq \mu_{n-1} \leq \cdots \leq \mu_1$ to be the L-spectrum of G. It is well known that μ_n=0 with multiplicity equal to the number of connected components of G and $\mu_{n-1} > 0$ if and only if G is connected [6]. Let $S_k(G) = \sum_{i=1}^{k} \mu_i$, $k = 1, 2, \ldots, n$, be the sum of k largest Laplacian eigenvalues of G and let $d_i^*(G) = |\{v \in V(G) : d_v \geq i\}|$, for $i = 1, 2, \ldots, n$. In 1994, Grone and Merris [9] observed the following.

Theorem 1. *(Grone-Merris Theorem) For any graph G and for any k, $1 \leq k \leq n$, $S_k(G) \leq \sum_{i=1}^{k} d_i^*(G)$.*

This was proved by Hua Bai [1] and is now called as Grone-Merris theorem. As an analogue to Grone-Merris theorem, Andries Brouwer [2] conjectured the following.

Conjecture 2. *If G is a graph with n vertices and m edges, then for any k, $k = 1, 2, \ldots, n$*

$$S_k(G) = \sum_{i=1}^{k} \mu_i \leq m + \binom{k+1}{2}.$$

Using computations on a computer, Brouwer [2] checked this conjecture for all graphs with at most 10 vertices. For $k = 1$, the conjecture follows from the well-known inequality $\mu_1(G) \leq n$ (see [3]). Also, the cases $k = n$ and $k = n - 1$ are straightforward. Haemers et al. [10] showed that the conjecture is true for all the graphs when $k = 2$ and is also true for the trees. Du et al. [5] obtained various upper bounds for $S_k(G)$ and proved that the conjecture is also true for unicyclic and bicyclic graphs. Rocha et al. [13] obtained various upper bounds for $S_k(G)$, which improve the upper bounds obtained in [5] for some cases and proved that the conjecture is true for all k with $1 \leq k \leq \lfloor \frac{g}{5} \rfloor$, where g is the girth of the graph G (the length of smallest cycle in G is called girth of the graph). They also showed that the conjecture is true for a connected graph G having maximum degree Δ with p pendant vertices and c cycles for all $\Delta \geq c + p + 4$. For the progress on this conjecture, we refer to [2, 8, 10, 12, 13].

A clique of a graph G is the maximum complete subgraph of the graph G. The order of the maximum clique is called the clique number of the graph G and is denoted by ω. A subset S of the vertex set $V(G)$ is said to be a covering set of G if every edge of G is incident to at least one vertex in S. A covering set with minimum cardinality among all covering sets is called the minimum covering set of G and its cardinality is called the vertex covering number of G, and is denoted by τ. The distance between any two vertices u and v is defined as the length of the shortest path between them and the diameter d of a graph G is the maximum distance among all pair of vertices of G. As usual, K_n and $K_{s,t}$ denote, respectively, the complete graph on n vertices and the complete bipartite graph on $s + t$ vertices.

If σ, $(1 \leq \sigma \leq n-1)$ is the number of Laplacian eigenvalues greater than or equal to average degree $\overline{d}$, a lower bound for $S_{\omega-1}(G)$ and an upper bound for $S_\sigma(G)$ in terms of m, Δ, σ and clique number ω of the graph can be seen in [11].

2 Upper bounds for $S_k(G)$

Du et al. [5] obtained an upper bound for $S_k(G)$ in terms of clique number ω as

$$S_k(G) \leq k\omega + 2m - \omega(\omega - 1). \tag{1}$$

Das et al. in [4] obtained an upper bound for $S_k(G)$, in terms of vertex covering number τ as

$$S_k(G) \leq m + k\tau, \tag{2}$$

if $G \cong K_{1,n-1}$ then equality occurs.

Now, we obtain an upper bound for $S_k(G)$, in terms of the clique number ω and the vertex covering number τ.

Theorem 3. *Let G be a connected graph of order $n \geq 2$ with m edges and let ω be the clique number and τ be the vertex covering number of G. Then*

$$S_k(G) \leq k(\tau + 1) + m - \frac{\omega(\omega - 1)}{2}, \tag{3}$$

with equality if and only if $G \cong K_n$.

It is an easy task to see that the upper bound (3) improves the upper bound (2) for $k \leq \frac{\omega(\omega-1)}{2}$. That is, for the higher values of the clique number ω, the upper bound (3) is better than the upper bound (2). We have the following upper bound for $S_k(G)$ in terms of the diameter d and the vertex covering number τ.

Theorem 4. *Let G be a connected graph of order n with m edges having diameter d and vertex covering number τ. Then*

$$S_k(G) \leq (\tau - \lfloor \frac{d}{2} \rfloor + 2)k + m - d + \cos\left(\frac{k\pi}{d}\right) + \frac{\cos(\frac{\pi}{d})\sin(\frac{k\pi}{d}) + \sin(\frac{k\pi}{d})}{\sin(\frac{\pi}{d})}, \tag{4}$$

with equality if and only if $G \cong P_n$.

A similar type of upper bound for $S_k(G)$ in terms of diameter, was obtained by Rocha et al. [13]. It can be easily seen that for the graphs with large number of edges, the upper bound (6) is better than the upper bound obtained by Rocha et al. in Theorem 1 of [13]. If K_{s_1,s_2} $(s_1 \leq s_2)$ is the maximal complete bipartite subgraph of graph G, using the fact that the vertex covering number of K_{s_1,s_2} $(s_1 \leq s_2)$ is s_1 and proceeding similarly as in Theorem 2.2, we have an upper bound for $S_k(G)$ as follows.

Theorem 5. *Let G be a connected graph of order $n \geq 2$ with m edges having the vertex covering number τ. If K_{s_1,s_2} $(s_1 \leq s_2)$, is the maximal complete bipartite subgraph of graph G, then*

$$S_k(G) \leq k(\tau + s_2 - s_1) + m - s_1(s_2 - 1), \tag{5}$$

with equality if and only if $G \cong K_{s_1,s_2}$ with $s_1 + s_2 = n$.

If $s_1 = s_2$, it is easy to see that the upper bound (5) is always better than the upper bound (2).

3 Brouwer's conjecture for some classes of graphs

Now, we show that the Brouwer's conjecture holds for some classes of graphs.

Theorem 6. *For a connected graph G of order $n \geq 24$ having clique number $\omega \geq \frac{7n}{8}$, we have $S_k(G) \leq m + \frac{k(k+1)}{2}$, for all $k = 1, 2, \ldots, \lfloor \frac{n}{2} \rfloor$.*

Let Ω_n be a family of connected graphs for which the clique number ω is one more than the vertex covering number τ, that is, $\Omega_n = \{G : G \text{ is connected of order } n \text{ with } \omega = \tau + 1\}$. For the family of graphs Ω_n, we have the following observation.

Theorem 7. *If $G \in \Omega_n$, for all k, $1 \leq k \leq n$, then $S_k(G) \leq m + \frac{k(k+1)}{2}$.*

Theorem 8. *Let G be a connected graph of order $n \geq 2$ with m edges having the vertex covering number $\tau \leq 1.3s_1$. If K_{s_1,s_1} is the maximal complete bipartite subgraph of the graph G, for all $k = 1, 2, \ldots, n$, then $S_k(G) \leq m + \frac{k(k+1)}{2}$.*

For connected bipartite graphs of order n, the vertex covering number $\tau \leq \frac{n}{2}$. For the bipartite graphs, we have the following observation.

Theorem 9. *Let G be a connected bipartite graph of order $n \geq 2$ with m edges having the vertex covering number τ. If K_{s_1,s_1} with $s_1 \geq \frac{n}{4}$ is the maximal complete bipartite subgraph of graph G, then $S_k(G) \leq m + \frac{k(k+1)}{2}$, for all $k \leq \frac{n}{7} - 1$ and $k \geq \frac{6n}{7}$.*

Let ρ be the family of those graphs in which the removal of the edges of the clique K_ω results in a forest or a unicyclic graph. We verify Brouwer's conjecture for some classes of graphs in ρ.

References

[1] H. Bai, The Grone-Merris conjecture, Trans. Amer. Math. Soc. 363 (2011) 4463–4474.

[2] A. E. Brouwer and W. H. Haemers, Spectra of graphs. Available from: http://homepages.cwi.nl/aeb/math/ipm.pdf.

[3] D. Cvetkovic, M. Doob and H. Sachs, Spectra of graphs-Theory and Application, Academic Press, New York, 1980.

[4] K. C. Das, S. A. Mojallal and I. Gutman, On Laplacian energy in terms of graph invariants, Applied Mathematics and Computation 268 (2015) 83–92.

[5] Z. Du and B. Zhou, Upper bounds for the sum of Laplacian eigenvalues of graphs, Linear Algebra Appl. 436 (2012) 3672-3683.

[6] M. Fiedler, Algebraic Connectivity of Graphs, Czechoslovak Math. J. 23 (1973) 298-305.

[7] E. Fritscher, C. Hoppen, I. Rocha and V. Trevisan, On the sum of the Laplacian eigenvalues of a tree, Linear Algebra Appl. 435 (2011) 371-399.

[8] H. A. Ganie, A. M. Alghamdi and S. Pirzada, On the sum of the Laplacian eigenvalues of a graph and Brouwer's conjecture, Linear Algebra Appl. 501 (2016) 376-389.

[9] R. Grone and R. Merris, The Laplacian spectrum of a graph II, SIAM J. Discrete Math. 7 (1994) 221-229.

[10] W. H. Haemers, A. Mohammadian and B. Tayfeh-Rezaie, On the sum of Laplacian eigenvalues of graphs, Linear Algebra Appl. 432 (2010) 2214-2221.

[11] S. Pirzada and Hilal A. Ganie, On the Laplacian eigenvalues of a graph and Laplacian energy, Linear Algebra Appl. 486 (2015) 454–468.

[12] S. Pirzada and Hilal A. Ganie, Spectra, energy and Laplacian energy of strong double graphs, Springer Proceedings of Mathematics and Statistics, Mathematical Technology of Networks, D. Mugnolo (ed.) 128, 175-189.

[13] I. Rocha and V. Trevisan, Bounding the sum of the largest Laplacian eigenvalues of graphs, Discrete Applied Math. 170 (2014) 95–103.

Geometry of gross substitutes valuations

Stephen Raach[1] and Sven de Vries[2]

[1]Trier University
[2]Trier University

We consider the set of gross substitutes valuations which were introduced by Kelso and Crawford [4]. The gross substitutes condition has important applications in auction theory, for example the existence of a Walrasian equilibrium is guaranteed if all bidders have gross substitutes valuations. Following an idea of Lehman [5] we interprete valuation functions $v : 2^N \to \mathbb{R}$ on $N := \{1, ..., n\}$ as points in $\mathbb{R}^{2^n-1}$. In that paper the authors point out that the set of gross substitutes valuations has Lebesque-measure zero and raise the question whether the dimension of the gross substitutes valuations is polynomial in the number of items n. The set of gross substitutes valuations turns out to be the union of finitely many polyhedral cones. We answer the question showing that the dimension of each of these cones is actually exponential in n and provide a lower bound of $\lceil \frac{1}{n+1} \cdot (2^n - n - 2) \rceil + 2n - 1$ for the cones dimension. By explicit calculation we verify that our lower bound matches the dimension in the case of $n \leq 3$ but has a deficit of 1 already in the case $n = 4$.

1 Introduction

The class of gross substitutes valuations has important impact on the theory of combinatorial auctions. This is due to the nice properties that combinatorial auctions with bidders posessing gross substitutes valuations have, as i.e. the problem of maximizing the social welfare can be solved in polynomial time or the existence of a Walrasian equilibrium is guaranteed. Combinatorial auctions with gross substitutes have been widely studied by Gul and Stacchetti and by Milgrom [3, 6]. The gross substitutes condition has been introduced by Kelso and Crawford [4] as a sufficient condition under which a salary adjustment process converges to an equilibrium in a market consisting of firms and workers. Gul and Stacchetti [2] showed the importance of this class by proving that the gross substitutes condition is necessary to ensure a Walrasian equilibrium. Important steps towards understanding the class of gross substitutes valuations were made by Murota [7], linking mathematical economics with discrete convex analysis. As a consequence of this connection a price free description of gross substitutes could be given by Fujishige and Yang and by Reijnierse et al. [1, 8]. We will make use of a variaton of the earlier description to prove the main result of our paper, showing that the set of gross substitutes valuations $v\colon 2^N \to \mathbb{R}_+$ interpreted as a subset of $\mathbb{R}^{2^N}$ consists of the union of finitely many polyhedral cones each having dimension exponential in n . The question whether the dimension of the mentioned polyhedrons is polynomial in n was posed by Lehmann et al. [5]. In the latter paper, it was shown that the set of submodular valuations is full-dimensional in $\mathbb{R}^{2^N-1}$. We use their construction of a full-dimensional polyhedral subset of the set of submodular valuations

and some graph theory to develop an at least $\lceil \frac{1}{n} \cdot 2^n \rceil + 2n - 3$ dimensional polyhedral cone contained in the set of gross substitutes valuations.

2 Preliminaries

2.1 Definitions

In this paper, we focus on a special class of valuation functions that agents in a combinatorial auction might have. We will begin by giving a few relevant definitions. We abbreviate $S \cup \{a\}$ by $S \cup a$, $S \cup \{a\} \cup \{b\}$ by $S \cup ab$ and $v(\{a\})$ by $v(a)$.

Definition 1. *We define a* ***valuation*** *on $n \in \mathbb{N}$ items, using the notation $N := \{1, ..., n\}$, as a function $v\colon 2^N \to \mathbb{R}_+$ being* ***nondecreasing*** *($v(A) \leq v(B)$ if $A \subseteq B$) and* ***normalized*** *($v(\emptyset) = 0$).*

Instead of the original definition of gross substitutes valuations which involves prices $p \in \mathbb{R}^n_+$ we use the price-free characterization by Reijnierse et al. [8] which is more convenient for our purposes:

Definition 2. *Let $n \in N$. A valuation $v\colon 2^N \to \mathbb{R}_+$ is gross substitutes, iff v is submodular and for all $S \subset N$,$a, b, c \in N \setminus S$ holds*

$$v(S \cup ab) + v(S \cup c) \leq \max\{v(S \cup ac) + v(S \cup b), v(S \cup bc) + v(S \cup a)\}.$$

A trivial consequence of this definitnion is the following reformulation of which we will make use later. A submodular valuation is gross substitutes, if for all $S \subset N$,$a, b, c \in N \setminus S$ the triple

$$\Big(v(S \cup ab) + v(S \cup c), v(S \cup ac) + v(S \cup b), v(S \cup bc) + v(S \cup a)\Big) \tag{1}$$

has a double maximum. We call this property the *triple condition*.

Definition 3. *A valuation $v\colon 2^N \to \mathbb{R}_+$ is called* ***additive*** *if $v(S) = \sum_{i \in S} v(i)$ for all $S \subseteq N$.*

Whilst general sums of gross substitutes valuations do not preserve gross substitutability it is easy to prove that the sum of an additive valuation and a gross substitutes valuation is gross substitutes.

2.2 Interpreting valuations as points of $\mathbb{R}^{2^n-1}$

With the idea of analyzing the structure of certain classes of valuation functions Lehman et al. [5] interpreted valuation functions $v\colon 2^N \to \mathbb{R}_+$ on n items as points of $\mathbb{R}^{2^n-1}$. This means we identify the axes of the real valued vector space $\mathbb{R}^{2^n}$ with the elements of 2^N. Since valuation functions are normalized, the value of the coordinate induced by $\emptyset$ is always 0 and it is reasonable to omit this axis. As usual the dimension of a polyhedral cone P, $dim(P)$, is the cardinality of the largest linear independent subset of P.
Using the mentioned identification with the real valued vector space $\mathbb{R}^{2^n-1}$ Lehman et al. [5] showed that the polyhedral cone of the submodular valuations, P_{sub}, is fully dimensional by constructing the following cone V_{sub}.

Theorem 4. *Let $n \in \mathbb{N}$, $n \geq 2$ with $g\colon 2^N \to \mathbb{R}_+$, $g(S) := 1 - \left(\frac{1}{2}\right)^{|S|}$ and $H := \Big\{h\colon 2^N \to \mathbb{R}$, with $h(\emptyset) = 0$, $|h(S)| \leq \left(\frac{1}{2}\right)^{n+2}$ for $S \subseteq N\Big\}$. Then $V_{sub} := \{v\colon v = g + h, h \in H\}$ has, as a translation of a hypercube, postive Lebesgue-measure in $\mathbb{R}^{2^n-1}$ and is a subset of the submodular valuations.*

Valuations for which holds $f(S) = f(T)$ whenever $|S| = |T| = k$ are called **symmetric for k**. Valuations for which holds $f(S) = f(T)$ whenever $|S| = |T|$ are called **symmetric**. Also, we will use $GS_n \subseteq \mathbb{R}^{2^n-1}$ for the set of gross substitutes valuations.

3 A bound for the dimension of the gross substitutes valuations describing cones

3.1 GS_n is the union of finitely many polyhedral cones

Before we take a closer look at the GS_n describing cones, we show that GS_n actually consists of the union of polyhedral cones. This statement was implicit in [5] et al. . We define $\mathcal{L} := \{(S, \{abc\})$ for $S \subseteq N, a, b, c \in N \setminus S\}$, $\mathcal{F} := \{(S, \{ab\}, c)$ for $(S, \{abc\}) \in \mathcal{L}\}$ and $\mathcal{M} := \{M \subseteq \mathcal{F}\colon$ For all $(S, \{abc\}) \in \mathcal{L}$ it holds either $(S, \{ab\}, c) \in M$ or $(S, \{ac\}, b) \in M$ or $(S, \{bc\}, a) \in M\}$. Furthermore we define the cone $P_{(S,\{ab\},c)} := \{v \in \mathbb{R}_+^{2^n-1}\colon v_{Sab} + v_{Sc} \leq v_{Sac} + v_{Sb} = v_{Sbc} + v_{Sa}\}$ and consequently $P_M = \bigcap_{(S,\{a,b\},c)\in M} P_{(S,\{ab\},c)}$ for $M \subseteq \mathcal{F}$. If $v \in GS_n$, then there has to exist a $M \in \mathcal{M}$ with $v \in P_M$. This suffices to see that GS_n is the union of finitely many polyhedral cones since $P_{sub} \cap P_M \subseteq GS_n$ for $M \in \mathcal{M}$.

3.2 A bound for the cones in GS_n

We want to find a subset $M \subseteq \mathcal{M}$ for which P_M has dimension exponential in n. Towards this we make use of symmetric valuations for all $k \leq n$ except for one k and the stirling formula. We show a simple bound:

Theorem 5. *Let $n \in \mathbb{N}$ be even and $GS_n^{eq} := \{v \in GS_n\colon v$ is symmetric for $k \leq n, k \neq \frac{n}{2}\}$. Then, there exist $M \in \mathcal{M}$ with $P_M \cap P_{sub} \subseteq GS_n^{eq}$ and $P_M \cap P_{sub}$ having dimension exponential in n.*

With further machinery, we improve this to:

Theorem 6. *There exist cones in GS_n with dimension at least $\lceil \frac{1}{n+1} \cdot (2^n - 2 - n) \rceil + 2n - 1$.*

4 Conclusion

We interpreted the set of gross substitutes valuations as a set in a real vector space and analyzed the geometry of that set. It has been shown that this set is the union of finitely many polyhedral cones and we analyzed the dimension of these cones. We started with a simple bound with the aim of showing that these cones have dimension exponential in n. After improving this bound step by step we obtained a bound that is best possible with $O(n)$. An important consequence of our main result is that an arbitrary gross substitutes valuation cannot be stored on a computer in polynomial time as the valuation takes in n exponentially many independent values.

References

[1] Satoru Fujishige and Zaifu Yang. A note on Kelso and Crawford's gross substitutes condition. *Mathematics of Operations Research*, 28(3):463–469, 2003.

[2] Faruk Gul and Ennio Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic theory*, 87(1):95–124, 1999.

[3] Faruk Gul and Ennio Stacchetti. The english auction with differentiated commodities. *Journal of Economic theory*, 92(1):66–95, 2000.

[4] Alexander Kelso Jr and Vincent Crawford. Job matching, coalition formation, and gross substitutes. *Econometrica: Journal of the Econometric Society*, pages 1483–1504, 1982.

[5] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *arXiv preprint cs/0202015*, 2002.

[6] Paul Robert Milgrom. *Putting auction theory to work*. Cambridge University Press, 2004.

[7] Kazuo Murota. *Discrete convex analysis*. SIAM, 2003.

[8] Hans Reijnierse, Anita van Gellekom, and Jos AM Potters. Verifying gross substitutability. *Economic Theory*, 20(4):767–776, 2002.

Coloring H-free Graphs: Structure, Algorithms, Open Problems

Oliver Schaudt[1]

[1]RWTH Aachen, Germany

Since the 00's the problem of coloring graphs with one (or two) forbidden induced subgraphs has been an active topic in the graph algorithms community. By now there exist three survey articles on this problem: by Chudnovsky, by Hell and Huang, and by Golovach et al.

In this talk I will discuss some interesting structural and algorithmic results on this problem, some of our own work, and some open problems for future research. I will tackle certifying algorithms and issues related to parameterized complexity in this context.

The Parameterized Complexity of the Equidomination Problem

Oliver Schaudt[1] and Fabian Senger[1]

[1]University of Cologne, Department of Computer Science,
schaudto@uni-koeln.de, senger@zpr.uni-koeln.de

A graph $G = (V, E)$ is called equidominating if there exists a value $t \in \mathbb{N}$ and a weight function $\omega\colon V \to \mathbb{N}$ such that the total weight of a subset $D \subseteq V$ is equal to t if and only if D is a minimal dominating set. To decide whether or not a given graph is equidominating is referred to as the EQUIDOMINATION problem.

In this paper we show that the EQUIDOMINATION problem is fixed-parameter tractable when parameterized in two different ways: the first parameterization considers the target value t leading to the TARGET-t EQUIDOMINATION problem. The second parameterization allows only weights up to a fixed value k, which yields the k-EQUIDOMINATION problem.

Keywords: minimal dominating set · equidominating graph · kernelization · parameterized complexity

1 Introduction

Let G be a simple, undirected graph. A subset S of the vertices of G is called a **dominating set** or simply **dominating**, if every vertex of G is an element of S or adjacent to a vertex of S. If a dominating set does not contain another dominating set as a subset, it is called a **minimal dominating set**. Throughout this paper we use the abbreviation **mds** for minimal dominating sets.

While the main stream of the research on dominating sets in graphs focuses on the optimization aspects of the problem, there are several interesting graph classes defined around this concept.

One example is the class of **domishold** graphs, which was introduced by Benzaken and Hammer in [BH78]. These are the graphs for which there are positive weights associated to the vertices of the graph such that a subset D of vertices is dominating if and only if the sum of the weights of the vertices of D exceeds a certain threshold t. In other words, the characteristic vectors of the dominating sets are exactly the zero-one solutions of a linear inequality, where the coefficients of the inequality correspond to the weights of the vertices.

This concept motivated Payan to define **equidominating** graphs ([Pay80]). Loosely speaking, these are the graphs for which the characteristic vectors of the minimal dominating sets are the zero-one solutions of a linear equality. Formally, equidominating graphs have the following definition.

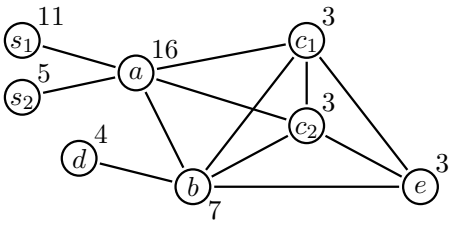


Figure 1: An equidominating graph on 8 vertices; the weights are drawn next to the vertices and the target value is $t = 23$.

Definition 1. *A graph $G = (V, E)$ is called* ***equidominating*** *if there exists a value $t \in \mathbb{N} = \{1, 2, 3, \ldots\}$ and a weight function $\omega \colon V \to \mathbb{N}$ such that for all $D \subseteq V$ the following equivalence holds:*

$$D \text{ is an mds} \iff \omega(D) := \sum_{v \in D} \omega(v) = t \ .$$

The pair (ω, t) is called an ***equidominating structure****, ω an* ***equidominating function*** *and t a* ***target value****.*

Figure 1 shows an equidominating graph. Every mds has a total weight of 23 and further, every subset of weight 23 is an mds. One advantage of having an equidominating structure of the graph at hand is that one can check whether a given vertex subset is an mds in $O(n)$ time, when n is the number of vertices.

The Equidomination problem is to decide whether a given graph is equidominating or not. Unfortunately, the computational complexity of this problem is unknown. It is not even clear whether Equidomination is actually in NP.

It can be seen that the following problem is coNP-complete: given a graph G, a weight function ω and some number $t \in \mathbb{N}$, the question is: is (ω, t) an equidominating structure of G? This remains true in the seemingly simple case when G is just the disjoint union of edges. That can be seen by applying literally the same reduction from the Weak Partition problem given by Milanič et al. ([MOR11]), who proved coNP-completeness for the the analogous problem of equistability.

We remark that there is no characterization of the class of equidominating graphs in terms of forbidden induced subgraphs: if one attaches a pendant vertex to every vertex of an arbitrary graph (the so-called corona of a graph with K_1), the resulting graph is equidominating. The only existing result in this direction (see Theorem 2 in [Pay80]) characterizes graphs that are both equidominating and domishold. Further, it is shown in [Pay80] that threshold graphs are equidominating.

To get a grip on the computational complexity of the problem, we introduce the following two parameterized notions of equidomination.

Definition 2. *For a given $t \in \mathbb{N}$ a graph $G = (V, E)$ is called* ***target-t equidominating*** *if there is an equidominating structure of the form (ω, t) for G.*

Definition 3. *For a given $k \in \mathbb{N}$ a graph $G = (V, E)$ is called* ***k-equidominating*** *if there exists an equidominating structure (ω, t) with $\omega \colon V \to \{1, \ldots, k\}$ for some $t \in \mathbb{N}$. In this case, ω a* ***k-equidominating function*** *and the pair (ω, t) is said to be a* ***k-equidominating structure****.*

Note that a k-equidominating graph is also k'-equidominating for all $k' \geq k$. It is clear that every target-t equidominating graph is also t-equidominating since every vertex is contained in some mds and thus its weight cannot exceed t. The opposite, however, is not true. Indeed, the edgeless graph on $t+1$ vertices is k-equidominating for every $k \in \mathbb{N}$ but not target-t equidominating.

In our paper, we study the following two parameterized versions of the EQUIDOMINATION problem:

k-EQUIDOMINATION:

Instance: A graph G and $k \in \mathbb{N}$.
Parameter: k.
Problem: Decide whether G is k-equidominating.

TARGET-t EQUIDOMINATION:

Instance: A graph G and $t \in \mathbb{N}$.
Parameter: t.
Problem: Decide whether G is target-t equidominating.

We show that both the k-EQUIDOMINATION problem and the TARGET-t EQUIDOMINATION problem are fixed-parameter tractable (FPT). We do this by using the so-called kernelization technique: we construct a regarding the considered problem equivalent subgraph the size of which is bounded by a function of the fixed parameter. To obtain the kernels we we first decompose a given graph into blocks such that vertices of different blocks cannot have the same weights in an equidominating structure. The decomposition is based on the so-called twin relation. Secondly we apply four different reduction rules to the blocks.

Further, we give FPT-algorithms for both problems by applying an XP-algorithm to the kernels. The XP-algorithms has a running time of $\mathcal{O}\left(nm^2 + n^k k^k + n^{2k+2}k^{-k-1} + k^{3k+3}\right)$ for the k-EQUIDOMINATION problem and uses the decomposition, too. It can also be used for the TARGET-t EQUIDOMINATION problem. The algorithm mainly follows the ideas and the algorithm for the k-EQUISTABILITY problem of Levit et al. ([KMS16, LMT12]). However, it has to be extended in the equidominating case. For a given graph on n vertices and m edges, the construction of the kernel and the application of the XP-algorithm leads to a running time of $\mathcal{O}(nm^2 + n^2 + 4^{t^2+t}t^{2(t+1)})$ for the TARGET-t EQUIDOMINATION problem and $\mathcal{O}(nm^2 + n^2 + 4^{k^2+k}k^{8(k+1)})$ for the k-EQUIDOMINATION problem.

References

[BH78] C. Benzaken and P.L. Hammer. Linear separation of dominating sets in graphs. In B. Bollobs, editor, *Advances in Graph Theory*, volume 3 of *Annals of Discrete Mathematics*, pages 1 – 10. Elsevier, 1978.

[KMS16] Eun Jung Kim, Martin Milanič, and Oliver Schaudt. Recognizing k-equistable graphs in fpt time. *Graph-Theoretic Concepts in Computer Science: 41st International Workshop, WG 2015, Garching, Germany, June 17-19, 2015, Revised Papers*, pages 487–498, 2016.

[LMT12] Vadim E. Levit, Martin Milanič, and David Tankus. On the recognition of k-equistable graphs. In *Graph-Theoretic Concepts in Computer Science*, volume 7551

of *Lecture Notes in Computer Science*, pages 286–296. Springer Berlin Heidelberg, 2012.

[MOR11] Martin Milanič, James Orlin, and Gábor Rudolf. Complexity results for equistable graphs and related classes. *Ann. Oper. Res.*, 188:359–370, 2011.

[Pay80] Charles Payan. A class of threshold and domishold graphs: equistable and equidominating graphs. *Discrete Mathematics*, 29(1):47–52, 1980.

Scheduling of Time-Dependent Asymmetric Nonmonotonic Processing Times permits an FPTAS

Helmut A. Sedding[1]

[1]Ulm University, Institute of Theoretical Computer Science, 89069 Ulm, Germany

In classical scheduling models, the given processing times are constant. In the field of time-dependent scheduling, the processing time of a job is defined by a function of start time. Here, most portrayed models are restricted on monotonic processing time functions. Recently, a model with an absolute value function is introduced and shown to be NP-hard in its decision version. We extend this model to allow for asymmetric and job-specific slopes. For problem instances with agreeable conditions it is possible to derive an FPTAS. This closes the gap to known monotonic piecewise-linear models in the literature.

1 Introduction

Research on single machine scheduling focuses on the optimization of a job sequence for a given objective. In the case of variable processing times, a common objective is to minimize the makespan of the schedule. In time-dependent scheduling, as reviewed in [4], the processing time $p_j(t_j)$ of each job j depends on the job's start time t_j. Let us consider processing time functions in the form of $p_j(t_j) = l_j + q_j(t_j)$. If q_j is monotonic and equal for all jobs, the sequencing problem is solvable in polynomial time by sorting the jobs nonincreasingly by l_j [12]. For job-specific q_j, the monotonic, linear form $q_j(t_j) = b_j\, t_j$ with $b_j > 0$ is also solved by sorting the jobs, in this case nonincreasingly by l_j/b_j [1]. Allowing for the piecewise-linear, job-specific, monotonic function $q_j(t_j) = \max\{0,\, b_j\,(t_j - T_j)\}$ for a given T_j, the problem becomes NP-hard (as usual, shown for the decision version), but permits a fully polynomial time approximation scheme (FPTAS) [2, 9, 10]. The symmetric problem $q_j(t_j) = \max\{-a_j\,(t_j - T_j),\, 0\}$, with $a_j \in (0, 1)$, behaves likewise: it is NP-hard and has an FPTAS [3, 7].

These well known monotonic forms are joined in $\tilde{q}_j(t_j) = \max\{-a_j\,(t_j - T_j),\, b_j\,(t_j - T_j)\}$. As visualized in Fig. 1, this processing time function is non-monotonic, convex, job-specific and piecewise-linear. Starting a job earlier or later than at the ideal time T_j results in an increased processing time. This is similar to the classic early/tardy objective, where due date deviation is minimized. However, we may note that the transfer of results is nontrivial. See, for example, the effect of swapping two jobs. In early/tardy scheduling, this has no effect on the timing of other jobs. Whereas if the jobs change their processing time, this leads to a ripple effect, as the timing of all succeeding jobs changes.

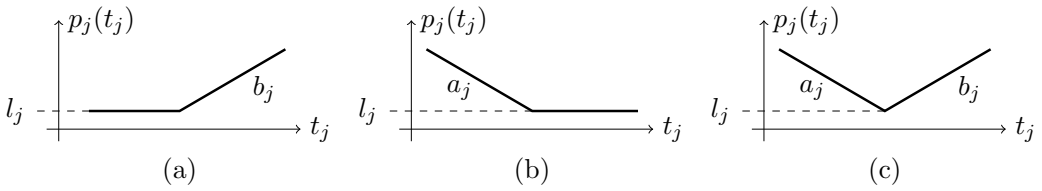


Figure 1: Piecewise-linear processing time functions in the literature are mostly monotonic, either nondecreasing (a) or nonincreasing (b). We combine both models to a convex, non-monotonic, piecewise-linear processing time (c).

In the literature, a special case of this model was introduced in [13]. They consider the symmetric case with uniform $a = a_j = b_j$ for all jobs j. A similar model is studied in [6]. Here as well, the processing time is specified by an absolute value function. However, the deviation is not measured from the start time. Instead, the midtime is used, which lies in the middle between the job's resulting start and completion time. By this, a job behaves symmetric before and after the ideal time. For this reason, the sequencing problem with a variable start time is solved by alternately appending and prepending jobs around the ideal time in nondecreasing order of l_j. The corresponding problem in classic scheduling is the early/tardy objective with a nonrestrictive, i.e., large common due date, and it is solved by the same procedure [8]. On the other hand, the variant of a restrictive common due date is NP-hard [5]. This is also the case in the described time-dependent problem with a fixed start time. Here, NP-hardness is shown by reduction of the even-odd partition problem [6].

We aim for a deeper understanding of the problem and address the classification of its complexity. This motivates the search for an approximation algorithm. The monotonic variant of our problem, with either $a_j = 0$ or $b_j = 0$, is NP-hard permit an FPTAS [7, 9, 14]. In fact, it is possible to extend on the scheme of [14] to construct an FPTAS for both $a_j > 0$ and $b_j > 0$. Moreover, it allows for job-specific a_j and b_j if they are agreeable. The term *agreeable* was coined in [11] for the weighted tardy scheduling problem $1 \mid\mid \sum_j w_j T_j$. They observe that the problem is solvable in pseudopolynomial time only with agreeable weights, else it is NP-hard in the strong sense.

2 Problem definition and properties

In this section, we define the studied single machine time-dependent scheduling problem $\mathcal{P}$ and describe several properties.

Definition 1 ($\mathcal{P}$: $1\big|p_j = l_j + \max\{-a\,(t_j - T),\, b\,(t_j - T)\}\big|C_{\max}$)**.** *Given factor $a \in (0,1) \cap \mathbb{Q}$, $b \in \mathbb{Q}_{\geq 0}$, an ideal start time T, a set of n jobs J, and for each job $j \in J$ a base length $l_j \in \mathbb{Q}_{\geq 0}$.*

A variable job sequence $S : J \to \{1, \ldots, n\}$ assigns each job to a position. The completion time of a job $j \in J$, for start time t_j, is

$$C_j(t_j) = t_j + l_j + \max\{-a\,(t_j - T),\, b\,(t_j - T)\}.$$

The first job $j \in J$ in the sequence, $S(j) = 1$, starts at time 0, i.e., $t_{S^{-1}(1)} = 0$. The succeeding jobs $j \in J$, $S(j) > 1$, start at the completion time of the predecessor, i.e., $t_{S^{-1}(i)} = C_{S^{-1}(i-1)}$. The objective is to minimze makespan $C_{\max} = C_{S^{-1}(n)}$.

The completion time function $C_j(t_j)$ is monotonically increasing. This eliminates the need for idle time, which would increase the objective value.

If all jobs $j \in J$ are either early ($t_j < T$) or tardy ($t_j \geq T$), the makespan can be calculated in a closed formula.

Property 2. *Given an instance of $\mathcal{P}$ and a sequence S that starts at a time $\tilde{t}$.*

(a) Let

$$X_J(\tilde{t}) = \tilde{t} \cdot \prod_{j \in J} (1 - a_j)^{-1} + \sum_{j \in J} l_j \cdot \prod_{k \in J,\, S(k) \geq S(j)} (1 - a_k)^{-1}. \tag{1}$$

If $X_J(\tilde{t}) \leq T$, then $C_{\max} = X_J(\tilde{t})$.

(b) Let

$$Z_J(\tilde{t}) = \tilde{t} \cdot \prod_{j \in J} (1 + b_j) + \sum_{j \in J} l_j \cdot \prod_{k \in J,\, S(k) \geq S(j)} (1 + b_k). \tag{2}$$

If $\tilde{t} \geq T$, then $C_{\max} = Z_J(\tilde{t})$.

For an arbitrary ideal time T, we divide the job set J. One subset A is scheduled before T, a *straddler job* $\chi \in J$ starts at or before T and completes at or after T, and subset B is scheduled after T. Then again, the makespan can be expressed by a closed formula.

Property 3. *Given a sequence S for an instance of $\mathcal{P}$ with a straddler job $\chi \in J$. This yields job sets $A = \{j \in J \mid S(j) < S(\chi)\}$ and $B = \{j \in J \mid S(j) > S(\chi)\}$. If $X_A(0) \leq T$ and $C_\chi(X_A(0)) \geq T$, then $C_{\max} = Z_B(C_\chi(X_A(0)))$.*

Definition 4. *Given an instance of $\mathcal{P}$. If there exists sequence S of the jobs J where $l_{j-1}/a_{j-1} \leq l_j/a_j \iff l_{j-1}/b_{j-1} \leq l_j/b_j$ for all $i = 2, \ldots, n$, we say that the instance is agreeable.*

Assume for any agreeable instance that the jobs $J = \{1, \ldots, n\}$ are indexed such that $l_{j-1}/a_{j-1} \leq l_j/a_j$ and $l_{j-1}/b_{j-1} \leq l_j/b_j$ for all $i = 2, \ldots, n$.

Property 5. *Given an agreeable instance of $\mathcal{P}$. If the reverse sequence $n, \ldots, 1$ satisfies the conditions in Prop. 2(a), then it is an optimum sequence that minimizes $C_{\max}$. Analogously, sequence $1, \ldots, n$ is optimum if it fulfills Prop. 2(b).*

To minimize the makespan of agreeable instances, we combine the result of Prop. 3 and Prop. 5. The minimum job sequence for job set A (and, respectively, B) sorts the jobs according to Prop. 5 with job subset A (and B). We thus need to decide on a straddler job $\chi \in J$, and partition the remaining jobs $J \setminus \{\chi\}$ into early jobs A and tardy jobs B.

3 FPTAS

We begin by introducing a dynamic programming algorithm $\mathcal{A}$ for agreeable instances of $\mathcal{P}$. It returns the optimum makespan $C^*_{\max}$ and a partition of the job set J into the sets $\{\chi\}, A, B$, where χ denotes the straddler job. We repeat the following procedure for each $\chi \in J$. In each iteration $j = 1, \ldots, n$, the algorithm creates at most two states: one by inserting job j into set B, and a second by inserting j into A if it completes at or before T. A dynamic programming state is represented by a vector $[x, y, z]$ of nonnegative rational numbers. Here, x is the makespan of the jobs in set A. The z value is the makespan of the jobs in set B when

starting at T; changing their start time as in Eq. (2) scales z by factor y. After adding all jobs, we calculate the makespan for each resulting partition $\{\chi\}, A, B$ as in Prop. 3. A partition with the minimum makespan solves the given instance.

We turn $\mathcal{A}$ into an approximation algorithm $\mathcal{A}_\varepsilon$ for arbitrary $\varepsilon \in (0,1)$ by trimming the states in each iteration. This principle is described, e.g., in [14]. A challenge is the exponentially increasing z component (for an intuition, see Eq. (2)). However, we cope this by spacing the trimming intervals exponentially increasing as well. Then, the number of states and, accordingly, the runtime is bounded by a polynomial of the input size and $1/\varepsilon$. Nontheless, $\mathcal{A}_\varepsilon$ yields a partition $\{\chi\}, A, B$ with makespan $C_{\max} \leq (1+\varepsilon)\, C^*_{\max}$. Our main result follows.

Theorem 6. *Algorithm $\mathcal{A}_\varepsilon$ is a FPTAS for agreeable instances of $\mathcal{P}$.*

References

[1] S. Browne and U. Yechiali. Scheduling deteriorating jobs on a single processor. *Operations Research*, 38(3):495–498, 1990.

[2] J.-Y. Cai, P. Cai, and Y. Zhu. On a scheduling problem of time deteriorating jobs. *Journal of Complexity*, 14(2):190–209, 1998.

[3] T. C. E. Cheng, Q. Ding, M. Y. Kovalyov, A. Bachman, and A. Janiak. Scheduling jobs with piecewise linear decreasing processing times. *Naval Research Logistics*, 50(6):531–554, 2003.

[4] S. Gawiejnowicz. *Time-dependent scheduling*. Monographs in Theoretical Computer Science. Springer, 2008.

[5] N. G. Hall, W. Kubiak, and S. P. Sethi. Earliness–tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date. *Operations Research*, 39(5):847–856, 1991.

[6] F. Jaehn and H. A. Sedding. Scheduling with time-dependent discrepancy times. *Journal of Scheduling*, 19(6):737–757, 2016.

[7] M. Ji and T. C. E. Cheng. An FPTAS for scheduling jobs with piecewise linear decreasing processing times to minimize makespan. *Information Processing Letters*, 102(2-3):41–47, 2007.

[8] J. J. Kanet. Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly*, 28(4):643–651, 1981.

[9] M. Y. Kovalyov and W. Kubiak. A fully polynomial approximation scheme for minimizing makespan of deteriorating jobs. *Journal of Heuristics*, 3(4):287–297, 1998.

[10] W. Kubiak and S. L. van de Velde. Scheduling deteriorating jobs to minimize makespan. *Naval Research Logistics*, 45(5):511–523, 1998.

[11] E. L. Lawler. A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1:331–342, 1977.

[12] O. I. Mel'nikov and Y. M. Shafranskiĭ. Parametric problem in scheduling theory. *Cybernetics*, 15 (3):352–357, 1979.

[13] H. A. Sedding and F. Jaehn. Single machine scheduling with nonmonotonic piecewise linear time dependent processing times. In *Proceedings of the 14th International Conference on Project Management and Scheduling*, pages 222–225. TU München, 2014.

[14] G. J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing*, 12(1): 57–74, 2000.

A path-based formulation for the Hydro Unit Commitment and Scheduling problem

Dimitri Thomopulos[1], Wim van Ackooij[2], Claudia D'Ambrosio[1], Leo Liberti[1], Raouia Taktak[3], and Sonia Toubaline[4]

[1]LIX CNRS (UMR7161), École Polytechnique, 91128 Palaiseau Cedex, France
[2]EdF R&D, OSIRIS, France
[3]ISIMS&CRNS, Université de Sfax, Tunisie
[4]Université Paris-Dauphine, PSL Research University, CNRS, LAMSADE, 75016 Paris, France

In this paper, we study a single-reservoir Hydro Unit Commitment Problem in a deterministic price-taker context, where production is assumed to be generated through discrete operational points. Under some hypotheses, we present a time expanded graph representation for the problem, where, at each time step, nodes correspond to the discrete operational points, and arcs refer to possible state changes. Based on the graph representation, we show that our problem reduces to a Constrained Shortest Path Problem.

Keywords: Hydro Unit Commitment Problem, time expanded graph representation, Constrained Shortest Path Problem

1 Introduction

In energy management, unit commitment problems are strategic in day-ahead operations (e.g., [4]). In countries where hydro-generation is abundant (e.g., France, Brazil, Canada) specifically dealing with the optimization of cascaded reservoirs is quite challenging too (e.g., [5]). The reason for this is that quite some modelling detail can be required to accurately model reality. One of the essential difficulties stems from representing the efficiency curves linking the flow rate with the actual amount of generated power.

One particular way of dealing with this difficulty is by disposing of an *a priori* discretization of it. The specific set of points is typically chosen by an operational team in order to have maximal efficiency (highest derivatives). One advantage is that taking primary spinning reservoirs into account becomes straightforward. This strongly contrasts with the non-linear programming approach wherein differences of mappings need to be accounted for. Several constraints such as sufficient stability of adjacent flow rates (ruling out a swift increase followed by a subsequent decrease of flow) can also be formulated more easily. When allowing for a continuous flow rate, such a constraint would become akin to complementarity constraints.

In this paper we will focus on the deterministic price-taker model with a single reservoir and potentially many discrete operational points for the underlying units. In particular, we propose a path formulation and show that, under some assumptions, it is equivalent to classic mixed integer programming models that can be found in the literature, see, for example, [1].

2 A path based modelling approach

We consider the single reservoir Hydro Unit Commitment Problem (HUCP) under some hypothesis: i) without head effect, i.e., without considering the non-linear effect of the level of the uphill reservoir on the efficiency ii) discrete operational points (o.p.) for the generated production. In particular, we focus on a single reservoir composed of several units $J = \{1, 2, \ldots, \bar{n}\}$ that can be either pumps or turbines and that the different units are ordered, thus can be aggregated in a unique unit.

2.1 Graph modelling

The main idea is to model the operational profiles of the aggregated units we are considering by means of a graph that includes all the possible operational points at each time step, see Figure 1. In this figure, for each time step we represent operational points of the different units of the reservoir, assumed to be ordered. Given T the number of steps in the time horizon, the graph shows $T+1$ periods of time $0, 1, 2, \ldots, T$ and a additional fictive period, $T+1$, that will be used in the sequel.

More formally, let us denote by $G = (N, A)$ the graph of Figure 1, where N is the set of operational points at each time step and A is the set of possible arcs between nodes of N. G is a weighted directed acyclic graph. For clarity we present the graph construction for the case in which, at each time step, the operational points are the same. However it is easy to generalize it to the case in which each time period has different operational points. Supposing to have $\bar{z}$ operational points at each time step, we will have $|N| = \bar{z}T + 2$, i.e., $\bar{z}$ nodes for each time step and 2 artificial nodes that represent time steps 0 and $T+1$, respectively. In the following, these two nodes will be also called source $s = 0$ and destination $d = T\bar{z} + 1$, respectively, as they represent the initial and final node of the path representing the operational profile of the unit.

The set of nodes N can be partitioned as follows: $\{0\} \cup \bigcup_{t=1}^{T} N_t \cup \{T\bar{z} + 1\}$ where $N_t = \{(t-1)\bar{z}+1, (t-1)\bar{z}+2, \ldots, t\bar{z}\}$ represent the nodes that correspond to the operational points at period t, $t \in \{1, \ldots, T\}$. Thus, arc $(i, j) \in A$ if $i \in N_t$, $j \in N_{t+1}$ and if it is possible to move from the operational point corresponding to node i to the operational point corresponding to j without violating a subset of the physical constraints of the problem. In particular, all the constraints of the considered problem (see e.g. [1]) are automatically included in the graph structure except: i) the bounds on the water volume at each time step, and ii) the target water volume in the reservoir, i.e., the minimum amount of water volume that has to be reached at the end of the time horizon. Finally, a water flow is associated to each node and a cost to every edge equal to the difference between the unit startup costs and the power selling.

A feasible solution of the hydro unit commitment problem is represented by a path in G between 0 and $T + 1$, see Figure 1. Note that the path consists of exactly $T + 1$ arcs. Thus, the single reservoir HUCP, under the assumptions previously mentioned, reduces to a Shortest Path Problem (SPP) from s to d with extra constraints, i.e., bounds on water volume of the reservoir.

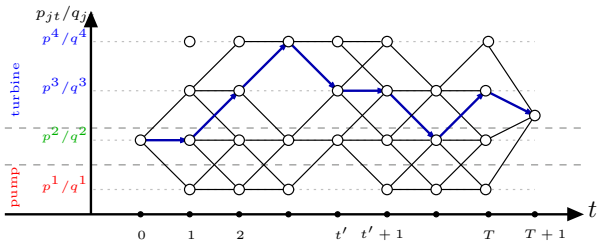


Figure 1: $G = (N, A)$

The Constrained Shortest Path Problem has largely been studied in the literature ([3]). The problem can be stated as a minimum-cost path problem subject to one or more resource constraints, a problem widely used in many Branch-and-Price algorithms (see [2]). The (Resource) Constrained Shortest Path Problem ((R)CSPP) constitutes in fact the pricing subproblem of many classical problems.

Different solutions methods have been proposed to solve the (R)CSPP. In all of these approaches, the resource function is assumed to be monotonically decreasing. This is not the case of our water volume constraints. Therefore, these methods cannot be used as defined to solve HUCP. In the following sections, we propose two approaches to solve it.

2.2 Integer Linear Programming formulation

Let us consider the weighted directed acyclic graph $G = (N, A)$ described in Section 2.1. Denote by:

- I_t = predicted water inflow in period t $(t = 1, \ldots, T)$ [m^3/s].
- Δt = period duration [s].
- $[\underline{V}, \overline{V}]$ = lower and upper bounds on water volume in the reservoir [m^3].
- V_T = target water volume in reservoir at the end of the time horizon [m^3].
- V_0 = initial water volume in reservoir [m^3].
- Q_i = water flow corresponding to node i $(i \in N)$ [m^3/s], where $Q_i = 0$ for $i \in N_{T+1}$.
- c_{ij} = cost of the arc $i(i, j) \in A$, depending on the cost of power generated or consumed, on the start up costs of units, and on the pumping cost.

Let x_{ij} $(\forall (i, j) \in A\ )$ be a binary variable defined as follows:

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is part of the selected path.} \\ 0 & \text{otherwise} \end{cases}$$

The short-term single reservoir HUCP is modelled by the following integer linear program

$$\min_{x} \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\sum_{(j,i)\in A} x_{ji} - \sum_{(i,j)\in A} x_{ij} = \begin{cases} -1 & \text{if} \quad i = s \\ 0 & \text{if} \quad i \in N \setminus \{s,d\} \\ 1 & \text{if} \quad i = d \end{cases} \qquad \forall i \in N \tag{1}$$

$$\underline{V} \leq v_0 + \Delta t \sum_{k=1}^{t} \left(I_k - \sum_{j\in N_k} \sum_{(i,j)\in A} Q_j x_{ij} \right) \leq \overline{V} \qquad \forall t \in 1,\ldots,T \tag{2}$$

$$v_0 + \Delta t \sum_{k=1}^{T} \left(I_k - \sum_{j\in N_k} \sum_{(i,j)\in A} Q_j x_{ij} \right) \geq V_T \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{4}$$

Constraints (1) are the flow conservation constraints that ensure a path between s and d. Constraints (2) model the bounds on the water volume in the reservoir at time step t. Finally, constraint (3) represents the minimum target water volume to be reached at the end of the time horizon.

2.3 Expanded graph reformulation including the volume dimension

The second approach that we propose is a reformulation of the graph $G = (N, A)$ considering an additional dimension representing the volume. As we mentioned in Section 2.1, volume constraints are not automatically considered in the graph. Thus, we propose to discretize the volume and to add a volume dimension to the graph, that represents the water volume value in the reservoir at time step t. In this way, all the considered constraints can be included in the new graph structure $G^* = (N^*, A^*)$, where N^* is the set of couples of operational points and feasible volume values at each time step and A^* is the set of possible arcs between nodes of N^*. Finally, we propose to use standard graph techniques to solve the SPP in $G^* = (N^*, A^*)$.

References

[1] A. Borghetti, C. D'Ambrosio, A. Lodi, and S. Martello. A MILP approach for short-term hydro scheduling and unit commitment with head-dependent reservoir. *IEEE Transactions on Power Systems*, 23(3):1115–1124, 2008.

[2] D. Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *4OR*, 8(4):407–424, 2010.

[3] L. Di Puglia Pugliese and F. Guerriero. A survey of resource constrained shortest path problems: Exact solution approaches. *Networks*, 62(3):183–200, 2013.

[4] M. Tahanan, W. van Ackooij, A. Frangioni, and F. Lacalandra. Large-scale unit commitment under uncertainty: a literature survey. *4OR*, 13(2):115–171, 2015.

[5] R. Taktak and C. D'Ambrosio. An overview on mathematical programming approaches for the deterministic unit commitment problem in hydro valleys. *Energy Systems*, 8(1):57–79, 2017.

k-metric antidimension in graphs and anonymity in social networks

Ismael G. Yero[1]

[1]Department of Mathematics, EPS Algeciras, University of Cadiz, Spain

Given a connected graph $G = (V, E)$, and an ordered set of vertices $S \subseteq V$, the metric representation of a vertex $u \in V$ with respect to S is the vector of distances between u and the vertices of S. The set S is a k-antiresolving set if k is the largest positive integer such that for every vertex $v \in V - S$ there exist at least other $k - 1$ different vertices $v_1, \ldots, v_{k-1} \in V - S$ such that $v, v_1, \ldots, v_{k-1}$ have the same metric representation with respect to S. The k-metric antidimension of G is the minimum cardinality among all the k-antiresolving sets for G. An application of k-antiresolving sets to privacy in social networks is described in this work. Moreover, several computational and complexity results concerning the k-metric antidimension of graphs are presented.

1 Introduction

Graph Theory and Social Sciences have lately witnessed a highly significant growth of their relationships. These connections arise because of a huge popularity of social networks. For instance, Facebook, Twitter, LindkedIn and similar ones have undergone an accelerated evolution and nowadays, society would not properly behaves without such virtual platforms, causing an urgent demand of its study. The power of social network analysis is indeed questionless. It might uncover previously unknown knowledge such as community-based problem, media use, individual engagement, etc. However, all these benefits and entertainments are not cost-free. A not well-behaved individual could compromise users privacy using the social network for harmful purposes, which would result in the disclosure of sensitive data such as e-mails, sicknesses or relationships. Natural actions to avoid this consist of an anonymization process of the social network. However, not always such anonymization process succeeds and the goal is then to at least measuring how much privacy the social network achieves. Concerning the latter, in this work we present some results aimed at evaluating the resistance of a social graph against active attacks to its privacy. A new and meaningful privacy measure for social networks has been recently introduced and theoretically studied (see [4]). Such privacy measure arises from the concept of k-metric antidimension of graphs.

Given a simple and connected graph $G = (V, E)$, and an ordered set of vertices $S = \{w_1, \ldots, w_t\} \subseteq V$, the *metric representation* of a vertex $u \in V$ with respect to S is the t-vector $r(u|S) = (d_G(u, w_1), \ldots, d_G(u, w_t))$, where $d_G(u, v)$ represents the length of a shortest $u - v$ path in G. The set S is a *k-antiresolving set* if k is the largest positive integer such that for every vertex $v \in V - S$ there exist at least other $k-1$ different vertices $v_1, \ldots, v_{k-1} \in V - S$

such that $v, v_1, \ldots, v_{k-1}$ have the same metric representation with respect to S. The *k-metric antidimension* of G is the minimum cardinality among all the k-antiresolving sets for G. Since there could not be a k-antiresolving set in a graph G for some values of k, a graph G is said to be *k-metric antidimensional* if k is the largest integer such that G contains a k-antiresolving set. Moreover, we notice that even so a graph G would be k'-metric antidimensional for some integer k', this does not imply that there will be k-antiresolving sets for every $k \leq k'$. Concepts above were introduced in [4] and further studied in [1, 2, 3].

If a set of attacker nodes S is a k-antiresolving set, then an adversary controlling the vertices of S cannot uniquely re-identify other nodes in the network with probability higher than $1/k$. However, given that S is unknown, any privacy measure for a social network should quantify over all possible subsets S. In this sense, a social network G meets (k, ℓ)-*anonymity* with respect to active attacks to its privacy, if k is the smallest positive integer such that the k-metric antidimension of G is lower than or equal to ℓ.

2 Results

In [1], a different approach for the k-metric antidimension of graphs was presented. Let $S = \{v_1, \ldots, v_r\}$ be an ordered subset of vertices of a graph $G = (V, E)$ and for any vertex $x \notin S$, consider the vector of distances $d_{x,S} = (d_G(x, v_1), \ldots, d_G(x, v_r))$ from x to the vertices in S. The equality relation over a set of such vectors, all of same length, is readily seen to define an equivalence relation. The following notations are used for such an equivalence relation over the set of vectors $\mathcal{D}_{V\setminus S,-S}$.

- The set of equivalence classes, which forms a partition of $\mathcal{D}_{V\setminus S,-S}$, is denoted by $\Pi^{=}_{V\setminus S,-S}$.
- Two nodes $v_i, v_j \in V \setminus S$ belong to the same equivalence class if $d_{v_i,-S}$ and $d_{v_j,-S}$ belong to the same equivalence class in $\Pi^{=}_{V\setminus S,-S}$, and thus $\Pi^{=}_{V\setminus S,-S}$ also defines a partition into equivalence classes of $V \setminus S$.
- The *measure* of the equivalence relation above is defined as $\mu\left(\mathcal{D}_{V\setminus s,-s}\right) = \min_{\mathcal{Y}\in\Pi^{=}_{V\setminus S,-S}} \left\{ \mid \mathcal{Y} \mid \right\}$.

Clearly, if S is a k-anti-resolving set, then $\mathcal{D}_{V\setminus S,-S}$ defines a partition into equivalence classes whose measure is *exactly* k.

In concordance with the k-metric antidimension of graphs, three problems were detected and studied from a computational point of view in [1]. The problems are the following ones, for which $G = (V, E)$ represents a connected simple graph.

PROBLEM 1: [metric anti-dimension or ADIM] Given G, find a subset of nodes $S \subset V$ that maximizes $\mu\left(\mathcal{D}_{V\setminus S,-S}\right)$.

The Problem 1 simply finds a k-anti-resolving set for the largest possible k. As an interpretation, a solution of it establishes a bound on the privacy violation probability of an adversary.

PROBLEM 2: [$k_{\geq}$-metric anti-dimension or ADIM$_{\geq k}$] Given G and a positive integer k, find a subset of nodes S of minimum cardinality such that $\mu\left(\mathcal{D}_{V\setminus S,-S}\right) \geq k$, if such a S exists.

The Problem 2 finds a k'-anti-resolving set of minimum possible cardinality for some specific $k' \geq k$. In contrast, the next problem finds a k-anti-resolving set of minimum possible

cardinality.

PROBLEM 3: [$k_{=}$-metric antidimension or ADIM$_{=k}$] Given G and a positive integer k, find a subset of nodes S of minimum cardinality such that $\mu\left(\mathcal{D}_{V\setminus S,-S}\right) = k$, if such a S exists.

All the problems above were studied in [1], where the following results were presented. As we can see, the two first problems can be efficienttly solved, while the third one belong to the hardest class of problems.

Theorem 1. [1]

- *Both ADIM and ADIM$_{\geq k}$ can be solved in $O\left(n^4\right)$ time.*
- *Both ADIM and ADIM$_{\geq k}$ can also be solved in $O\left(\frac{n^4 \log n}{k}\right)$ time "with high probability" (with a probability of at least $1 - n^{-c}$ for some constant $c > 0$.*

Theorem 2. [1]

- *ADIM$_{=k}$ is NP-complete for any integer k in the range $1 \leq k \leq n^{\epsilon}$ where $0 \leq \epsilon < \frac{1}{2}$ is any arbitrary constant, even if the diameter of the input graph is 2.*
- *Assuming NP is not a subset of the class* DTIME$\left(n^{\log\log n}\right)$*, there exists a universal constant $\delta > 0$ such that ADIM$_{=k}$ does not admit a $\left(\frac{1}{\delta}\ln n\right)$-approximation for any integer k in the range $1 \leq k \leq n^{\epsilon}$ where $0 \leq \epsilon < \frac{1}{2}$ is any arbitrary constant, even if the diameter of the input graph is 2.*

Although ADIM$_{=k}$ is NP-complete, some cases are shown to be efficiently solved, or on the other side, the problem can be approximated as the next result shows.

Theorem 3. [1]

- *ADIM$_{=1}$ admits a $(1 + \ln(n-1))$-approximation in $O\left(n^3\right)$ time.*
- *If G has at least one node of degree 1, then ADIM$_{=1}$ can be solved in $O\left(n^3\right)$ time.*
- *If G does not contain a cycle of 4 edges, then ADIM$_{=1}$ can be solved in $O\left(n^3\right)$ time.*

A particular problem which relates to the ones above consists of finding the graphs G in which $k = 1$ is the maximam value for k such that G is k-metric antidimensional. Of course, in such case every set of the graph is a k-antiresolving set, which means two things. The first one is that the 1-metric antidimension of G is one, and the second one that, if an adversary controls at least one vertex of such graph, then with certainty the attacker can recognize at least one vertex of the graph. In order to avoid this, graphs that are 1-metric antidimensional should be correctly detected.

It is readily seen that the particular case of 1-metric antidimensional graphs can only satisfy $(1, 1)$-anonymity [4]. In this sense, the privacy guarantees against active attacks, for the case of 1-metric antidimensional graphs represent the worst possible scenario. As a consequence, one should abstain of publishing compromising information in such networks. According to these facts, those classes of trees and unicyclic graphs that are 1-metric antidimensional were studied in [3].

Given a tree T rooted in a vertex u, for any vertex $v \in V(T)$, the sets $p(v)$, $C(v)$ and $D(v)$ are the parent, the children and the descendants of v in T. For a given vertex $v \in V(T)$ and

a set $A \subset C(v)$, $T_{v,A}$ represents the subtree of T induced by $\{v\} \cup A \cup \left(\bigcup_{u \in A} D(u)\right)$. Now, an *u-branch* of T at v is the subtree $T_{u,v}$ induced by $\{u, v\} \cup D(v)$. Accordint to this, two x-branches T_{x,y_1} and T_{x,y_2} are *ϵ-equivalent* if $\epsilon_{T_{x,y_1}}(x) = \epsilon_{T_{x,y_1}}(x)$, where $\epsilon_G(v)$ is the eccentricity of the vertex v in G.

For any vertex x of a tree T, the *balancing factor* $\xi_T(x)$ is the maximum number of ϵ-equivalent x-branches in T. If ϵ-equivalent x-branches do not exist in T, then it is assumed $\xi_T(x) = 1$.

Theorem 4. [3] *Let T be a tree. Then T is 1-metric antidimensional if and only if $\xi_T(v) = 1$ for every $v \in V(T)$.*

In concordance with the characterization above, a polynomial algorithm that checks whether a given tree is 1-metric antidmensional was given in [3]. Moreover, a polynomial algorithm to solve the same problem for the case of unicyclic graphs was also presented in [3]. This was based on the next result, which use the following terminology and notation. Given n trees $T_{v_1}, \ldots, T_{v_n}$ rooted in $v_1, \ldots, v_n$, respectively, and a cycle $C_G = \{u_1, \ldots, u_n\}$, we denote by $G = \{T_{v_1}, \ldots, T_{v_n}\}$ the unicyclic graph obtained by identifying the vertex v_i with u_i for every $i \in \{1, \ldots, n\}$. Given a vertex u of a graph G, let $d_i(u) = \{v \in V(G) : d(v, u) = i\}$ for every $i \in \{1, \ldots, \epsilon_G(u)\}$. The *antiresolving factor* of u in G is denoted by $\phi_G(u)$ and is defined as: $\phi_G(u) = \min_{1 \leq i \leq \epsilon_G(u)}\{|d_i(u)|\}$. The antiresolving factor of the whole graph G (denoted by $\phi(G)$) is: $\phi(G) = \max_{v \in V(G)}\{\phi_G(v)\}$.

Theorem 5. [3] *Let $G = \{T_{v_1}, \ldots, T_{v_n}\}$ be a unicyclic graph. G is 1-metric antidimensional if and only if G holds the following properties.*

- *For every $i \in \{1, \ldots, n\}$, it follows that T_{v_i} is 1-metric antidimensional.*
- *For every $i \in \{1, \ldots, n\}$, every $x \in V(T_{v_i})$ and every $Y \subseteq C(x)$ it follows that $\phi_{G^i_{x,Y}}(x) = 1$.*
- *If $|C_G|$ is even, then for every two adjacent vertices $a, b \in C_G$, it follows that $\xi_A(a) = 1$ or $\xi_B(b) = 1$, where A and B are the two trees obtained from G by deleting the edge (a, b) and its diametral edge.*
- *If $|C_G|$ is even, then for every two vertices $c, d \in C_G$ which are diametral in C_G, there exist two vertices $v_i, v_j \in C_G$ with $d(v_i, c) = d(v_j, c)$, such that $\epsilon_{T_{v_i}}(v_i) \neq \epsilon_{T_{v_j}}(v_j)$.*

References

[1] T. Chatterjee, B. DasGupta, N, Mobasheri, V. Srinivasan and and I. G. Yero, On the computational complexities of three privacy measures for large networks under active attack. arXiv:1607.01438v1 [cs.CC]. Submitted (2016).

[2] S. Mauw, R. Trujillo-Rasua , B. Xuan, Counteracting Active Attacks in Social Network Graphs. Lecture Notes in Computer Science 9766 (2016), 233–248.

[3] R. Trujillo-Rasua and I. G. Yero, Characterizing 1-metric antidimensional trees and unicyclic graphs. The Computer Journal 59 (8) (2016), 1264–1273.

[4] R. Trujillo-Rasua and I. G. Yero, k-metric antidimension: A privacy measure for social graphs. Information Sciences 328 (2016), 403–417.

Alphabetical list of authors